Министерство науки и высшего образования Российской Федерации

ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ СИСТЕМ УПРАВЛЕНИЯ И РАДИОЭЛЕКТРОНИКИ (ТУСУР)

Кафедра телевидения и управления

В.А. Кормилин

ВЫЧИСЛИТЕЛЬНАЯ ТЕХНИКА

Учебное пособие

2019

Кормилин В.А.
Вычислительная техника: Учебное пособие. – Томск: Томский государственный университет систем управления и радиоэлектроники (ТУСУР), 2019. – 140 с.

Пособие предназначено для студентов радиотехнических направлений подготовки ТУСУРа, обучающихся на всех формах обучения и содержит учебный материал и указания для организации работы при изучении дисциплины.

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Дисциплина «Вычислительная техника» предназначена для освоения принципов построения микропроцессорных устройств и применения микропроцессоров в различных радиотехнических устройствах и системах.

Разработка цифровых прикладных устройств на основе микропроцессоров, микроконтроллеров и микро-ЭВМ давно перестала быть уделом узкого круга специалистов, владеющих знаниями схемотехники и программирования микропроцессорных структур. В настоящее время использование МП для решения множества практических задач техники является насущной необходимостью.

В микропроцессорах – наиболее сложных микроэлектронных устройствах – воплощены самые передовые достижения инженерной мысли. В условиях свойственной данной отрасли производства жесткой конкуренции и огромных капиталовложений, выпуск каждой новой модели микропроцессора – так или иначе, связан с очередным научным, конструкторским, технологическим прорывом.

Поэтому важным делом является изучение принципов построения микропроцессоров, МП систем на их основе, а также методов программирования МП на низком, аппаратном уровне. Без этого невозможно обеспечить высокую грамотность будущих инженеров.

Изучение дисциплины связано с формированием компетенций:

ОПК-4 - способность применять современные компьютерные технологии для подготовки текстовой и конструкторско-технологической документации с учетом требований нормативной документации;

ПКС-1 – способность выполнять расчет и проектирование элементов и устройств инфокоммуникационных систем в соответствии с техническим заданием с использованием средств автоматизации проектирования.

1 МИКРОПРОЦЕССОРЫ И МИКРО-ЭВМ

1.1 Введение в микропроцессоры

Наиболее совершенной из разработанных цифровых систем является электронная вычислительная машина (ЭВМ) с хранимой программой. Гипотетический вычислитель, демонстрирующий логику и методику программирования, впервые был предложен Тьюрингом. При всем разнообразии современных ЭВМ, все они выполняют одни и те же основные функции: ввод информации, хранение, арифметическое и логическое преобразование, вывод информации и управление работой входящих в структуру устройств.

Микропроцессор **(МП)** – это набор компонент, реализованных в виде одной микросхемы или небольшого числа микросхем, выполняющих арифметические и логические операции над цифровыми данными и осуществляющих программное управление вычислительным процессом.

Микро-ЭВМ называют устройство обработки данных, содержащее один или несколько микропроцессоров, интегральные схем постоянной и оперативной памяти, схемы управления вводом/выводом информации и некоторые другие схемы (рисунок 1.1), связанные друг с другом с помощью сигнальных шин.

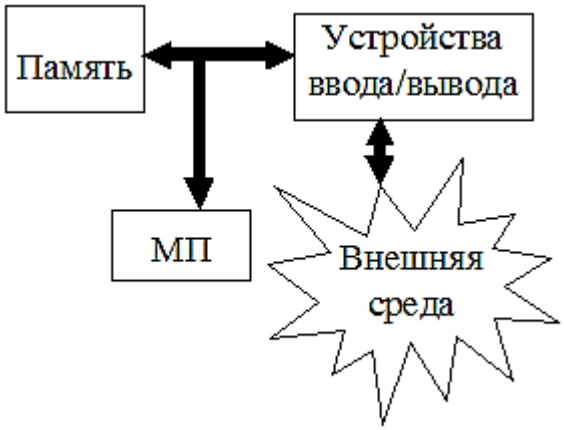

Рисунок 1.1 - Структура типичной микро-ЭВМ

В памяти находится ***программа*** (набор инструкций) и данные. Память обычно организована в виде совокупности M слов, содержащих по N бит информации каждое. Для обращения к определенному слову в памяти требуется указать его адрес, определяющий положение этого слова по отношению к другим словам.

При каждом обращении к памяти по определенному адресу МП, как правило, выполняет операцию записи или чтения одного слова памяти – данных или команды.

Сигнальная шина, предназначенная для передачи адресной информации, получила название ***шины адреса*** **(ША)**, а шина, через которую выполняется чтение и передача данных, часто называется ***шиной данных*** **(ШД)** (рисунок 1.2).

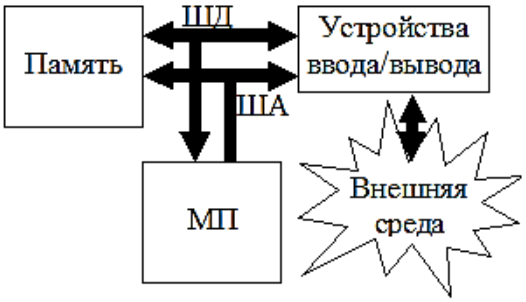

Рисунок 1.2 - Структура микро-ЭВМ с шинами данных и адреса

1.2 Типовая структура микропроцессора

Определим набор компонент, необходимых микропроцессору, для реализации присущих ему функций.

Арифметические и логические операции с данными в микропроцессоре выполняет ***арифметико-логическое устройство*** **(АЛУ)**. В АЛУ имеется два входа для операндов и один выход для результата, причем, чаще всего, операция выполняется одновременно над всеми битами каждого операнда. Типичный набор функций, реализуемых в АЛУ, включает в себя следующие операции:

a) сложение двух операндов;
b) вычитание операндов;
c) десятичная коррекция результата арифметической операции;
d) инкремент (увеличение на единицу) одного операнда;
e) декремент (уменьшение на единицу) одного операнда;
f) логическая функция И двух операндов;
g) логическая функция ИЛИ двух операндов;
h) логическая функция ИСКЛЮЧАЮЩЕЕ ИЛИ двух операндов;
i) логическая функция НЕ (инверсия) одного операнда;
j) сдвиг вправо на один разряд одного операнда;
k) сдвиг влево на один разряд одного операнда;
l) сдвиг по циклу вправо на один разряд одного операнда;

m) сдвиг по циклу влево на один разряд одного операнда.

При работе с операндами в АЛУ возникают различные состояния, связанные, например, с нулевым результатом, отрицательным результатом, переполнением разрядной сетки и т.п. Чтобы фиксировать эти события и оперативно влиять на ход обработки данных, в структуре МП используется специальный регистр, в котором значения отдельных бит соответствуют различным событиям: состояние переноса/заема, значение знака, нулевой результат и т.п. Такой регистр получил название ***слова состояния процессора*** **(ССП)**. Биты событий называют ***флажками (флагами)***, а регистр иногда называют ***регистром флагов***.

При выполнении операции в АЛУ на входы схемы поступают операнды. Операнды на входы АЛУ могут поступать из внешней памяти. Туда же можно помещать и результат операции. Однако операции обращения к внешней памяти являются очень медленными, и такой порядок функционирования АЛУ характеризовался бы низким быстродействием.

Поэтому в составе МП обычно имеется несколько ячеек памяти, называемых ***рабочими регистрами***. Один из этих регистров часто наделяется особыми функциями: он содержит один из операндов, участвующих в операции, и в него помещается результат, формируемый в АЛУ. Этот регистр называют ***аккумулятором*** или регистром А. Использование такого регистра упрощает адресацию в различных командах, позволяет не указывать в команде адрес одного операнда и адрес регистра результата.

Существуют структуры микропроцессоров, в которых отсутствует аккумулятор, операнды могут располагаться в произвольных внутренних регистрах, а результат выполнения операции в АЛУ записывается в любой указанный регистр. Такие структуры называют безаккумуляторными.

Остальные регистры обычно имеют одинаковое назначение, однообразно используются в операциях и называются ***регистрами общего назначения (РОН)***. Один из операндов, участвующий в операции, часто хранится в одном из регистров общего назначения. Регистров сравнительно мало, поэтому для их вызова обычно требуется короткий адрес.

Для передачи адресной информации во внешнюю память в структуре МП необходим специальный ***регистр адреса*** **(РА)**, а для передачи и приема данных используется ***регистр данных*** **(РД)**.

Для связи всех перечисленных компонент в МП используется ***внутренняя шина***, к которой подключаются все элементы микропроцессора. Для выбора любого регистра РОН применяется ***мультиплексор***, который определяет регистр по короткому адресу, передаваемому по внутренней шине (рисунок 1.3). Приведенная структура МП является неполной.

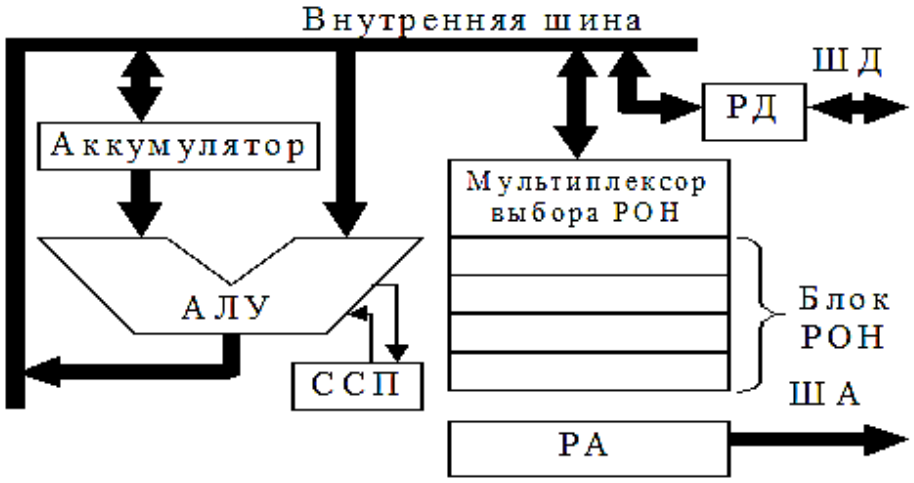

Рисунок 1.3 - Структура МП с АЛУ, регистрами и внутренней шиной

Микропроцессор сам не знает, что делать с данными. Для указания требуемых действий используются команды, поступающие из внешней памяти через шину данных. Для приема и хранения текущей команды используется ***регистр команд***. Для идентификации команды используется ***дешифратор команды***, в функции которого входит определение типа текущей команды и инициирование (запуск) такой последовательности действий, которая выполнит требуемую функцию.

Сам процесс управления всей совокупностью элементов МП выполняет ***устройство управления*** **(УУ)**, в функцию которого также входит прием внешних сигналов и формирование необходимых ответных, управляющих и синхронизирующих сигналов, необходимых для работы внешних устройств. Такую совокупность входных и выходных внешних сигналов часто называют ***шиной управления*** **(ШУ).**

Кто же формирует адрес текущей команды и загружает его в РА? В общем случае для этого используется специальное устройство управления адресом. Часто элементы программы (команды и данные) располагаются в последовательных ячейках памяти, начиная с некоторого, заранее определенного адреса, например, нулевого. В этом случае указание адреса следующей команды или элемента данных может быть ре-

ализовано по принципу счетчика, путем приращения значения адреса после каждого обращения к памяти.

Элемент, формирующий адрес по указанному принципу, называется ***счетчиком команд*** **(СК)**. Содержание СК загружается в РА перед чтением очередной команды или данных.

Однако такой метод управления адресом имеет недостаточную гибкость. В процессе выполнения программы может возникнуть необходимость изменения последовательного характера задания адреса. Например, если при проверке результата выполнения последней команды требуется одна из двух различных последовательностей команд, или требуется произвести обращение к непоследовательно расположенной ячейке данных. Для этого случая обычно имеется возможность занесения новых данных как непосредственно в счетчик команд для изменения хода программы, так и в регистр адреса для обращения к произвольной ячейке памяти.

Для расширения функциональных возможностей МП прибегают к организации ***стека*** – набора внутренних или внешних ячеек памяти, используемых для временного хранения обрабатываемых данных. При этом загрузка и выгрузка стека происходит в последовательные ячейки. Для адресации текущей ячейки (***верхушка стека***) используется регистр ***указатель стека***. При каждом обращении к стеку, указатель стека уменьшает или увеличивает свое содержимое, позволяя производить последовательную загрузку и считывать в обратном порядке ранее загруженные данные. При обращении к стеку в РА загружается адрес из регистра указателя стека.

Таким образом, мы, в основном, определили состав элементов, входящих в структуру микропроцессора, и указали необходимые связи между компонентами. Приведем полную структурную схему микропроцессора (рисунок 1.4).

Дополнительно в функции УУ МП может входить обслуживание прерываний, режим прямого доступа к памяти, режим синхронизации с работой медленных внешних устройств т.д.

Прерыванием называется некое внешнее или внутреннее событие, требующее немедленной реакции, например, готовность устройства к вводу/выводу, которое специальными аппаратными средствами прерывает текущий программный процесс и вынуждает микропроцессор загрузить специальную подпрограмму обслуживания прерывания. После ее завершения МП возвращается к выполнению прерванной программы.

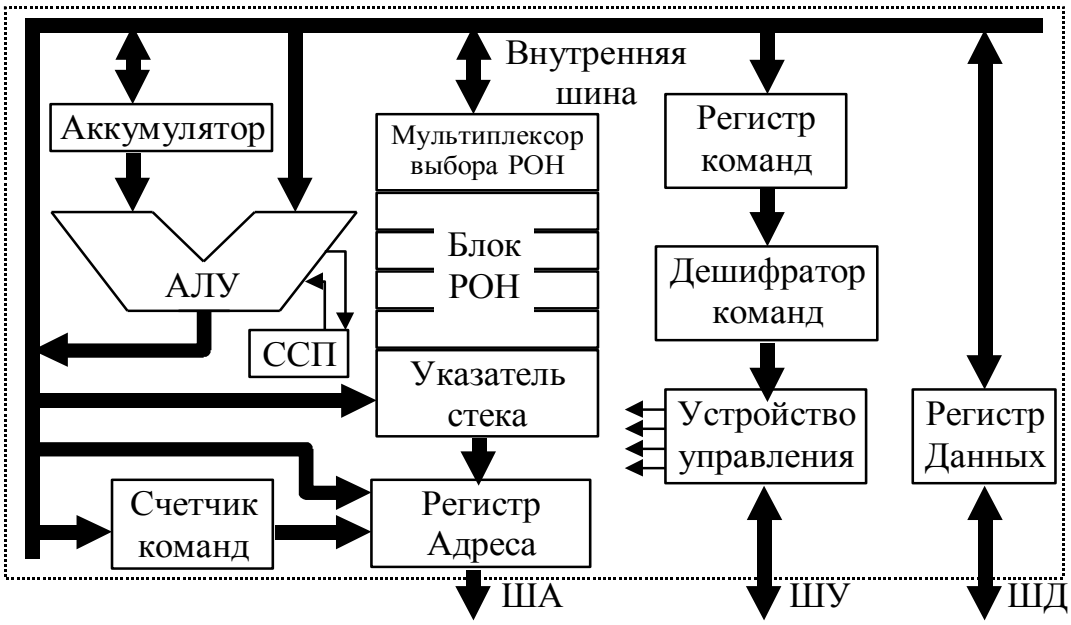

Рисунок 1.4 - Полная структурная схема микропроцессора

Режим ***прямого доступа к памяти*** **(ПДП)** предполагает, что на некоторое время МП полностью отключается от всех шин. В это время аппаратное устройство, называемое контроллер ПДП, формирует адреса памяти и сигналы записи/чтения, а некоторое устройство ввода/вывода напрямую записывает или считывает ячейки памяти. Этот режим позволяет наиболее быстро загрузить или считать большой блок памяти.

1.3 Контрольные вопросы

1. Перечислите основные функции, которые выполняет обычная ЭВМ.
2. Чем отличаются понятия микро-ЭВМ и микропроцессор?
3. Для каких целей в состав МП введены регистры общего назначения?
4. Какие преимущества можно отметить у структуры МП, в которой используется аккумулятор?
5. Для чего в структуре МП используется внутренняя шина?
6. Перечислите основные функции каждого из компонентов, входящих в состав микропроцессора.

2 СИСТЕМЫ СЧИСЛЕНИЯ В ЭВМ

2.1 Позиционный принцип представления чисел

Основанием системы счисления **p** будем называть число различных цифр-символов, используемых для представления любого числа в данной системе счисления. Любое число представляется в виде последовательности цифр, разделенных запятой на две группы. Группа цифр, расположенных левее запятой, образует целую часть числа. Правее запятой расположена дробная часть числа.

Каждая цифра числа занимает в нем определенную позицию, называемую разрядом. Разрядам приписываются различные весовые коэффициенты. Коэффициенты соответствуют различным целым степеням, в которые возводится основание системы счисления. Цифры данного числа представляются в виде символов некоторого счетного алфавита, обозначающих целые числа, находящиеся в диапазоне $0 < a_j < (p-1)$.

Используя позиционный принцип можно представить любое число N в любой системе счисления с основанием **p** (формула (2.1)):

$$N = a_n p^n + a_{n-1} p^{n-1} + \cdots + a_0 p^0 + a_{-1} p^{-1} + a_{-2} p^{-2} + \cdots + a_{-m} p^{-m} \quad (2.1)$$

В оформлении программ для микропроцессоров широкое распространение получили несколько систем счисления. Рассмотрим наиболее употребительные системы счисления.

Десятичная система счисления. Десятичная система счисления, знакомая всем со школы, имеет основание **p**=10. Для представления любого числа достаточно десяти цифр: 0, 1, 2, 3,…,9, из которых и составляются разряды числа. Цифры целой части числа имеют вес 10^0, 10^1, 10^2, …, а дробная часть имеет коэффициенты вида 10^{-1}, 10^{-2}, 10^{-3}, и т.д.

Таким образом, число 9875,432 в десятичной системе счисления можно представить в следующем виде (формула (2.2)):

$$9875{,}432 = 9\cdot 10^3 + 8\cdot 10^2 + 7\cdot 10^1 + 5\cdot 10^0 + 4\cdot 10^{-1} + 3\cdot 10^{-2} + 2\cdot 10^{-3} \quad (2.2)$$

При работе только в одной системе счисления проблем для обозначения системы числа не возникает. Труднее обстоит дело при возможности одновременного использования нескольких систем. Одинаковое число в разных системах счисления будет иметь разные значения. Поэтому появилась потребность обозначать систему счисления числа.

Чтобы отличать десятичные числа, можно указывать индекс $_{10}$.Запись будет выглядеть как 24687_{10}. Использование индексов не всегда удобно. Было предложено для обозначения системы счисления применять буквы, указываемые после самого числа. В программировании принято, что числа, не помеченные никакой буквой, или имеющие в конце букву D (сокращение от DECIMAL) являются десятичными числами. Например, правильно десятичное число можно записать, как 1101D.

Двоичная система счисления. Основанием двоичной системы является **p**=2, а для представления разрядов чисел используются два символа: 0 и 1.

Укажем пример числа в двоичной системе: $101101{,}01_2$. Позиционный принцип дает следующее значение данного числа:

$$N = 1\cdot 2^5 + 0\cdot 2^4 + 1\cdot 2^3 + 1\cdot 2^2 + 0\cdot 2^1 + 1\cdot 2^0 + 0\cdot 2^{-1} + 1\cdot 2^{-2} = 45{,}25_{10} \qquad (2.3)$$

Вес дробных разрядов, таким образом, равен 0,5, 0,25, 0,125 и далее как дробь $(1/2)^n$. Коэффициенты для разрядов целой части числа равны 1, 2 , 4, 8, 16, 32 и т.д.

Для обозначения двоичной системы числа можно применять индекс 2, как показано выше, а можно использовать букву B (BINARY), указываемую после числа. Например, двоичное число можно записать как 011010110B.

Восьмеричная система счисления. Основание системы счисления **p**=8. В каждом разряде может использоваться один из восьми символов: 0, 1, 2, 3, 4, 5, 6 или 7. Пример числа в данной системе счисления имеет вид: $72645{,}27_8$. Значение числа равно:

$$N = 7\cdot 8^4 + 2\cdot 8^3 + 6\cdot 8^2 + 4\cdot 8^1 + 5\cdot 8^0 + 2\cdot 8^{-1} + 7\cdot 8^{-2} = 30117{,}359_{10} \qquad (2.4)$$

Вес дробных разрядов в данной системе равен 1/8, 1/64, 1/256 и далее как дробь $(1/8)^n$. Коэффициенты для веса разрядов в целой части числа равны 1, 8, 64, 256, 4096 и т.д.

Для обозначения чисел используется индекс $_8$ или указывается буква O (сокращение от OCTAL). В связи с возможностью совпадения начертания буквы O с цифрой 0, восьмеричные числа иногда обозначают буквой Q. Пример правильного обозначения чисел 163527O, 267453Q.

Шестнадцатеричная система счисления. Основание системы **p**=16. Для обозначения цифр каждого разряда требуется 16 различных символов. Цифры от 0 до 9 соответствуют цифрам из десятичной системы счисления. Для обозначения недостающих шести символов, соответствующих значениям 10, 11, 12, 13, 14 и 15 используются буквы, соответственно, A, B, C, D, E и F. Рассмотрим пример числа $2F34,C8_{16}$ (формула (2.5)):

$$N = 2\cdot 16^3 + 15\cdot 16^2 + 3\cdot 16^1 + 4\cdot 16^0 + 12\cdot 16^{-1} + 8\cdot 16^{-2} = 12084{,}78125_{10} \qquad (2.5)$$

В формуле (2.5) числом 15 заменен символ F, а числом 12 – символ C.

Индекс $_{16}$ позволяет отличать числа шестнадцатеричной системы от любой другой. Допустимо обозначение буквой H. Буквенное обозначение H взято от английского слова HEXADECIMAL. Пример числа в данной системе имеет вид 4FA6H. Кроме этого, если шестнадцатеричное число начинается с буквы, то принято добавлять в начале цифру 0. Другой пример запишем как: 0B3A4H. При этом соблюдается соглашение, которое гласит: числа должны начинаться с цифры, а не буквы. Этим устраняется схожесть в обозначении шестнадцатеричных чисел и обозначении служебных имен и меток.

Двоично-десятичные числа. Микропроцессор хранит и обрабатывает числа в двоичной форме. Для обработки и хранения десятичных чисел, каждая цифра десятичного числа представляется в двоичной форме. Для записи любой десятичной цифры достаточно четырех двоичных битов. Поэтому в одном байте можно упаковать две двоично-десятичных цифры.

Например, число $1357,26_{10}$ можно представить в следующем виде:

$$1357{,}26 = \underbrace{0001}_{1}\ \underbrace{0011}_{3}\ \underbrace{0101}_{5}\ \underbrace{0111}_{7},\underbrace{0010}_{2}\ \underbrace{0110}_{6}$$

Двоично-десятичное число по значению отличается от аналогичного двоичного числа. Например, попробуем рассмотреть предыдущее число как двоичное. Переведем его в десятичное число. При этом получим $4951,1484_{10}$. Результирующее число кардинально отличается от исходного значения.

Такой способ кодирования десятичных цифр называется *кодом 8421*. Название составлено из весовых коэффициентов разрядов двоичного числа. Этот код является естественным представлением десятичных цифр в двоичной системе. Распространены и другие коды (таблица 2.1).

Таблица 2.1 – Коды двоично-десятичных систем

Десятичные цифры	Двоично-кодированные системы				
	8421	2421	С избытком 3	7421	Либау-Крейга
0	0 0 0 0	0 0 0 0	0 0 1 1	0 0 0 0	0 0 0 0 0
1	0 0 0 1	0 0 0 1	0 1 0 0	0 0 0 1	0 0 0 0 1
2	0 0 1 0	0 0 1 0	0 1 0 1	0 0 1 0	0 0 0 1 1
3	0 0 1 1	0 0 1 1	0 1 1 0	0 0 1 1	0 0 1 1 1
4	0 1 0 0	0 1 0 0	0 1 1 1	0 1 0 0	0 1 1 1 1
5	0 1 0 1	1 0 1 1	1 0 0 0	0 1 0 1	1 1 1 1 1
6	0 1 1 0	1 1 0 0	1 0 0 1	0 1 1 0	1 1 1 1 0
7	0 1 1 1	1 1 0 1	1 0 1 0	1 0 0 0	1 1 1 0 0
8	1 0 0 0	1 1 1 0	1 0 1 1	1 0 0 1	1 1 0 0 0
9	1 0 0 1	1 1 1 1	1 1 0 0	1 0 1 0	1 0 0 0 0

Код *7421* содержит в каждой кодовой комбинации не более двух единиц.

В коде *2 из 5* каждая кодовая комбинация содержит точно две единицы.

В коде *2421* и коде *с избытком 3* кодовая комбинация, соответствующая любой десятичной цифре, представляет инверсию комбинации, соответствующей цифре, дополняющей данную до девяти. Например, в коде 2421 паре взаимно дополняющих цифр 3 и 6 соответствуют коды 0011 и 1100.

В *коде Либау-Крейга* каждая последующая комбинация отличается от предыдущей только в одном разряде и изменяется по циклу.

Код Грея. Двоичные коды очень удобны для выполнения вычислений, но они создают серьезные осложнения в практических приложениях, где используются переходы между двумя соседними значениями. Например, система управления линейным перемещением стержня или углом поворота вала. Если датчик положения будет кодироваться в двоичном коде, то некоторые соседние положения могут отличаться друг от друга во многих битах.

Например, позиции 3 и 4 будут иметь код 0011 и 0100. Из-за несинхронности изменения разрядов может возникать кратковременная неопределенность в значениях кода. Код на короткое время может при-

нять значения 0010, 0001, 0110, 0101 или 0111. Все эти значения сильно отличаются как от старого, так и нового значения кода. В результате, система управления зафиксирует эту переходную погрешность и начнет вырабатывать управляющий сигнал для корректировки позиции.

Чтобы устранить эту переходную погрешность коды в соседних позициях должны отличаться только в одном разряде. Поскольку при смене кода остальные биты не изменяются, то неопределенность не возникает. Фактически мы имеем или старое или новое значение кода. Код Грея относится к рефлексным двоичным кодам и имеет такую же разрядность и такой же старший бит, как и двоичный код (таблица 2.2).

Таблица 2.2 – Код Грея и двоичные коды

Десятичный код	Двоичный код	Шестнадца-теричный код	Код Грея	Десятичный код	Двоичный код	Шестнадца-теричный код	Код Грея
0	0 0 0 0	0	0 0 0 0	8	1 0 0 0	8	1 1 0 0
1	0 0 0 1	1	0 0 0 1	9	1 0 0 1	9	1 1 0 1
2	0 0 1 0	2	0 0 1 1	10	1 0 1 0	A	1 1 1 1
3	0 0 1 1	3	0 0 1 0	11	1 0 1 1	B	1 1 1 0
4	0 1 0 0	4	0 1 1 0	12	1 1 0 0	C	1 0 1 0
5	0 1 0 1	5	0 1 1 1	13	1 1 0 1	D	1 0 1 1
6	0 1 1 0	6	0 1 0 1	14	1 1 1 0	E	1 0 0 1
7	0 1 1 1	7	0 1 0 0	15	1 1 1 1	F	1 0 0 0

Код Грея отлично удовлетворяет требованиям к датчикам положения и активно применяется во многих практических случаях.

2.2 Перевод чисел из разных систем счисления

Перевод чисел из двоичной системы и обратно. Перевод чисел из двоичной системы счисления в шестнадцатеричную систему и обратно выполняется очень просто. Основание шестнадцатеричной системы выражается целой степенью двойки ($16=2^4$), поэтому для представления двоичного числа в шестнадцатеричной форме необходимо каждую четверку битов (тетраду) заменить эквивалентной цифрой. Например, $0110\ 1011\ 0011\ 1110_2$ в шестнадцатеричной форме имеет

вид: $6B3E_{16}$, т.е. $0110_2 \Rightarrow 6_{16}$, $1011_2 \Rightarrow B_{16}$, $0011_2 \Rightarrow 3_{16}$, $1110_2 \Rightarrow E_{16}$. Обратное преобразование в двоичную форму выполняется заменой каждой цифры двоичным эквивалентом.

Аналогично, перевод между двоичной и восьмеричной системами очень прост. Двоичное число надо разбить на тройки битов ($8=2^3$) и каждую тройку заменить соответствующей восьмеричной цифрой. Например, число, рассмотренное ранее, разобьем на тройки битов и запишем в восьмеричной форме. Получаем $0110\ 1011\ 0011\ 1110_2=110\ 101\ 100\ 111\ 110_2=65476_8$. При разбиении двоичного числа на группы по 3 бита, левая крайняя неполная группа дополняется нулями.

Перевод числа из восьмеричной системы в двоичную выполняется путем преобразования каждой восьмеричной цифры числа в двоичную форму.

Перевод из десятичной системы и обратно. Большую сложность представляет перевод между десятичной и двоичной системами счисления. Метод перевода зависит от системы счисления, в которой производятся вычисления. Для человека удобнее вычислять в десятичной форме и переводить числа из десятичной в двоичную систему. Микропроцессор работает в двоичной системе, результат из которой необходимо показать в десятичной форме. Поэтому нужно уметь переводить числа из десятичной в двоичную систему и обратно. Будем рассматривать случаи отдельно.

Пользуясь тем, что легко преобразовать числа между двоичной и шестнадцатеричной системами, для простоты изложения будем рассматривать перевод между десятичной и шестнадцатеричной формами представления.

Перевод в десятичную систему из шестнадцатеричной формы можно выполнить, пользуясь позиционным принципом. Например, выберем число $3F5C,5A6_{16}$. Позиционное представление числа (формула (2.1)) дает:

$$3F5C,5A6_{16} = (3\cdot16^3 + 15\cdot16^2 + 5\cdot16^1 + 12\cdot16^0 + 5\cdot16^{-1} + 10\cdot16^{-2} + 6\cdot16^{-3})_{10} \qquad (2.6)$$

Здесь шестнадцатеричная цифра F заменена на десятичное число 15, а цифра C – на число 12. Для сокращения числа операций умножения и деления удобно воспользоваться *схемой Горнера*. Для ее реализации в правой части формулы (2.6) выполним последовательную группировку членов и вынесем общие сомножители за скобки (формула (2.7)):

$$3F5C,5A6_{16} = ((((3\cdot16+15)\cdot16+5)\cdot16+12)+16^{-1}\cdot(5+16^{-1}\cdot(10+16^{-1}\cdot6)))_{10} \quad (2.7)$$

Это и есть реализация схемы Горнера для вычисления значения многочлена. В данной формуле всего 3 операции умножения и 3 операции деления против 6 операций умножения и 6 операций деления в предыдущей формуле.

При выполнении вычисления над целой частью нужно старший разряд числа умножить на основание системы счисления. К результату прибавить значение следующей, более младшей шифры. Полученную сумму нужно умножить на основание, прибавить очередную цифру и повторять до последней цифры исходного числа. При вычислении дробной части числа необходимо брать самую младшую цифру, делить ее на основание системы (умножать на 16^{-1}), прибавлять предыдущую цифру и повторять деление и суммирование до исчерпания всех дробных цифр.

Целая часть числа вычисляется точно, дробная часть получается приближенно. Для нашего числа получаем $3F5C,5A6_{16} \cong 16220,3525390625_{10}$. При вычислении дробных разрядов точность зависит от числа двоичных разрядов. В нашем примере точность ограничена десятью битами.

Рассмотрим обратный перевод чисел из десятичной формы в шестнадцатеричную систему счисления. Воспользуемся предыдущим примером.

Пусть нам задано целое число 16220_{10}. Как получить шестнадцатеричное число $3F5C_{16}$? Из равенства $3F5C_{16} = (((3\cdot16+15)\cdot16+5)\cdot16+12)_{10}$ можно вывести следующее правило получения цифр.

Если разделить правую часть формулы на 16, то в частном мы получим ((3·16+15)·16+5) и в остатке получим 12 (=C). Снова разделим полученное частное на 16. Получим (3·16+15) и остаток будет равен 5. Еще раз разделим новое частное на 16. Получим 3 и остаток 15 (=F). Поскольку, частное меньше 16, прекращаем процесс деления. Видим, что мы последовательно в обратном порядке получили все цифры шестнадцатеричного числа $3F5C_{16}$:

$$
\begin{array}{lllll}
16220 & \vert\underline{16} & & & \\
\underline{-16208} & 1013 & \vert\underline{16} & & \\
\{12 = C\} & \underline{-1008} & 63 & \vert\underline{16} & \\
 & \{5\} & \underline{-48} & \{3\} & \\
 & & \{15 = F\} & &
\end{array}
$$

Дробная часть числа $0{,}3525390625_{10}$ после преобразования должна дать $0{,}5A6_{16}$.

Рассматривая формулу (2.7) относительно дробной части, делаем вывод, что дробную часть числа нужно последовательно умножать на 16. При этом будут получаться числа, целые части которых дадут искомые цифры дробной части числа в шестнадцатеричной системе. Эти целые части нужно будет вычитать из получающихся чисел, снова получать дробные части и продолжать умножение:

$$
\begin{array}{lll}
 & & 0{,}3525390625 \\
 & & \times\ \underline{\qquad\qquad 16} \\
5 & \leftarrow & 5{,}640625000 \\
 & & 0{,}640625000 \\
 & & \times\ \underline{\qquad\qquad 16} \\
A & \leftarrow & 10{,}2500000000 \\
 & & 0{,}2500000000 \\
 & & \times\ \underline{\qquad\qquad 16} \\
4 & \leftarrow & 4{,}0000000000 \\
 & & 0{,}0000000000 \\
 & & \times\ \underline{\qquad\qquad 16} \\
0 & \leftarrow & 0{,}0000000000
\end{array}
$$

Получили $0{,}5A4_{16} \cong 0{,}5A6_{16}$. Несоответствие результатов объясняется ограниченной точностью представления исходного числа. Еще раз убеждаемся, что дробные числа преобразуются неточно.

Думаете, так можно преобразовывать только в шестнадцатеричную систему счисления? Ничего подобного. Оказывается, таким образом, можно преобразовывать целые и дробные десятичные числа в любую мыслимую систему счисления, даже в систему с основанием **p**=36 .

Сформулируем основные правила преобразования из любой системы счисления в десятичную и обратно.

Правило 1 *перевода в десятичную систему счисления из произвольной*. Целую часть числа в произвольной системе счисления расписываем по схеме Горнера (формула (2.7)). Старшую цифру числа умножаем на основание системы счисления **p**. К произведению прибавляем следующую, более младшую, цифру. Результат умножаем на **p**. Умножение и суммирование повторяем до исчерпания всех цифр целой части числа.

Дробная часть числа преобразуется, начиная с самой младшей цифры. Делим эту цифру на **p** (умножаем на $\mathbf{p}^{-1}$). К результату прибавляем следующую, более старшую, цифру дробной части числа. Делим результат на **p**. Повторяем суммирование и деление до исчерпания всех цифр дробной части. Результат округляем до заранее заданной точности.

Правило 2 *перевода в произвольную систему счисления из десятичной*.

Делим целую часть десятичного числа на основание выбранной системы счисления **p**. Запоминаем остаток, как значение самой младшей цифры преобразованного числа. Делим полученное частное на **p.** Снова фиксируем остаток, как значение следующей, более старшей цифры. Повторяем процесс деления до получения частного, меньшего, чем **p**. Последовательно получаемые в процессе деления остатки являются цифрами искомого числа, от младшей к старшей цифре.

Дробную часть числа преобразуем отдельно. Умножаем дробную часть числа на основание **p**. Запоминаем целую часть результата, как первую цифру вычисляемого дробного числа. Дробную часть полученного после умножения числа снова умножаем на **p**. Снова фиксируем целую часть результата, как очередную цифру дробной части числа. Из-за неточного характера преобразования процесс необходимо прекратить либо при получении нулевого результата, либо при получении требуемого числа цифр преобразованного числа. Т.е. что называется “уже хватит”.

Указанные правила позволяют легко преобразовывать числа между десятичной и двоичной системами, между десятичной и восьмеричной системами и т.д.

Таким образом, мы научились преобразовывать числа между произвольными системами счисления, наиболее распространенными в вычислительной технике и программировании микропроцессоров.

2.3 Арифметические операции с числами

Основными арифметическими операциями являются операции суммирования, вычитания, умножения и деления чисел. Со школьной скамьи мы знаем, что операции умножения и деления можно свести к операциям суммирования и вычитания. В свою очередь, операцию вычитания можно заменить операцией суммирования с отрицательным числом.

Поэтому будем считать операцию суммирования основной арифметической операцией, при условии, что числа могут быть как положительными, так и отрицательными.

При записи кода числа знак плюс принято обозначать цифрой 0, а знак минус – цифрой 1. Если в числе необходимо отобразить знак, то числа будем называть знаковыми, иначе беззнаковыми. Для двоичных знаковых чисел старший разряд числа называется знаковым. Для простоты изложения мы будем знаковый бит заключать в квадратные скобки и рассматривать все числа, как целые.

Сложение положительных чисел. Рассмотрим операцию сложения на примере двоичных положительных чисел. При этом основным правилом сложения двоичных чисел является правило побитного сложения. Всего возможно 4 сочетания битов. Рассмотрим результаты сложения пары битов для всех возможных сочетаний.

Первый бит	1	0	1	0
Второй бит	1	1	0	0
Сумма	0	1	1	0
перенос	1			

Здесь приведены все возможные сочетания суммы двух битов. При появлении переноса, он учитывается при суммировании других, более старших битов. Рассмотрим пример сложения многоразрядных чисел:

Переносы	111
Первое слагаемое	[0]01011
Второе слагаемое	[0]01110
Сумма	[0]11001

При суммировании чисел возникают переносы, которые суммируются с последующими разрядами.

Сложение в дополнительном коде. При сложении положительных чисел трудностей обычно не возникает. Труднее складывать числа, имеющие знаки. Для чисел со знаком операция сложения выполняется только при равенстве знаков. Для чисел с разными знаками вместо сложения приходится применять операцию вычитания. Фактически, вид операции зависит от сочетания знаковых разрядов.

Такие операции упрощает использование *дополнительного кода.* Для пояснения принципа использования дополнительного кода, рассмотрим простой пример вычитания положительных десятичных чисел. Пусть $N_1 = 756$, а $N_2 = 279$. Запишем знак чисел N_1=[0]756, а N_2=[0]279. Необходимо вычислить разность N_1- N_2. Заменим операцию вычитания операцией сложения с отрицательным числом, т.е. изменим знак второго числа. Тогда N_1=[0]756, а N_2=[1]279 .

Выполнение такого сложения по школьным правилам потребует последовательности действий с заемами из старших разрядов.

```
  [0]756
+ [1]279
---------
  [0]477
```

В цифровом устройстве не обязательно предусматривать такую последовательность действий. Преобразуем отрицательное число в *дополнительный код* следующим образом: во всех разрядах, кроме знакового, заменим каждую цифру дополнением до девяти к значению этой цифры (при этом каждую цифру нужно как бы вычесть из цифры 9). Цифра 2, вычтенная из 9, превратится в 7, цифра 7 – аналогично в 2. Дополнение до 9 цифры 9 дает 0. Получим $N_{2доп-1}$=[1]720. В заключении к числу приплюсуем единицу в младший разряд. Мы получили дополнительный код отрицательного числа $N_{2доп}$=[1]721. Почему дополнение до 9? Мы фактически вычли наше отрицательное число из максимального числа, которое можно получить при заданном числе разрядов. А это максимальное число ровно на единицу больше числа, в котором все разряды равны 9. Цифра 9 является наибольшей цифрой в данной системе счисления. Отсюда и понятен способ получения дополнительного кода с вычитанием из 9 и прибавлением 1 в заключении.

Теперь будем выполнять операцию суммирования, не обращая внимания на знаки чисел и результата.

```
  [0]756
+[1]721
---------
  [0]477
```

перенос 1 1

В примере суммируются и двоичные цифры знаковых разрядов. При этом отбрасывается перенос, возникающий из знаковых разрядов. С удовлетворением замечаем, что получен правильный результат операции.

В двоичной системе в дополнительном коде потребуется дополнение каждого бита до единицы (помните, максимальная цифра в данной системе счисления). Это можно выполнить простым инвертированием (замена 0 на 1 , а 1 на 0) всех битов, кроме знакового. Не забывайте в заключении прибавить единицу к младшему разряду числа. Например, N=[1]0010110 преобразуется в $N_{доп}$=[1]1101010. Обратное преобразование выполняется по тому же правилу.

Рассмотрим примеры сложения двоичных чисел в дополнительном коде. Пусть первое слагаемое $N_1 = [0]0110100_2 = 52_{10}$, имеет положительный знак, а второе слагаемое $N_2 = [1]0101101_2 = -45_{10}$ задано со знаком минус. Преобразуем второе число в дополнительный код. Для этого выполним инверсию числа, за исключением знакового бита, и прибавим к числу единицу. Получим $N_{2доп.} = [1]1010011_2$. Выполняем саму операцию сложения:

```
    [0]0110100
    [1]1010011
    ----------
    [0]0000111
переносы  1 111
```

Не забываем, что возникающий из знаковых разрядов перенос нужно отбросить. Получили число $[0]0000111_2 = 7_{10}$. Видим, что это правильный результат сложения исходных чисел с учетом их знаков.

Изменим на обратные знаки слагаемых из предыдущего примера. Первое слагаемое $N_1 = [1]0110100_2 = -52_{10}$ теперь отрицательно, а второе слагаемое $N_2 = [0]0101101_2 = +45_{10}$ имеет положительный знак. Ожидаемый результат сложения чисел равен $[1]0000111_2 = -7_{10}$. Преобразуем отрицательное число N_1 в дополнительный код $N_{1доп} = [1]1001100_2$. Выполняем сложение чисел.

```
    [1]1001100
    [0]0101101
    ----------
    [1]1111001
переносы  11
```

Ответ получился отрицательным, но что-то не похож на ожидаемый результат! Да ведь он, наверно, в дополнительном коде! Преобразуем число из дополнительного кода по известному алгоритму: инвер-

сия всех битов, за исключением знакового, и суммирование с единицей. Получим результат СУММА=[1]0000111. Да, это то, что ожидалось!

Подведем некоторый итог. При использовании дополнительного кода нужно заранее преобразовать числа. Дополнительный код положительного числа равен самому числу, а дополнительный код отрицательного числа получается инвертированием всех битов числа, за исключением знакового разряда, и увеличением на единицу полученного числа. При выполнении операции сложения в дополнительном коде нужно отбрасывать единицу переноса из знакового бита. Причем положительный результат суммирования равен самому числу, а отрицательный результат получается в дополнительном коде.

Таким образом, применение дополнительного кода позволяет отказаться от операции вычитания с заменой ее на операцию суммирования в дополнительном коде. Этот прием широко используется при знаковых вычислениях в микропроцессорах.

Однако, дополнительный код не позволяет обнаружить ***переполнение разрядной сетки*** – явление, при котором результат операции содержит большее число разрядов, чем число разрядов в устройстве, предназначенном для его хранения. При выполнении операций в дополнительном коде это явление приводит к тому, что у результата получается неправильный знак. Например, сумма положительных чисел при переполнении имеет отрицательный знак результата, или сумма отрицательных чисел дает положительный результат. Например:

```
          [0]1100001
        + [0]0111101
        -------------
          [1]0011110
переносы      11
```

Результат сложения двух положительных чисел имеет знак минус, а ошибка объясняется переполнением разрядной сетки. Самое неприятное в этой ситуации то, что сам ошибочный результат ничем особенным не выделяется, ошибка себя не обнаруживает и неправильный результат может быть использован в дальнейших вычислениях. А это приведет к другим ошибкам. Выходом является использование *модифицированного дополнительного кода*.

Сложение в модифицированном коде. Модифицированный код отличается наличием двух одинаковых знаковых битов. В модифицированном дополнительном коде у положительных чисел знаковые разряды равны нулю, а у отрицательных – единице. При выполнении опера-

ций правильным результатом будет число с одинаковыми знаковыми битами. Наличие комбинации 01 или 10 в знаковых разрядов однозначно показывает переполнение разрядной сетки. А это убережет от использования ошибочного результата в дальнейших вычислениях. Рассмотрим сложение в модифицированном дополнительном коде. Сначала проверим предыдущий пример:

```
          [00]1100001
          [00]0111101
          [01]0011110
переносы      11    1
```

Мы видим, что результат содержит разные знаковые биты, поэтому он ошибочен. Ошибка объясняется тем, что для размещения значащей части результата необходимо 8, а не 7 разрядов, поскольку $N_1 = 97_{10}$, а $N_2 = 61_{10}$. Результат $N_1 + N_2 = 158_{10}$ больше предельного значения 127.

Рассмотрим сложение отрицательных чисел в модифицированном дополнительном коде. Числа уже преобразованы в дополнительный код и равны N_1 = [11]0110100 и N_2 = [11]0101101.

```
          [11]0110100
          [11]0101101
          [10]1100001
переносы  11   1111
```

В результирующем числе знаковые биты 10 сигнализируют о переполнении разрядной сетки, т.е. результат не помещается в семи двоичных разрядах. Действительно, если перевести числа в десятичную систему счисления, мы получим $N_1 = -76_{10}$, а $N_2 = -83_{10}$. Результат $N_1 + N_2 = -159_{10}$. Это действительно превышает значение -128_{10}, которое является наибольшим по модулю отрицательным числом, занимающим семь двоичных разрядов.

Рассмотрим другие примеры.

```
          [00]0110100
          [00]0101101
          [00]1100001
переносы      1111
```

Суммирование положительных чисел дало правильный результат. Действительно, $N_1 = 52_{10}$, а $N_2 = 45_{10}$. Результат $N_1 + N_2 = 97_{10}$. Рассмотрим еще один пример.

```
          [11]1100001
          [11]0111101
```

[11]0011110
переносы 11 11 1

Результат правильный и в десятичных числах записывается как $N_1 = -31_{10}$, а $N_2 = -67_{10}$. Результат $N_1 + N_2 = -98_{10}$. Для следующего примера возьмем числа с разными знаками $N_1 = -52_{10}$, а $N_2 = +45_{10}$. Результат $N_1 + N_2 = -7_{10}$.

[11]1001100
[00]0101101
[11]1111001
переносы 11

Результат получился правильным и представлен в модифицированном дополнительном коде. Можете поверить на слово, что и сложение большого по модулю положительного числа с маленьким (по модулю) отрицательным числом даст правильный положительный результат.

Справедливости ради следует сказать, что кроме указанных кодов существуют другие коды, в которых определяются операции в цифровых устройствах и микропроцессорах. Например, это обратный код, код Джонсона и другие. Все они имеют свои преимущества и недостатки и могут применяться в различных случаях. Однако их рассмотрение выходит за рамки данной дисциплины.

Умножение двоичных чисел. При выполнении операции умножения нужно определить знак результата и найти абсолютное значение произведения.

Знаковый разряд произведения можно определить суммированием битов знаковых разрядов по модулю 2. Эта операция обозначается как $\oplus$. Действительно, при равенстве знаков сомножителей (0 и 0 или 1 и 1) произведение будет иметь положительный знак (обозначаемый как 0), а сумма по модулю 2 дает нуль, т.е. $0 \oplus 0 = 0$ и $1 \oplus 1 = 0$. При несовпадении знаков произведение будет иметь знак минус (т.е. 1), а $0 \oplus 1 = 1$ и $1 \oplus 0 = 1$.

Определение абсолютного значения произведения выполняется без учета знаков сомножителей. Наиболее просто перемножение выполняется методом сдвига множимого влево и суммированием частичных произведений. Приведем пример умножения:

```
        1 0 0 1 1
*       1 0 1 1
---------------
  1 0 0 1 1
```

+			1	0	0	1	1	
+		0	0	0	0	0		
+	1	0	0	1	1			
Итог	**1**	**1**	**0**	**1**	**0**	**0**	**0**	**1**
Переносы			1	1	1	1	1	

Как видно из примера, в процессе выполнения операции умножения суммируются сдвинутые на заданное число разрядов частичные произведения (произведения множимого на цифры разрядов множителя). Такие частичные произведения равны либо сдвинутому влево множимому, либо нулю. Это определяется соответствующим битом множителя. Например, второй бит множителя равен 1, значит, сдвинутое влево на 1 разряд множимое суммируется с промежуточной суммой.

В цифровых устройствах процессу суммирования частичных произведений придают последовательный характер, что и объясняется появлением термина «промежуточная сумма».

Нетрудно догадаться о причине появления среди частичных произведений нулевых членов. Они являются результатом умножения множимого на нулевой бит множителя. В общем случае, процесс суммирования можно начинать с частичного произведения, получаемого умножением множимого на старший, либо младший биты множителя. Этим определяется только направление сдвига частичного произведения относительно накапливаемой суммы (вправо или влево).

При умножении целых чисел результат, в общем виде, требует наличия числа разрядов, равных сумме числа разрядов сомножителей. Так, перемножение 5-ти битового числа на 4-х битовое дает 9-ти битовый результат.

Умножение дробных чисел (меньших единицы) выполняется по аналогичному алгоритму: суммирование частичных произведений с их сдвигом относительно промежуточной суммы. При умножении дробных чисел количество разрядов определяется пользователем. В общем случае, результат часто округляют до определенного числа разрядов.

Деление двоичных чисел. При делении двоичных чисел можно использовать различные алгоритмы. Один из простых алгоритмов предполагает вычитание делителя из делимого и подсчитывание количества вычитаний до получения отрицательного результата. Существенный недостаток данного алгоритма: зависимость времени выполнения операций от соотношения делимого и делителя.

Рассмотрим алгоритм, который называется делением целых чисел с восстановлением остатка. Правильные результаты получаются, если число разрядов делимого больше числа разрядов делителя. Рассмотрим пример деления числа $N_1 = 10010111_2 = 151_{10}$ на $N_2 = 1011_2 = 11_{10}$. Обозначим бит переноса в фигурных скобках.

			1	0	0	1	0	1	1	1	
		-	1	0	1	1					Вычитаем делитель
0	←	{1}	1	1	1	0	0	1	1	1	есть заем, восстанавливаем остаток
			1	0	0	1	0	1	1	1	остаток
		-		1	0	1	1				Сдвиг делителя вправо. Вычитаем
1	←	{0}	0	0	1	1	1	1	1	1	Нет заема, продолжаем
		-			1	0	1	1			Сдвиг делителя вправо. Вычитаем
1	←	{0}	0	0	0	1	0	0	1	1	Нет заема
		-				1	0	1	1		Сдвиг делителя вправо. Вычитаем
0	←	{1}	1	1	1	1	1	1	0	1	есть заем, восстанавливаем остаток
	←		0	0	0	1	0	0	1	1	остаток
		-					1	0	1	1	Сдвиг делителя вправо. Вычитаем
1	←	{0}	0	0	0	0	1	0	0	0	Нет заема. Завершаем деление

Процесс прекращаем при достижении исходного значения делителя. Биты частного равны 01101_2 , а остаток равен 1000_2. Это правильный результат деления $151_{10}/11_{10}=13_{10}$ и 8_{10} в остатке. Биты частного получились фиксацией инверсии бита переноса, начиная с первого, как самого старшего бита результата.

Опишем алгоритм деления.

1. Задаем начальное значение частного равным 0. Совмещаем делитель со старшими разрядами делимого.
2. Вычитаем из делимого делитель.
3. Сдвигаем частное на разряд влево и т.о. определяем младший бит. Младший бит частного оценивается по состоянию бита переноса. Если бит переноса равен 1, нужно в бите частного запомнить 0, а остаток восстановить (к остатку прибавить делитель). Если перенос равен 0, то в результате фиксируем 1, и ничего восстанавливать не нужно.
4. Сдвигаем делитель на разряд вправо, заполняя нулями свободные левые биты. Повторяем вычитание уже сдвинутого делителя из имеющегося остатка.
5. Оцениваем бит частного, аналогично шагу 3.
6. Если делитель еще не сравнялся со своим исходным значением (младший бит делителя не совместился с младшим битом делимого), идем на наг 4, иначе 7.

7. Останавливаем процесс деления.

Данный алгоритм требует значительно меньшее число итераций, чем простой подсчет количества вычитаний делителя из делимого. Недостатком алгоритма является необходимость восстанавливать остаток при получении отрицательного результата вычитания. Некоторый выигрыш в количестве операций дает алгоритм деления без восстановления остатка. Рассмотрим пример, используя числа из предыдущего примера. При вычислениях используется суммирование с прямым или дополнительным кодом делителя.

10010111 – делимое
+0101 – суммируем дополнительный код делителя
0 ← {0} 11100111 – переноса нет, нужен прямой код делителя
+01011 – сдвинем делитель вправо и прибавим его
1 ← {1} 00111111 – есть перенос, нужен дополнительный код
+110101 – сдвиг делителя вправо и дополнит. код
1 ← {1} 00010011 – есть перенос, нужен дополнительный код
+1110101 – сдвиг делителя вправо и дополнит. код
0 ← {0} 11111101 – переноса нет, нужен прямой код делителя
+ 00001011 – сдвинем делитель вправо и прибавим его
1 ← {1} 00001000 – есть перенос, пишем 1 и завершаем работу.

Опишем алгоритм деления без восстановления остатка.

1. Присваиваем частному значение 0. Совмещаем делитель со старшими разрядами делимого.
2. Формируем дополнительный код делителя
3. Суммируем делимое с дополнительным кодом делителя.
4. Сдвигаем частное влево. Младший бит частного фиксируется по биту переноса. Если бит переноса равен 1, пишем в младший бит частного 1, а если перенос равен 0, то пишем 0.
5. Сдвигаем делитель на разряд вправо, заполняя нулями свободные левые биты.
6. Если младший бит частного равен 0, то берем прямой код делителя, иначе образуем дополнительный код делителя.
7. Суммируем делимое с полученным на шаге 6 кодом делителя.
8. Оцениваем бит частного, аналогично шагу 4.
9. Если делитель еще не сравнялся со своим исходным значением, идем на шаг 5, иначе 10.
10. Завершаем процесс деления.

Как видно из описания, после каждого цикла суммирования делителя с делимым, делитель уменьшается в 2 раза за счет сдвига вправо. По значению бита переноса берем прямой или дополнительный код делителя и повторяем суммирование. Окончание процесса наступает при совмещении младших битов делителя и делимого. Время выполнения операции деления минимально.

Можно предложить и другие алгоритмы деления, например, составление таблицы всех возможных произведений. Количество строк будет равно количеству значений первого сомножителя, а количество столбцов – количеству значений второго сомножителя. Элементы таблицы равны произведениям сомножителей.

Деление нацело происходит по нахождению номера столбца, в котором находится ближайшее к делимому число в строке с номером, равным делителю. Такой метод деления обеспечивает самую высокую скорость деления. Существенный недостаток метода – очень большой размер таблицы.

2.4 Форматы представления чисел в МП

Микропроцессоры часто используются для вычисления сложных математических функций, проведения расчетов, решения систем уравнений, при этом диапазон изменения операндов достигает большой величины. Простым примером может являться калькулятор.

В виду того, что разрядность шины данных ограничена, при выполнении арифметических операций с большими числами возникают ошибки округления и усечения, приводящие к потере точности при вычислениях. Поэтому форма представления чисел в МП играет значительную роль.

Существуют следующие формы представления чисел:

- ♦ целые числа без знака;
- ♦ целые числа со знаком;
- ♦ числа с фиксированной запятой;
- ♦ числа с плавающей запятой.

Рассмотрим существующие формы представления чисел.

Наиболее просто представляются ***целые числа без знака***. При 8-ми разрядной шине данных числа представлены одним байтом и возможный диапазон изменения составляет от 0 до 255. Этого, конечно,

недостаточно при проведении большинства вычислений. Для повышения точности вычислений увеличивают разрядность чисел до двух байт. В этом случае диапазон допустимого изменения чисел составляет от 0 до 65535.

Операции с числами в такой форме представления часто называют вычислением с двойной точностью. При проведении вычислений с двойной точностью надо помнить о том, что время выполнения операций возрастает примерно в 1.5 - 2 раза, а при выполнении арифметических действий необходимо учитывать возникающую единицу переноса из младшего байта в старший.

При представлении ***целых чисел со знаком*** старший бит числа отводится для представления знака. Для положительных чисел знаковый бит равен 0, для отрицательных - 1. Диапазон изменения чисел в этом случае составляет от -128 до +127. Отрицательные числа получаются обратным отсчетом от нуля. Это действие аналогично работе вычитающего счетчика - если содержимое счетчика равно 0000-0000 и происходит уменьшение этого числа на единицу, то следующим показанием счетчика будет 1111-1111 (0FFH). Следовательно, 0FFH является шестнадцатеричным представлением числа -1. Такое представление соответствует дополнительному коду числа.

При представлении чисел со знаком в виде двух байт диапазон изменения составляет от -32768 до +32767. Двойная точность расширяет диапазон представления целых чисел, однако, как быть с числами меньше единицы или числами, лежащими между 3 и 4, т.е. дробными числами?

Представление дробных чисел производится в форме с фиксированной или плавающей запятой.

В форме с фиксированной запятой для хранения целой и дробных частей отводят фиксированное число разрядов, например, два байта. Первый байт - под целую часть, а второй - под дробную. Указанный способ позволяет, например, представлять столь малые числа, как 2^{-8} = 1/256, а также дробные числа, например, 3.17. Однако разрешающая способность для дробных чисел при этом методе ограничивается величиной 1/256 ≅ 0.004, а представляемые числа лежат в диапазоне ± 127. Область применения указанного способа можно расширить, если использовать для записи каждого числа не один, а несколько байт. Однако это требует относительно большого объема памяти и существенно снижает скорость вычислений.

Представление дробных чисел в формате с плавающей запятой позволяет использовать память МП системы более рационально. В этом случае число представляется в виде отдельных частей: мантиссы и порядка. Мантисса - это собственное значение числа, лежащие в диапазоне от 0 до 1. Например, число 360 0000 0000 можно записать как $0.36 \cdot 10^{12}$, при этом 0.36 - мантисса, а 12 - порядок числа.

Предположим, что для представления каждого числа используется два байта. Один байт отводится под мантиссу, а другой для порядка числа. Диапазон изменения чисел, при условии, что и мантисса и порядок представлены в дополнительных кодах, составляет $\pm 0.127 \cdot 2^{\pm 127}$. Это очень большой диапазон изменения чисел. В целях дальнейшего повышения разрешающей способности часто увеличивают количество разрядов мантиссы за счет использования большего количества байт.

2.5 Контрольные вопросы

1. Почему в цифровой технике широкое распространение получили двоичная, восьмеричная, шестнадцатеричная системы счисления?
2. Для чего нужна двоично-десятичная система счисления?
3. Какие системы кодирования двоично-десятичных чисел Вы знаете? Чем они отличаются друг от друга?
4. Для чего нужен перевод чисел из одной системы счисления в другую?
5. Как отличать двоичное число 10110 от восьмеричного 10110, шестнадцатеричного 10110 и десятичного 10110?
6. Объясните суть схемы Горнера.
7. Почему после прямого и обратного преобразования число часто отличается от исходного в дробных разрядах?
8. Что дает использование дополнительного кода в арифметических операциях?
9. Какое преимущество имеет модифицированный дополнительный код?
10. Почему операция суммирования является основной среди арифметических операций?
11. Для чего применяется код Грея?

3 МИКРОПРОЦЕССОРЫ В РАДИОТЕХНИЧЕСКИХ УСТРОЙСТВАХ И СИСТЕМАХ

3.1 Общие понятия

Наиболее совершенной из разработанных цифровых систем является электронная вычислительная машина (ЭВМ) с хранимой программой. Гипотетический вычислитель, демонстрирующий логику и методику программирования, впервые был предложен Тьюрингом. При всем разнообразии современных ЭВМ, все они выполняют одни и те же основные функции: ввод информации, хранение, арифметическое и логическое преобразование, вывод информации и управление работой входящих в структуру устройств.

Повторим некоторые понятия, необходимые для изучения материала пособия.

Микропроцессор (МП) – это набор компонент, реализованных в виде одной микросхемы или небольшого числа микросхем, выполняющих арифметические и логические операции над цифровыми данными и осуществляющих программное управление вычислительным процессом.

Микропроцессоры служат основой для создания различных универсальных и специализированных микро-ЭВМ, программируемых микроконтроллеров, микропроцессорных приборов и устройств контроля, управления и обработки данных.

Микро-ЭВМ называют устройство обработки данных, содержащее один или несколько микропроцессоров, интегральные схемы постоянной и оперативной памяти, схемы управления вводом/выводом информации и некоторые другие схемы, связанные друг с другом с помощью сигнальных шин. Однокристальные микро-ЭВМ реализованы в виде одной микросхемы.

Микро-ЭВМ с небольшими вычислительными ресурсами и упрощенной системой команд, ориентированную не на производство вычислений, а на выполнение процедур логического управления различным оборудованием, называют *программируемым микроконтроллером* или просто ***микроконтроллером.***

3.2 Архитектуры МП

Не претендуя на полноту, рассмотрим некоторые особенности построения и применения микропроцессоров и микропроцессорных систем.

При анализе системы команд различных МП можно выделить две *архитектуры микропроцессоров*:

- *CISC* – МП с полным набором команд (Complex Instruction Set Computer);
- *RISC* – МП с сокращенным набором команд (Reduced Instruction Set Computer).

Наиболее важными отличительными особенностями CISC-процессоров являются:

- большое число команд, обычно до 250;
- команды, которые предназначены для выполнения специализированных заданий и используются довольно редко;
- как правило, от 5 до 20 различных адресных режимов;
- форматы с переменной длиной команды;
- команды, которые управляют операндами в памяти;
- специализированные команды, которые предназначены для выполнения некоторых процедур высокоуровневых языков и позволяют сократить машинный код;
- микропрограммное управление в микропроцессоре.

Идея RISC-архитектуры – попытаться уменьшить время исполнения команд путем упрощения набора команд и добиться исполнения одной операции за один такт. Наиболее важными отличительными особенностями микропроцессора с RISC-архитектурой следует отметить:

- простой формат команды, относительно небольшое количество команд;
- относительно небольшое количество адресных режимов: регистровая адресация, непосредственная, относительная;
- относительно большое количество регистров, что способствует обработке операций внутри пространства регистров микропроцессора;
- перекрывающиеся окна регистров;
- команды обращения к памяти, включающие только операции загрузки и записи;
- выполнение одной команды за один такт с использованием конвейеризации;

- аппаратное, а не микропрограммное управление;

Достоинства и недостатки каждого из этих подходов довольно прозрачны. CISC-архитектура проигрывает в быстродействии RISC-архитектуре, но выигрывает по сложности реализуемых функций и уменьшению размера программ.

Каждая команда CISC-процессора является микропрограммой и состоит из нескольких микрокоманд. Поэтому даже самые простые команды обычно не могут выполняться за один такт. Зато язык ассемблера CISC-процессора намного легче для пользователя. Программы получаются более компактными, занимают меньше памяти и требуют меньше времени на разработку и отладку, по аналогии с крупнопанельным домостроением. CISC-процессор при выполнении программы меньше загружает системную шину, что может быть существенным в мультипроцессорных системах. При этом, к сожалению, в архитектуре сравнительно мало пользовательских регистров. Ведь большое количество регистров занимает существенное число адресов, а это значительно увеличивает длину и число команд, а их и так довольно много.

С другой стороны, далеко не всегда полностью используется вся мощность системы команд CISC-архитектуры. Это приводит к простою дорогостоящей аппаратуры. Выгодой другой архитектуры является возможность оптимизации сложных операций за счет простоты RISC команд. Это повышает быстродействие МП RISC-системы. Очень существенную выгоду дает наличие большого числа внутренних пользовательских регистров в этой архитектуре. Однако с точки зрения пользователя сложнее программировать для данной архитектуры. Для решения одинаковых задач RISC-программа будет иметь большую длину за счет простых команд.

Можно еще долго продолжать этот ряд сравнений указанных архитектур. Но нам важны, скорее, общие представления о различиях и сходствах МП архитектур.

По *областям применения* можно выделить универсальные микропроцессоры, специализированные микропроцессоры и микроконтроллеры.

Универсальные микропроцессоры предназначены для применения в вычислительных системах: персональных ЭВМ, рабочих станциях, в массово-параллельных супер-ЭВМ. Основной их характеристикой является наличие развитых средств для эффективной реализации операций с плавающей точкой над многоразрядными (64 и более разрядов)

операндами. Предназначаются они в основном для проведения научно-технических расчетов.

Огромную часть множества ***специализированных процессоров*** составляют процессоры цифровой обработки сигналов (цифровые сигнальные процессоры DSP). Цифровые сигнальные процессоры рассчитаны на обработку в реальном времени цифровых потоков, образованных путем оцифровки аналоговых сигналов. Это обусловливает их сравнительно малую разрядность и преимущественно целочисленную обработку. Однако современные сигнальные процессоры способны проводить вычисления с плавающей точкой над 32-40-разрядными операндами. Кроме того, появился класс медийных процессоров, представляющих собой законченные системы для обработки аудио- и видеоинформации.

Наибольшей специализацией по областям и разнообразием реализуемых функций обладают ***микроконтроллеры***, используемые во встроенных системах управления, в том числе в бытовых приборах. Общее число типов кристаллов с различными системами команд превышает 500, и все они, в силу существования изделий с их использованием, имеют свою устойчивую долю рынка.

3.3 Характеристики МП

При указании *характеристик микропроцессоров* часто используются следующие параметры:

- ***Тип арифметики***. Здесь говорят о формах представления и обработки данных в МП: с фиксированной или плавающей точкой. Форма с плавающей точкой более предпочтительна в расчетных задачах. Формат фиксированной точки лучше подходит для управленческих задач. Процессоры с плавающей точкой являются более сложными и дорогими устройствами. Процессоры с фиксированной точкой применяются чаще.
- ***Разрядность данных***. Количество двоичных разрядов для представления данных при их обработке. Количество битов обычно кратно числу 8. Типовые значения разрядности: 8, 16, 24, 32, 48, 64 бита и т.д. Параметр часто равен разрядности шины данных МП системы.
- ***Объем, разновидность внутренней памяти.*** Эти параметры наиболее часто используют для микро-ЭВМ, особенно однокристальных. Наличие встроенного *ОЗУ* (оперативное запоминающее устрой-

ство с прямым доступом или *RAM* – Random Access Memory) определяет возможности построения системы без использования внешней памяти, применяемой как для хранения данных, так и загружаемой программы. Наличие встроенного *ПЗУ* (постоянное запоминающее устройство или *ROM* – Read Only Memory), программируемого при изготовлении процессора на заводе или «на рабочем месте» позволяет минимизировать размер разрабатываемой прикладной системы и защищать разработанное программное обеспечение от пиратского копирования. Память типа *OTP ROM* (One Time Programmable ROM, однократно программируемое ПЗУ) выгодна при тиражировании уже работающей системы. Память типа *Flash* позволяет неоднократно перезаписывать программу и данные в системе даже непосредственно на изготовленной плате системы или даже в процессе работы. Очень удобна при моделировании системы и отладке новых программ.

- ***Объем адресного пространства памяти***. Определяется разрядностью шины адреса. Характеризует максимально возможный общий объем памяти, используемый в МП системе. В реальной системе объем подключенной памяти определяется минимальными потребностями и часто меньше допустимого объема памяти.
- ***Количество и вид портов ввода/вывода информации***. Параметр определяет возможности МП по связи с внешними устройствами.
- ***Напряжение питания, мощность потребления***. Параметры важны для МП, применяемых в системах с автономным батарейным питанием. Потребляемая мощность часто зависит от выполняемой программы. Во многих процессорах допустимы режимы пониженного потребления: холостой ход, ожидание, сон. Общая тенденция показывает постепенное снижение напряжения питания для разрабатываемых новых МП. Это актуально и с целью решения проблем теплоотвода, обусловленных увеличивающейся плотностью компонентов на единице площади электронного кристалла.
- ***Быстродействие.*** Для характеристики быстродействия применяют различные параметры, но каждый из них не является исчерпывающим. Рассмотрим часто применяемые характеристики быстродействия.
- ***Тактовая частота*** работы МП. Обычно указывается внешняя тактовая частота, подаваемая на процессор. Внутренняя частота может отличаться от внешней из-за наличия системы деления или умножения

частоты. Поэтому параметр не может полностью характеризовать быстродействие процессора.

- ***Время командного цикла***. Параметр связан с внутренней частотой работы процессора. Определяет длительность интервала времени (количество тактов внутренней частоты) для реализации команды. Это самая неоднозначная характеристика быстродействия, поскольку отдельная операция может выполняться как за несколько тактов, так и за один. К тому же, в некоторых процессорах используется параллельное выполнение команд.
- ***Количество миллионов команд, выполняемых за секунду MIPS*** (Millions instructions per second). Параметр плохо характеризует быстродействие, если используются комбинированные команды, для которых одновременно выполняется несколько операций. Поэтому в разных процессорах одной команде может соответствовать различная по объему работа.
- ***Количество миллионов операций за секунду MOPS*** (Millions operations per second). Эта характеристика лучше отражает реальное быстродействие, поскольку учитывает и параллельно выполняемые команды, и наличие нескольких операционных модулей, функционирующих одновременно. Только отсутствие стандартного определения операции несколько смазывает картину. Под операцией можно понимать сложение операндов, выборку команд или запись в память полученных результатов. А указанные действия сильно различаются по значимости.
- ***Количество миллионов операций с плавающей точкой за секунду MFLOPS*** (Millions of floating-point operations per second). Этот параметр применяется в процессорах с плавающей запятой. Имеет аналогичные MOPS достоинства и недостатки.
- ***Количество единиц*** какого-нибудь ***теста***. Выбирают единую последовательность некоторых операций (например, обработка определенного массива при вычислении суммы, разности, произведения элементов или поиска наибольшего/наименьшего компонента и т.д.) в качестве теста и реализуют ее на разных МП. Отношение времен выполнения и дает сравнительную характеристику быстродействия.

Применение МП в устройствах и системах радиосвязи, радиовещания и телевидения обычно связано с двумя возможными вариантами. В первом случае на базе микропроцессорной структуры или однокристальной микро-ЭВМ реализуется система управления устройствами.

Второй случай связан с построением МП системы цифровой обработки сигналов.

3.4 МП в системах обработки сигналов

Другой областью применения микропроцессоров и микроконтроллеров в радиотехнических задачах являются задачи цифровой обработки сигналов и изображений.

Среди задач цифровой обработки можно назвать преобразование Фурье, свертку функций, кодирование, цифровую фильтрацию, спектральные преобразования. В типичном случае входной аналоговый сигнал с выхода датчика (видеосигнал, радиосигнал) подвергается аналогово-цифровому преобразованию (АЦП) (рис. 3.1).

Рисунок 3.1– МП для задач обработки сигналов

Оцифрованный и дискретизованный сигнал X(nT) является входным для микропроцессора или микро-ЭВМ. В микро-ЭВМ обработка входных данных выполняется с помощью программных средств или аппаратных ресурсов по заранее заданному алгоритму. Выходной цифровой сигнал часто подвергается обратному преобразованию – цифро-аналоговому (ЦАП).

Для каждой из указанных областей применения имеются свои классы микропроцессоров и микро-ЭВМ, наиболее приспособленных для решения соответствующих задач. Использование МП в «своей» области позволяет обеспечить наилучшие характеристики по производительности системы и ее стоимости.

3.5 МП в системах управления

Под ***управлением*** обычно понимают целенаправленное воздействие на объект, в результате которого он переходит в требуемое состояние. ***Объектом*** управления назовем ту часть окружающего мира, на которую можно воздействовать с определенной целью. В качестве объ-

ектов управления можно понимать различные природные и искусственные системы, устройства, явления.

В нашем случае в качестве объектов управления будем понимать отдельные компоненты радиоэлектронной аппаратуры и всю систему в целом.

В каждый момент времени объект находится в одном из своих возможных состояний. Любой объект управления (рисунок 3.2) существует не сам по себе, а в окружающей его среде.

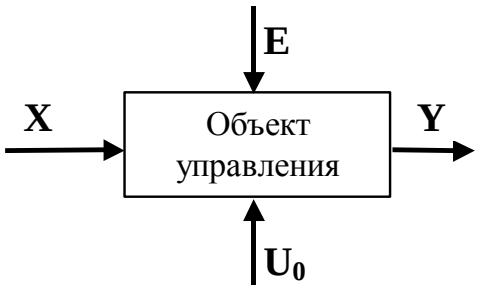

Рисунок 3.2– Взаимодействие объекта управления с окружающей средой

Среда постоянно воздействует на состояние объекта. Эти воздействия можно разделить на три группы:

- объективно существующие и наблюдаемые воздействия (вход объекта **X**);
- управляющие воздействия, с помощью которых происходит управление объектом (управляющий вход объекта $\mathbf{U_0}$);
- неизмеряемые параметры среды и случайные изменения объекта (вход возмущений **E**).

Состояние объекта (выход объекта **Y)** можно представить параметрами, характеризующими его в каждый момент времени.

Управляющие воздействия $\mathbf{U_0}$ подаются на объект с определенной целью. ***Цель управления*** – это требуемое состояние или последовательность состояний объекта во времени. Цель должна быть описана с помощью параметров **Y.**

Таким образом, для управления объектом необходима следующая информация:

- перечень возможных состояний объекта;
- перечень входных параметров объекта и диапазоны изменения их значений;

- допустимые управляющие воздействия;
- характер возмущений;
- цель управления объектом.

На основе этой информации для достижения цели управления требуются алгоритмы и средства преобразования входов объекта в необходимые управляющие воздействия и цели управления – в последовательность состояний объекта. Под ***алгоритмом*** понимают конечный набор правил для однозначного преобразования исходных данных в выходные.

На рисунке **3.3** приведена обобщенная структура системы управления. В ней имеются датчики, предназначенные для измерения состояний внешней среды и объекта управления, устройство управления, формирующее команды управления, исполнительные устройства, преобразующие команды в управляющие воздействия на входе управления объекта.

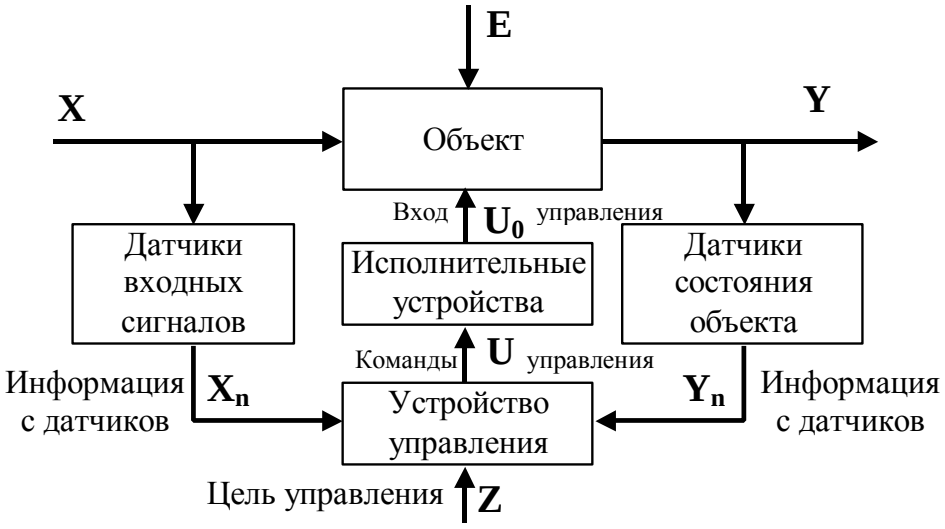

Рисунок 3.3– Структурная схема системы управления

Для целенаправленного функционирования устройства управления ему необходимо задать цель управления **Z**. Достижение цели осуществляется по алгоритму управления, представляющему собой набор блоков аппаратных средств или набор программ.

Программы используются в случае, когда в качестве устройства управления применяются управляющие микроконтроллеры или микро-ЭВМ. Важной особенностью работы таких управляющих микро-ЭВМ и контроллеров, в отличие от обычных микро-ЭВМ, является выполнение ими всех операций в реальном масштабе времени.

Термин «реальное время» обозначает, что задержка реакции устройства управления должна быть конечной и не превышать определенного значения. Это касается не столько скоростных характеристик микро-ЭВМ, сколько относится к сложности алгоритмов и программ, реализованных в управляющей микро-ЭВМ.

При работе микро-ЭВМ и контроллеров в составе системы управления им приходится выполнять различные действия:

- принимать информацию от датчиков о состоянии окружающей среды и объекта;
- рассчитывать в реальном времени управляющие воздействия и передавать их на исполнительные устройства;
- отображать, при необходимости, информацию о текущем состоянии системы пользователю на индикаторах;
- принимать и обрабатывать команды пользователя по изменению условий процесса управления.

Управляющие микро-ЭВМ отличаются от обычных микро-ЭВМ и способом разработки программного обеспечения. Микро-ЭВМ, встроенные в оборудование, не имеют соответствующего набора внешних устройств (дисплеи, принтеры, внешние накопители и т.д.) и поэтому не пригодны для разработки и отладки программного обеспечения. В этом случае, либо программное обеспечение создается на аналогичной микро-ЭВМ, но имеющей необходимый набор внешних устройств, либо программы разрабатываются и отлаживаются на универсальных ЭВМ, с использованием симуляторов – программных моделей управляющей микро-ЭВМ или контроллера.

Последний способ разработки программного обеспечения называется ***кросс-технологией***. При этом применяются программы: кросс-транслятор, кросс-компоновщик и кросс-отладчик.

Как было определено ранее, наиболее подходящими для задач управления являются микро-ЭВМ, особенно однокристальные и микроконтроллеры. Для полноты знаний разработчик и пользователь должны представлять структуру микро-ЭВМ. Это позволит повысить эффективность новых разработок, поднять производительность (оперативность) существующих систем управления.

Огромное множество микро-ЭВМ и микроконтроллеров не позволяет рассмотреть их все. Но некоторые микро-ЭВМ нашли широкое применение. Их популярность подтверждается большим числом наработок на их основе, пополняющимся банком полезных программ и биб-

лиотек, доступностью во всем мире, выпуском новых моделей, совместимых на программном уровне. Этот факт можно объяснить удачным набором компонентов структуры микро-ЭВМ, высокими рабочими характеристиками и простотой программирования.

3.6 Контрольные вопросы

1. Чем отличается микропроцессор от микроконтроллера.
2. Расскажите особенности RISC и CISC архитектуры МП
3. Чем отличаются универсальные МП от специализированных МП.
4. Какие Вы знаете характеристики микроппроцессоров.
5. Чем отличается MIPS от MOPS?

4 ОДНОКРИСТАЛЬНЫЙ МИКРОКОНТРОЛЛЕР СЕМЕЙСТВА MCS-51

4.1 Структура ОЭВМ КМ1816ВЕ51

КМ1816ВЕ51 – высоко интегрированный 8-битный микроконтроллер, основанный на CISC-архитектуре MCS®-51. Семейство микроконтроллеров MCS-51 весьма многочисленно, разнообразно и выпускается многими фирмами. Одна только фирма INTEL выпустила более 50 видов однокристальных МК этого семейства. Различия видов состоят в размере внутренней памяти программ с объемом от 0 Кбайт до 32 Кбайт, ее типе, объеме резидентной памяти данных от 128 до 256 байт, тактовой частоте от 12 МГц до 24 МГц, количестве таймеров/счетчиков от 2 до 3, наличии АЦП, типе корпуса, технологии изготовления и др. (табл. 4.1)

Таблица 4.1 – Основные характеристики ряда МК семейства MSC 51

Тип прибора	Объем памяти программ	Объем памяти данных, байт	Макс. такт. частота, МГц	Таймеры/ счетчики, кол-во и разрядность	Другие отличия
8051AH	4 Кбайт	128x8	12	2x16	32 линий I/O, UART
8052AH	8 Кбайт	256x8	12	3x16	32 линии I/O, UART
87C51	4 Кбайт	128x8	20	2x16	32 линии I/O, UART
87C54	16 Кбайт	256x8	24	3x16	32 линии I/O, UART
87L54	16 Кбайт	256x8	20	3x16	32 линии I/O, UART, пит. 2.7 В
87L58	32 Кбайт	256x8	20	3x16	32 линии I/O, UART, пит. 2.7 В
80C152JB	8 Кбайт	256x8	16	2x16	56 линии I/O, UART
87C51GB	8 Кбайт	256x8	16	3x16	48 линии I/O, UART, 8x8 АЦП, програмир. счетный массив
87C51SL-AL	16 Кбайт	256x8	16	2x16	87 линии I/O, UART, 4x8 АЦП, питание 3-3.6 В

Микроконтроллеры семейства MSC-51 выполнены по HMOS и CHMOS технологии в различных корпусах (PDIP, PLSS, SQFP).

КМ1816ВЕ51 (MCS-51) выполнен по n-МОП технологии в корпусе PDIP с 40 выводами. Информационные выводы MCS-51 совместимы с ТТЛ-схемами по входу, выходу и третьему состоянию.

В MCS-51 имеются следующие аппаратные возможности и периферийные устройства:

- Четыре 8-битных двунаправленных параллельных порта ввода/вывода.
- Два 16-битных таймера/счетчика с программируемой частотой выходного сигнала.
- Полный дуплексный программируемый последовательный порт с разными скоростями работы и режимами обмена.
- 4 Кбайт резидентной памяти программ с возможностью расширения до 64 Кбайт за счет внешней памяти программ.
- 128 байт резидентной памяти данных с возможностью расширения до 64 Кбайт за счет подключения внешней памяти данных.
- Один источник питания +5В.
- Асинхронный сброс.
- Максимальная тактовая частота 12 МГц.
- 5 источников прерывания с двумя уровнями приоритета.
- Режим пониженного энергопотребления и холостого хода.

В основе структурной схемы MCS-51 (рис. 4.1) имеется 8-битная внутренняя шина, к которой подключены практически все элементы: резидентная память данных (РПД), резидентная память программ (РПП), арифметико-логическое устройство (АЛУ), устройство управления, порты ввода/вывода, регистры специального назначения SFR (РСФ).

4.2 Организация памяти MCS-51

4.2.1 Разделение памяти

Для всего семейства MCS-51 характерно разделение адресного пространства для памяти программ и памяти данных. Такая архитектура ЭВМ получила название «Гарвардская». Память разделена на логическом и физическом уровнях, управляется разными сигналами и выполняет разные функции. Разделение затрагивает как резидентную память, так и внешнюю память.

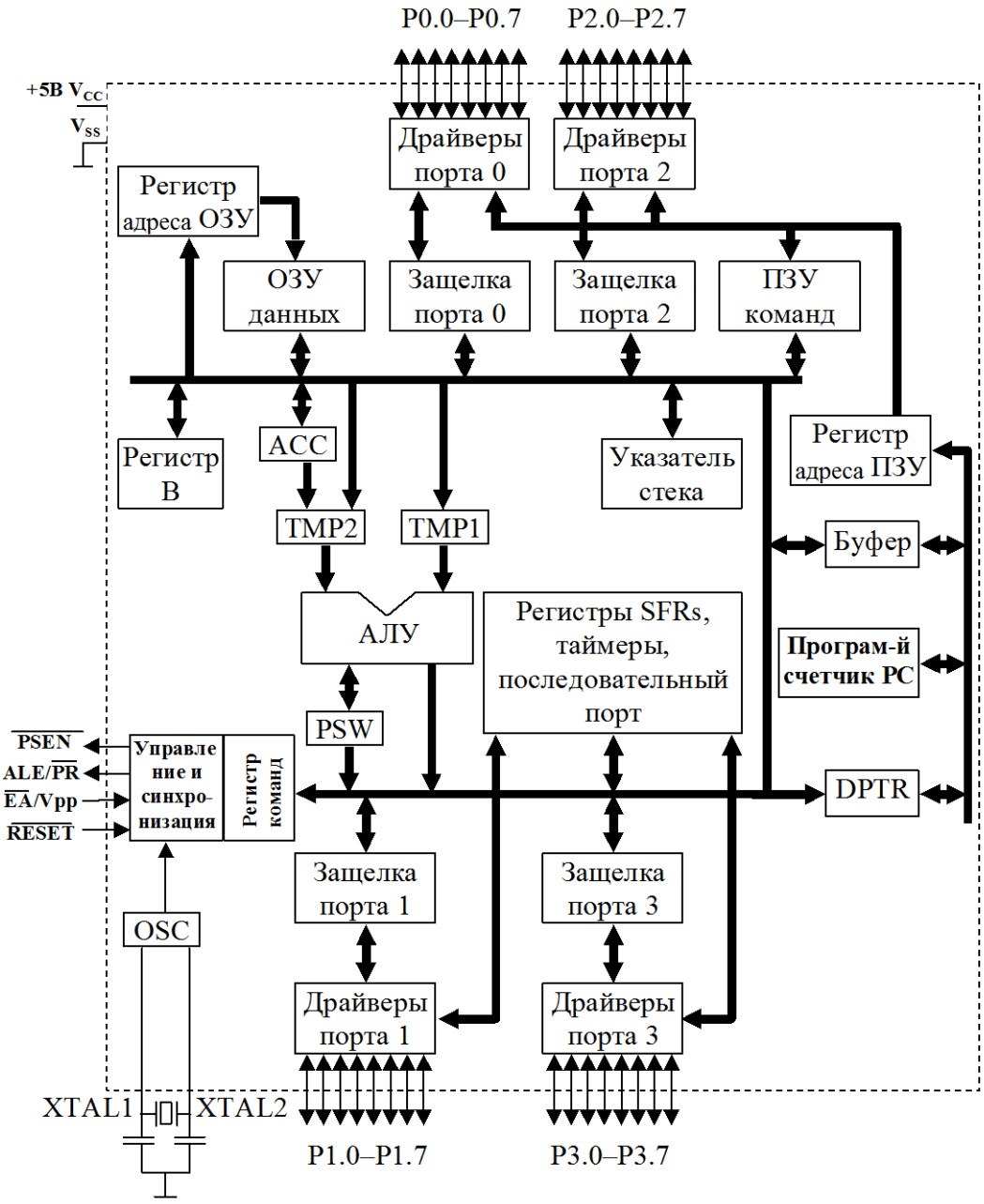

Рисунок 4.1 – Структурная схема ОЭВМ MCS-51

Альтернативой является структура ЭВМ «фон Неймана», имеющая общую память программ и данных.

4.2.2 Память программ

На кристалле MCS-51 расположена резидентная память программ (РПП) объемом 4 Кбайт. Память программ используется для хранения:

a) команд;

b) констант;

c) управляющих слов инициализации;

d) таблицы перекодировки входных и выходных данных.

После выполнения сигнала сброса $\overline{RESET}$ МК начинает выполнение программы с адреса 0000. При любой конфигурации памяти программ из нее можно только читать информацию, и принципиально отсутствует возможность производить в нее запись средствами МК.

За счет подключения внешней памяти программ (ВПП) суммарный объем памяти программ можно довести до 64 Кбайт.

4.2.3 Память данных

Внутреннее ОЗУ данных МК содержит 256 байт (рис. 4.2). Адресное пространство памяти данных разделено на два блока: 128 байт резидентной памяти данных (РПД) и область регистров специальных функций (SFR). В памяти данных хранятся переменные, используемые при выполнении прикладной программы. Все байты ОЗУ данных могут быть доступны с помощью либо прямой, либо косвенной адресации, для адресации используется один байт.

Самые младшие 32 байта РПД сгруппированы в 4 банка по 8 регистров. К регистрам можно обращаться по именам как R0-R7. В каждый момент времени может быть активным только один из четырех банков. Для переключения банков используются программно изменяемые биты RS1 и RS0 в слове состояния процессора PSW.

В интервале адресов РПД с 32 (20H) по 47 (2FH) адрес расположена область битовой адресации, содержащая 128 битов пользователя. Обращение в этой области возможно как к байтам, так и к отдельным битам при использовании прямого 8-битового адреса (bit).

В интервале адресов РПД с 48 (30H) по 127 (7FH) адрес расположена сверхоперативная область, в которую можно записывать и считывать байты с помощью прямого адреса (ad) либо с помощью косвенной адресации, используя регистры R0 или R1 в качестве байта адреса (MOV @Ri).

Начальные значения всех ячеек РПД являются случайными величинами. Изменение значения любой ячейки РПД происходит только при операции записи в нее байта.

Память данных можно расширить за счет подключения до 64 Кбайт внешней памяти (ВПД).

Адреса РПД	(D7)		Прямые адреса битов области битовой адресации					(D0)	
7FH	Сверхоперативная область ЗУ								
30H									
2FH	7F	7E	7D	7C	7B	7A	79	78	Область битовой адресации байтов РПД
2EH	77	76	75	74	73	72	71	70	
2DH	6F	6E	6D	6C	6B	6A	69	68	
2CH	67	66	65	64	63	62	61	60	
2BH	5F	5E	5D	5C	5B	5A	59	58	
2AH	57	56	55	54	53	52	51	50	
29H	4F	4E	4D	4C	4B	4A	49	48	
28H	47	46	45	44	43	42	41	40	
27H	3F	3E	3D	3C	3B	3A	39	38	
26H	37	36	35	34	33	32	31	30	
25H	2F	2E	2D	2C	2B	2A	29	28	
24H	27	26	25	24	23	22	21	20	
23H	1F	1E	1D	1C	1B	1A	19	18	
22H	17	16	15	14	13	12	11	10	
21H	0F	0E	0D	0C	0B	0A	09	08	
20H	07	06	05	04	03	02	01	00	
1FH	Банк 3 (8 регистров)								**Банки регистров РОН**
18H									
17H	Банк 2 (8 регистров)								
10H									
0FH	Банк 1 (8 регистров)								
08H									
07H	R7	Банк 0 (8 регистров)							
06H	R6								
05H	R5								
04H	R4								
03H	R3								
02H	R2								
01H	R1								
00H	R0								

Рисунок 4.2 – Распределение резидентной памяти данных

4.3 Регистры специального назначения

К области РПД примыкает область регистров специальных функций (специального назначения), адреса которых расположены в интервале от 128 (80H) до 255 (0FFH) адреса. К этим регистрам специального назначения относятся регистры-защелки портов, регистры таймеров, регистры управления периферийными устройствами, аккумулятор и т.д.

Из распределения адресов области регистров специального назначения (табл. 4.2) видно, что не все адреса заняты. Не используемые адреса физически отсутствуют на кристалле. Чтение по этим адресам возвращает случайные значения, а запись не результативна. Пользователю не рекомендуется производить запись по незанятым адресам, так как эти адреса могут быть задействованы в новых версиях MCS-51.

Таблица 4.2 – Блок регистров специального назначения SFR

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

Среди SFR есть несколько регистров, допускающих как байтовую, так и битовую адресацию. Двоичные адреса этих регистров оканчива-

ются на три нуля, т.е. имеют в конце цифру 0 или 8H. К отдельным битам этих регистров можно обращаться при использовании прямого 8-битового адреса (bit) в диапазоне 128 (80H) – 255(0FFH) (рис. 4.3).

Прямой адрес байта	(D7)		Прямой адрес бита					(D0)	Имя регистра
0FFh 0F1h	Несуществующие ячейки								
0F0h	**F7**	**F6**	**F5**	**F4**	**F3**	**F2**	**F1**	**F0**	**B**
0EFh 0E1h	Несуществующие ячейки								
0E0h	**E7**	**E6**	**E5**	**E4**	**E3**	**E2**	**E1**	**E0**	**Acc**
0DFh 0D1h	Несуществующие ячейки								
0D0h	**D7**	**D6**	**D5**	**D4**	**D3**	**D2**	**D1**	**D0**	**PSW**
0CFh 0B9h	Несуществующие ячейки								
0B8h	**–**	**–**	**–**	**BC**	**BB**	**BA**	**B9**	**B8**	**IP**
0B7h 0B1h	Несуществующие ячейки								
0B0h	**B7**	**B6**	**B5**	**B4**	**B3**	**B2**	**B1**	**B0**	**P3**
0AFh 0A9h	Несуществующие ячейки								
0A8h	**AF**	**–**	**–**	**AC**	**AB**	**AA**	**A9**	**A8**	**IE**
0A7h 0A1h	Несуществующие ячейки								
0A0h	**A7**	**A6**	**A5**	**A4**	**A3**	**A2**	**A1**	**A0**	**P2**
09Fh 099h	Несуществующие ячейки								
098h	**9F**	**9E**	**9D**	**9C**	**9B**	**9A**	**99**	**98**	**SCON**
097h 091h	Несуществующие ячейки								
090h	**97**	**96**	**95**	**94**	**93**	**92**	**91**	**90**	**P1**
08Fh 089h	Несуществующие ячейки								
088h	**8F**	**8E**	**8D**	**8C**	**8B**	**8A**	**89**	**88**	**TCON**
087h 081h	Несуществующие ячейки								
080h	**87**	**86**	**85**	**84**	**83**	**82**	**81**	**80**	**P0**

Рисунок 4.3 – Карта адресуемых бит в SFR

Назначением этих битов, чаще всего, является управление режимами и контроль состояния периферийных устройств и микроконтрол-

лера. Назначение и выполняемые этими битами функции будут рассмотрены ниже, при описании периферийных устройств.

Рассмотрим функции регистров, входящих в блок SFR.

Регистр аккумулятора (ACC). Аккумулятор является источником операнда и местом записи результата при выполнении арифметических, логических операций, операций сдвига, проверки на нуль и т.п. В зависимости от вида адресации аккумулятор в инструкциях именуется как (A) или как (ACC) (при задании прямого адреса).

Регистр B. Используется в операциях умножения и деления. В других случаях можно использовать как обычный сверхоперативный регистр.

Указатель стека (SP). 8-разрядный регистр SP увеличивается на 1 (инкрементируется) перед записью информации в память при выполнении операций, связанных со стеком, типа PUSH и CALL. Стек может занимать любое место в РПД. После сброса указателю SP присваивается значение 07H, означая, что по умолчанию стек начинается с адреса 08H.

Указатель данных (DPTR). Указатель данных состоит из старшего байта (DPH) и младшего (DPL). Предназначен для задания 16-битного адреса в операциях с обращением к внешней памяти ВПД, но может использоваться как рабочий 16-разрядный регистр или как два независимых 8-битных регистра.

Регистры портов (P0 – P3). P0, P1, P2 и P3 являются регистрами-защелками портов 0 – 3. Через эти регистры выполняется обращение к портам ввода/вывода

Регистры таймеров. Регистровые пары (**TH0, TL0**), (**TH1, TL1**) – это 16-битные счетные регистры для таймеров/счетчиков 0 и 1. Для управления таймерами используется регистр состояния (**TCON**) и регистр управления (**TMOD**).

Регистры последовательного порта. Регистр последовательных данных (**SBUF**) – в действительности это два раздельных регистра: передающий и принимающий буферные регистры. Для управления и контроля состояниями последовательного порта используется регистр (**SCON**).

Регистры прерываний. В регистре (**IE**) находятся биты индивидуального разрешения прерываний. Для управления приоритетами используется регистр (**IP**).

Регистр управления энергопотреблением (PCON). Управляет скоростью последовательного порта, режимом пониженной мощности и режимом холостого хода.

Слово состояния процессора (PSW). Регистр содержит информацию о состоянии процессора (табл. 4.3) в виде битов признаков (флагов). Часть флагов изменяется только программно, другая часть флагов изменяется при выполнении большинства команд, использующих арифметико-логическое устройство.

Таблица 4.3 – Регистр PSW

<table>
<tr><th colspan="4">Регистр слова состояния процессора PSW Адрес 0D0h
Имеет битовую адресацию</th></tr>
<tr><th>Название бита</th><th>Адрес бита</th><th>Позиция бита</th><th>Имя и назначение</th></tr>
<tr><td>C</td><td>0D7H</td><td>PSW.7</td><td>Флаг переноса. Установка/сброс аппаратно или программно при выполнении арифметических и логических операций</td></tr>
<tr><td>AC</td><td>0D6H</td><td>PSW.6</td><td>Флаг вспомогательного переноса. Сброс/установка аппаратно, после команд сложения/вычитания. Отображает перенос/заем в бите 3 ACC</td></tr>
<tr><td>F0</td><td>0D5H</td><td>PSW.5</td><td>Флаг пользователя F0. Установка/сброс и проверка программно пользователем.</td></tr>
<tr><td>RS1
RS0</td><td>0D4H
0D3H</td><td>PSW.4
PSW.3</td><td>Выбор банка регистров R0-R7. Установка/ сброс программно для выбора банка.
<table>
<tr><th>RS 1</th><th>RS 0</th><th>Банк рабочих регистров</th><th>Адрес</th></tr>
<tr><td>0</td><td>0</td><td>Банк 0</td><td>(00H-07H)</td></tr>
<tr><td>0</td><td>1</td><td>Банк 1</td><td>(08H-0FH)</td></tr>
<tr><td>1</td><td>0</td><td>Банк 2</td><td>(10H-17H)</td></tr>
<tr><td>1</td><td>1</td><td>Банк 3</td><td>(18H-1FH)</td></tr>
</table></td></tr>
<tr><td>OV</td><td>0D2H</td><td>PSW.2</td><td>Флаг переполнения. Установка/сброс аппаратно при выполнении арифметических операций.</td></tr>
<tr><td>–</td><td>0D1H</td><td>PSW.1</td><td>Не используется</td></tr>
<tr><td>P</td><td>0D0H</td><td>PSW.0</td><td>Флаг паритета. Установка/сброс аппаратно в каждом цикле команды для указания четного/нечетного числа «1» в аккумуляторе</td></tr>
</table>

Наиболее «активный» флаг – это флаг переноса C, участвующий в арифметических, логических операциях и сдвигах, некоторых битовых операциях. Флаг переполнения (OV) используется при выполнении арифметических операций со знаковыми числами. Флаг вспомогательного переноса (AC) нужен для десятичной коррекции результата сложения или вычитания.

4.4 Устройство управления и синхронизации

К внешним выводам XTAL1 и XTAL2 MCS-51 подключается кварцевый резонатор, который управляет работой внутреннего генератора OSC. С его помощью устройство управления генерирует сигналы синхронизации. Устройство управления и синхронизации формирует машинный цикл фиксированной длительности. Каждый машинный цикл равен 12 периодам резонатора или 6 состояниям первичного управляющего автомата.

Состояния обозначаются как S1 – S6. Каждое состояние содержит две фазы P1 и P2. В фазе P1, как правило, выполняется операция в АЛУ, а в фазе P2 выполняется межрегистровая передача. Большинство сигналов устройства управления являются внутренними и ненаблюдаемыми. Внешними, наблюдаемыми сигналами синхронизации MCS-51 являются только сигнал резонатора XTAL2 и строб адреса внешней памяти ALE (рис. 4.4).

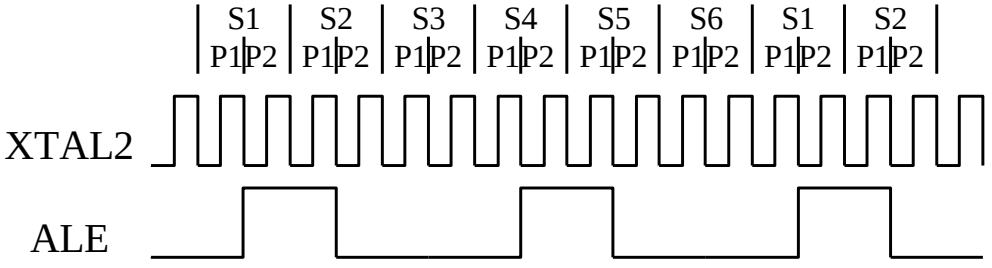

Рисунок 4.4 – Внешние сигналы синхронизации

Сигнал ALE формируется дважды за машинный цикл (S1P2 – S2P1 и S4P2 – S5P1). Этот сигнал используется для управления процессом обращения к внешней памяти. Этот же сигнал можно использовать для синхронизации работы и тактирования внешних устройств, подключенных к MCS-51.

Весь машинный цикл состоит из 12 фаз, начиная с S1P1 и кончая S6P2. Поэтому при частоте кварцевого резонатора 12 МГц период выполнения машинного цикла составляет $1 \cdot 10^{-6}$ сек. В этом случае период сигнала ALE составляет $0.5 \cdot 10^{-6}$ сек.

Большинство команд MCS-51 выполняются за один машинный цикл. Команды, оперирующие 16-битовыми словами или требующие обращения к внешней памяти программ или данных, выполняются за два машинных цикла. Только команды умножения и деления выполняются за четыре машинных цикла. На основе знания этих особенностей в работе схемы синхронизации выполняется расчет необходимого времени выполнения прикладных программ.

4.5 Порты ввода/вывода

Все четыре порта MCS-51 являются двунаправленными. Каждый из них состоит из регистра-защелки, выходного драйвера и входного буфера. Выходные драйверы портов P0 и P2 и входной буфер порта P0 используются при обращении к внешней памяти программ и внешней памяти данных. Через порт P0 в режиме временного мультиплексирования выводится младший байт адреса внешней памяти, а затем передается или принимается байт данных/команды. Через порт P2 выдается старший байт адреса при 16-битовых адресах.

Все выводы порта P3 могут быть использованы для альтернативных функций (табл. 4.4). Для разрешения альтернативных функций в соответствующие разряды регистра-защелки порта P3 необходимо записать «1».

Каждый бит любого порта может быть независимо настроен на ввод или вывод информации. Для использования разряда порта на ввод необходимо записать «1» в соответствующий триггер-защелку, что закрывает выходной МОП-транзистор. Есть некоторая разница в строении порта P0 и портов P1, P2 и P3. Порт P0 является двунаправленным, а порты P1 – P3 – квазидвунаправленными.

Для вывода информации в нужный разряд порта P0 (рис. 4.5) используется сигнал «Запись в порт». Сигнал «Управление» переключает мультиплексор для вывода через порт P0 байта адреса или данных, поступающих по линии «Адрес/данные». В этом режиме верхний МОП-транзистор открывается только при выдаче бита «1». В других случаях этот транзистор заперт. Поэтому линии порта P0, используемые для вывода, являются выходами с открытым стоком.

Таблица 4.4 – Альтернативные функции порта P3

Порт ввода/вывода P3 Адрес 0B0h Имеет битовую адресацию			
Название бита	Адрес бита	Позиция бита	Имя и назначение
$\overline{RD}$	0B7H	P3.7	Чтение. При выполнении команды чтения ВПД аппаратно формируется сигнал низкого уровня
$\overline{WR}$	0B6H	P3.6	Запись. При выполнении команды записи в ВПД аппаратно формируется сигнал низкого уровня
T1	0B5H	P3.5	Вход таймера/счетчика 1 или тест-вход.
T0	0B4H	P3.4	Вход таймера/счетчика 0 или тест-вход.
$\overline{INT1}$	0B3H	P3.3	Вход запроса прерывания 1. Активен спад или низкий уровень
$\overline{INT0}$	0B2H	P3.2	Вход запроса прерывания 0. Активен спад или низкий уровень
TXD	0B1H	P3.1	Выход передатчика последовательного порта. Выход синхронизации в режиме сдвигающего регистра
RXD	0B0H	P3.0	Вход приемника последовательного порта. Ввод/вывод данных в режиме сдвигающего регистра

Строение битов порта P2 близко к строению битов порта P0, но с некоторыми упрощениями, а строение порта P1 аналогично строению порта P3 (рис. 4.6), но без альтернативных функций.

Для выполнения альтернативных функций порта P3 в его схеме имеются соответствующие выводы – «Альтернативная функция выхода» и «Альтернативная функция входа». Для разрешения использования альтернативных функций в соответствующие регистры-защелки порта P3 необходимо записать «1».

При чтении информации из разрядов порта P0 и других портов могут использоваться два сигнала:

a) Чтение буфера-защелки («Чтение буфера»);

b) Чтение контакта вывода («Ввод»).

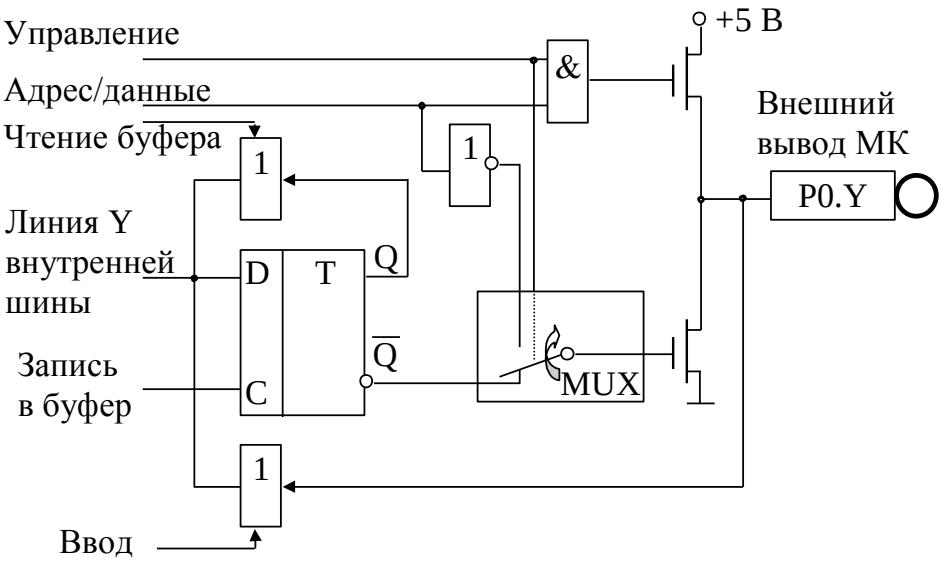

Рисунок 4.5 – Структура одного разряда порта P0

Соответственно, информация будет читаться из буфера-защелки или с внешнего вывода порта. Вид чтения определяется типом команды. Разные способы чтения необходимы, чтобы исключить неправильное чтение содержимого порта в некоторых командах, искажаемое за счет мощной внешней нагрузки разрядов порта.

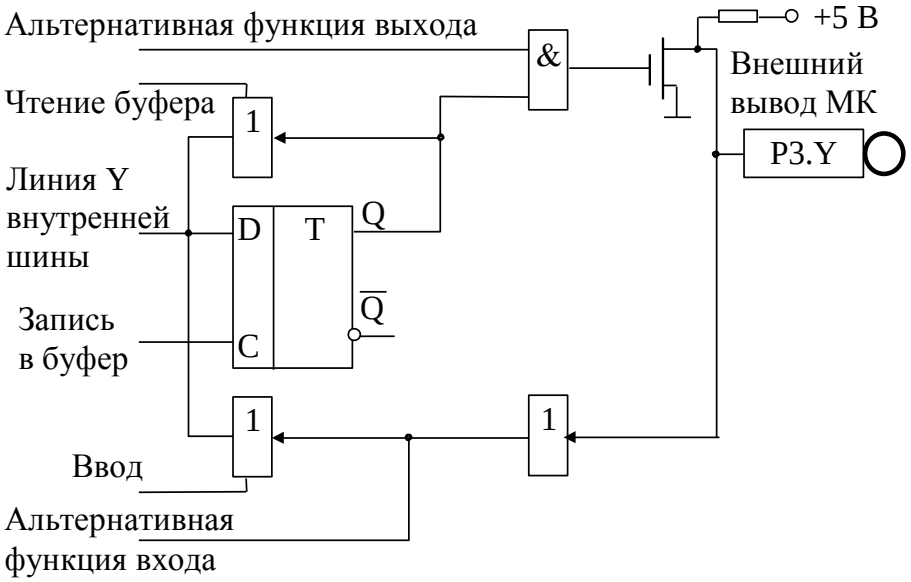

Рисунок 4.6 – Структура одного разряда порта P3

По сигналу сброса во все регистры-защелки всех портов записывается «1», и, таким образом, все порты настраиваются на режим ввода.

Выходные линии портов P1, P2 и P3 по нагрузочной способности могут работать на один вход ТТЛ-схемы. Выходы порта P0 могут быть подключены к двум ТТЛ-входам. Входные сигналы всех портов могут обслуживаться как выходами ТТЛ-схем, так и n-МОП элементами.

4.6 Доступ к внешней памяти

Доступ к внешней памяти бывает двух типов: доступ к внешней памяти программ и доступ к внешней памяти данных. При доступе к ВПП в качестве сигнала чтения используется сигнал $\overline{\text{PSEN}}$ (Program Store Enable). Для доступа к ВПД используются сигналы чтения $\overline{\text{RD}}$ и записи $\overline{\text{WR}}$.

4.6.1 Доступ к внешней памяти программ

Доступ к ВПП возможен в двух случаях:

1) если сигнал $\overline{\text{EA}}$ (External Access) низкий, что соответствует отключению РПП;

2) если сигнал $\overline{\text{EA}}$ высокий, а счетчик команд (PC) содержит адрес больше, чем 0FFFH (4К – объем РПП).

При выборке команд из ВПП всегда используется 16-битный адрес. В этом случае старший байт адреса (PCH OUT) выводится в порт P2 и удерживается в течение полного цикла чтения команды. Младший байт адреса (PCL OUT) выводится через порт P0 и должен быть по срезу сигнала ALE записан во внешнем регистре, поскольку через порт P0 считывается байт команды. При наличии ВПП пользователь не должен производить запись информации в порт P0.

Диаграмма сигналов при доступе к внешней памяти программ (рис. 4.7) привязана к тактовым сигналам синхронизации. Для фиксации младшего байта адреса во внешнем регистре необходимо использовать отрицательный перепад сигнала ALE (Address Latch Enable). Сигнал ALE дважды в каждом машинном цикле принимает значение 1, даже если в цикле нет обращения к ВПП. Фиксация байта команды КОП, поступающего из ВПП через порт P0, происходит по положительному фронту сигнала $\overline{\text{PSEN}}$.

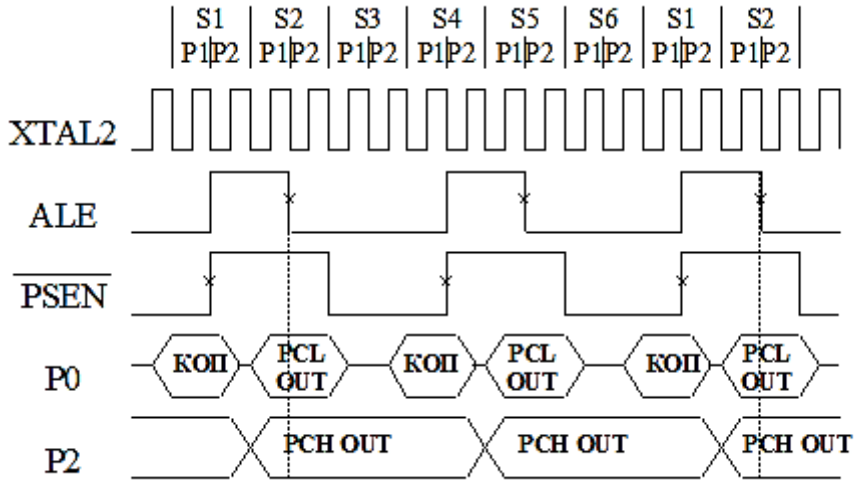

Рисунок 4.7 – Цикл доступа к внешней памяти программ

4.6.2 Доступ к внешней памяти данных

Доступ к внешней памяти данных возможен как с использованием 16-битового адреса, так и 8-битового адреса. При 16-битовом адресе порты P0 и P2 используются аналогично обращению к ВПП. При использовании 8-битового адреса содержимое порта P2 в цикле обращения к внешней памяти не изменяется. Обращение к ВПД возможно при отсутствии сигнала ALE. Поэтому первый сигнал ALE во втором машинном цикле команды обращения к ВПД подавляется. В цикле чтения ВПД (рис. 4.8) формируется сигнал $\overline{RD}$, связанный с альтернативной функцией вывода P3.7.

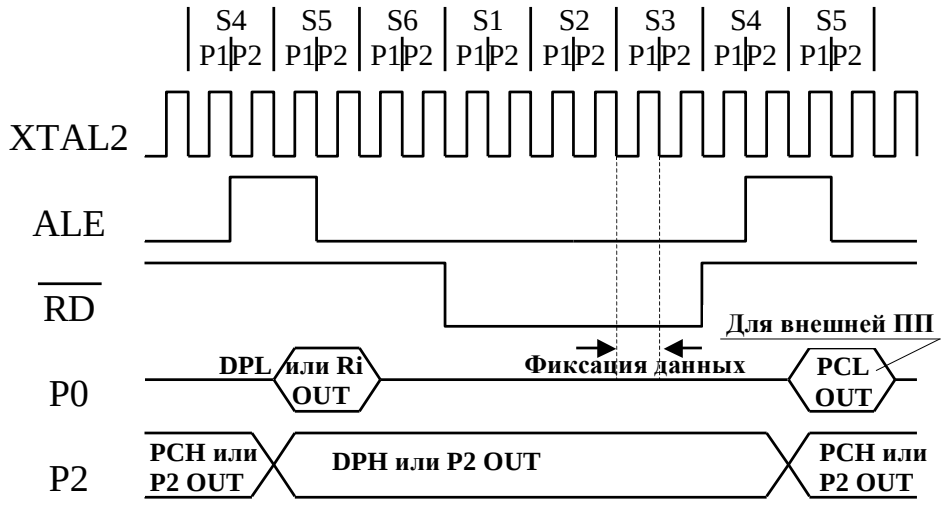

Рисунок 4.8 – Цикл чтения внешней памяти данных

Через порт P0 по спаду сигнала ALE передается 8-битовый адрес (Ri OUT) (команда типа MOVX @Ri) или младший байт DPTR (DPL OUT) (команда типа MOVX @DPTR). Фиксация данных портом P0 происходит до завершения сигнала $\overline{RD}$.

Для записи данных в ВПД (рис. 4.9) используется сигнал $\overline{WR}$.

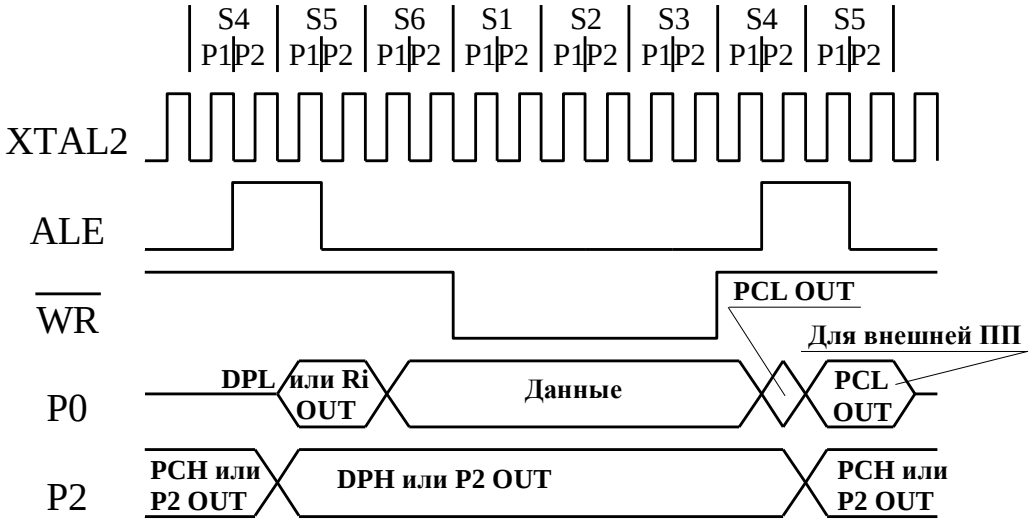

Рисунок 4.9 – Цикл записи во внешнюю память данных

4.6.3 Совмещение адресов ВПП и ВПД

Часто при разработке и отладке МП систем требуется объединить адресное пространство программ и данных. Это можно сделать аппаратно, объединив с помощью логического элемента «И» сигнал чтения памяти данных $\overline{RD}$ и сигнал чтения памяти программ $\overline{PSEN}$ (рис. 4.10).

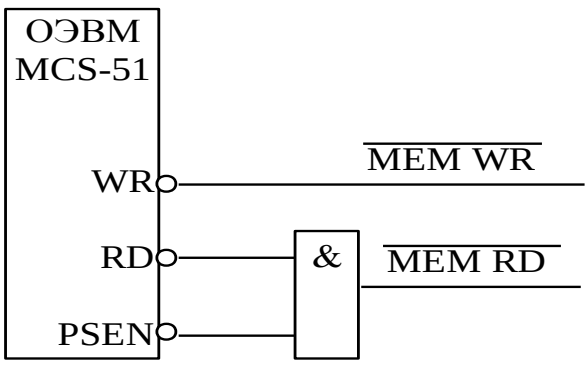

Рисунок 4.10 – Схема совмещения адресного пространства ВПП и ВПД

В этом случае внешняя память может быть выполнена в виде двухпортового ОЗУ, запись и чтение которого можно выполнять как из МК, так и из внешнего устройства. Запись программы также можно выполнять средствами МК как запись массива данных. Чаще всего такой режим используется при отладке МП системы с последующим разделением адресов программ и данных.

4.7 Таймеры/счетчики

В MCS-51 имеются два независимых 16-разрядных таймера/счетчика (T/C0 и T/C1). Каждый из них может работать в режиме таймера или счетчика внешних событий. В режиме таймера содержимое T/C увеличивается на единицу (инкрементируется) в каждом машинном цикле, т.е. через каждые 12 периодов резонатора.

В режиме счета внешних событий счетчик T/C инкрементируется по каждому отрицательному переходу (из «1» в «0») на соответствующем входе T0 или T1(альтернативные функции разрядов P3.4 и P3.5 порта P3). На распознавание перехода требуется два машинных цикла, поэтому максимальная частота счета внешних событий равна $F_{OSC}/24$. На длительность сигнала ограничений нет, но для гарантированного срабатывания счетчика необходимо обеспечить значение «1» в течение как минимум одного машинного цикла MCS-51.

Для задания режимов работы обоих таймеров/счетчиков имеется специальный регистр TMOD (табл. 4.5). Старшие четыре бита регистра управляют режимами $C/\overline{T}1$, а младшие – $C/\overline{T}0$. Битами M0 и M1 выбирается режим работы соответствующего таймера/счетчика, бит $C/\overline{T}$ отвечает за выбор таймера или счетчика.

Каждый таймер может независимо друг от друга работать в одном из 4 режимов.

Режим 0. В этом режиме оба таймера работают как 13-битовые делители, причем старший регистр TH0 (TH1) имеет 8 бит, а младший TL0 (TL1) – 5 младших бит. Старшие 3 бита регистра TLx не определены и игнорируются. Перед началом работы таймера/счетчика в регистры THx и TLx можно записать некоторое предварительное значение, с которого начнет увеличивать свое значение таймер.

Таблица 4.5 – Регистр управления режимами таймера/счетчика

Регистр режима таймера/счетчика TMOD Адрес 089H Не имеет битовой адресации		
Назва-ние бита	Адрес бита	Имя и назначение битов
GATE1	TMOD.7	Управление блокировкой. При GATE1=1 для разрешения работы таймера/счетчика 1 необходимо сочетание INT1=1 & TR1=1. При GATE1=0 для разрешения таймера/счетчика 1 достаточно TR1=1
$C/\overline{T}1$	TMOD.6	Выбор таймера (C/T1=0) или счетчика (C/T1=1)
$M1_1$ $M0_1$	TMOD.5 TMOD.4	$M1_1$ $M0_1$ Режим работы таймера/счетчика 0 0 13 бит. делитель, TH1→8 бит, TL1→5 бит 0 1 16 бит. делитель, TH1→8 бит, TL1→8 бит 1 0 8 бит. автозагружаемый делитель 1 1 Останов таймера/счетчика 1
GATE0	TMOD.3	Управление блокировкой. При GATE0=1 для разрешения работы таймера/счетчика 0 необходимо сочетание INT0=1 & TR0=1. При GATE0=0 для разрешения таймера/счетчика 0 достаточно TR0=1
$C/\overline{T}0$	TMOD.2	Выбор таймера (C/T0=0) или счетчика (C/T0=1)
$M1_0$ $M0_0$	TMOD.1 TMOD.0	$M1_0$ $M0_0$ Режим работы таймера/счетчика 0 0 13 бит. делитель, TH0→8 бит, TL0→5 бит 0 1 16 бит. делитель, TH0→8 бит, TL0→8 бит 1 0 8 бит. автозагружаемый делитель 1 1 TL0–8-битный таймер/счетчик 0, TH0 – 8 битный независимый таймер 1

Когда содержимое таймера/счетчика перейдет из состояния «все единицы» в состояние «все нули», устанавливается флаг прерывания TF0 (TF1). Если разрешена реакция на прерывание от данного таймера, MCS-51 вызывает подпрограмму обслуживания прерывания соответствующего таймера. Если реакция на прерывание запрещена, МК может программно опросить состояние флага прерывания и зафиксировать момент переполнения таймера.

Биты управления TR0 (TR1) и TF0 (TF1) находятся в регистре управления/статуса таймера TCON (табл. 4.6). Кроме этого, в регистре имеются биты, связанные с внешними прерываниями.

Таблица 4.6 – Регистр управления/статуса таймера

Регистр управления/статуса таймера TCON Адрес 088h Имеет битовую адресацию			
Название бита	Адрес бита	Позиция бита	Имя и назначение
TF1	08FH	TCON.7	Флаг переполнения таймера 1. Установка аппаратно при переполнении таймера/счетчика 1. Сброс аппаратно при обслуживании прерывания
TR1	08EH	TCON.6	Бит управления таймера 1. Установка /сброс программно для запуска/останова таймера/счетчика 1
TF0	08DH	TCON.5	Флаг переполнения таймера 0. Установка аппаратно при переполнении таймера/счетчика 0. Сброс аппаратно при обслуживании прерывания
TR0	08CH	TCON.4	Бит управления таймера 0. Установка /сброс программно для запуска/останова таймера/счетчика 0
IE1	08BH	TCON.3	Флаг фронта прерывания 1. Установка аппаратно по активному $\overline{\text{INT1}}$. Сброс програм./аппаратно
IT1	08AH	TCON.2	Бит типа прерывания 1. Устан./сброс прогр. для типа запроса прерывания 1 (срез/низкий уровень)
IE0	089H	TCON.1	Флаг фронта прерывания 0. Установка аппаратно по активному $\overline{\text{INT0}}$. Сброс програм./аппаратно
IT0	088H	TCON.0	Бит типа прерывания 0. Устан./сброс прогр. для типа запроса прерывания 0 (срез/низкий уровень)

Схема функционирования таймера/счетчика в режиме 0 (на примере T/C1, рисунок 4.11) показывает, как воздействуют биты управления на работу таймера.

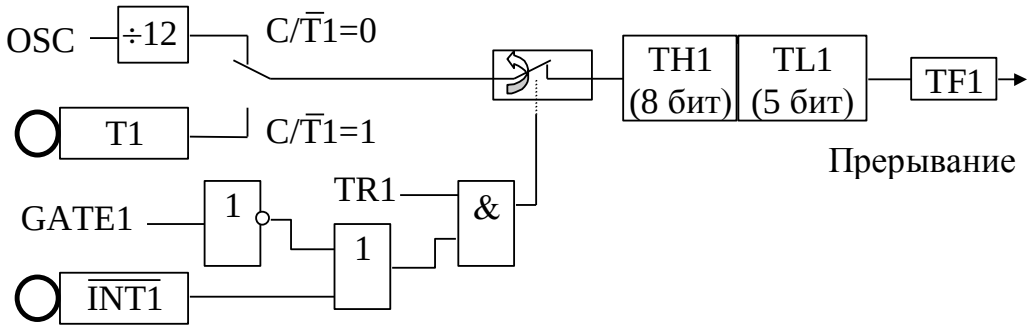

Рисунок 4.11 – Таймер/счетчик в режиме 0

Счет таймеру разрешается при установке бита TR0 (TR1) и запрещается при его сбросе, при условии, что соответствующий бит GATEx (TMOD.3 или TMOD.7) задан равным «0». Этот метод управления запуском/остановом таймера/счетчика можно назвать программным запуском и остановом.

При задании бита GATE0 (GATE1) равным «1» разрешение счета для таймера/счетчика определяется при значении «1» бита TR0 (TR1) и высокого уровня на входе $\overline{\text{INT0}}$ ($\overline{\text{INT1}}$). Такой метод управления запуском/остановом можно назвать аппаратным запуском и остановом. Если на вход $\overline{\text{INT0}}$ ($\overline{\text{INT1}}$) подавать импульсный сигнал, то можно по значению таймера измерить длительность вершины импульса, определяемое количеством тактов, изменяющих значение таймера.

Режим 1. Этот режим отличается от режима 0 только тем, что оба таймера имеют 16-битовую разрядность (рис. 4.12).

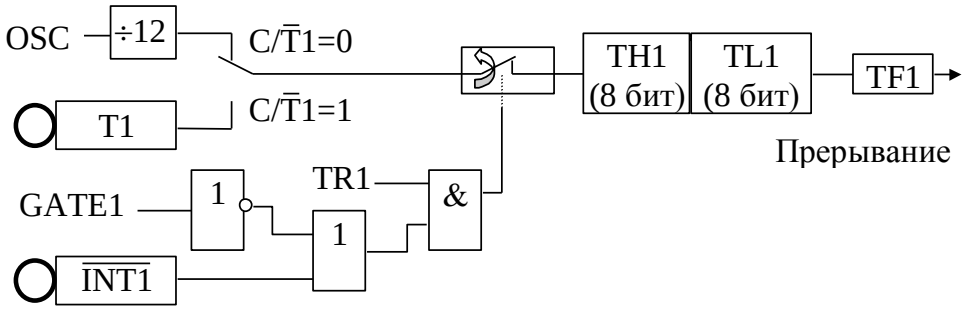

Рисунок 4.12 – Таймер/счетчик в режиме 1

Режим 2. В этом режиме таймер имеет разрядность 8 бит и использует для счета младший байт TL0 (TL1). В старший байт таймера TH0 (TH1) программно заносится коэффициент. При переполнении младшего счетчика не только устанавливается флаг TF0 (TF1), но и со-

держимое TH0 (TH1) перегружается в TL0 (TL1) (рис. 4.13). Перезагрузка не изменяет содержимого TH0 (TH1).

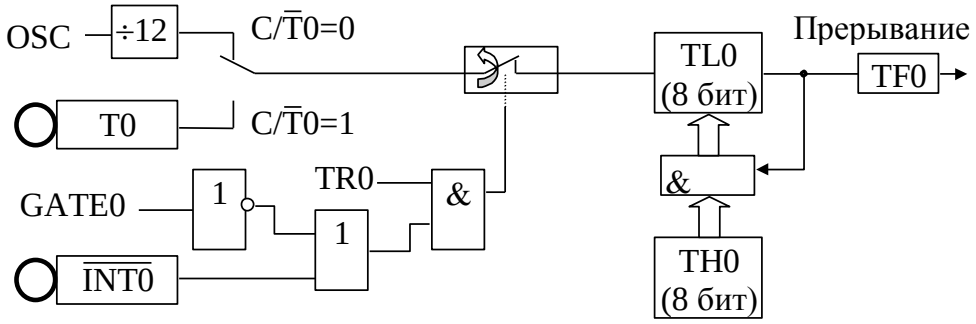

Рисунок 4.13 – Таймер/счетчик в режиме 2

За счет программно задаваемого коэффициента деления таймера этот режим очень удобен при управлении повторяющимися процессами, например, в программно перестраиваемом генераторе.

Оба таймера в этом режиме работают аналогично.

Режим 3. В этом режиме T/C1 просто хранит свое значение. T/C0 представляет собой два раздельных 8-битных счетчика (TH0 и TL0). Регистр TL0 управляется битами управления для таймера 0: GATE0, $C/\overline{T}0$, TR0, TF0. Регистр TH0 работает только в режиме таймера и использует биты таймера 1: TR1, TF1 (рис. 4.14).

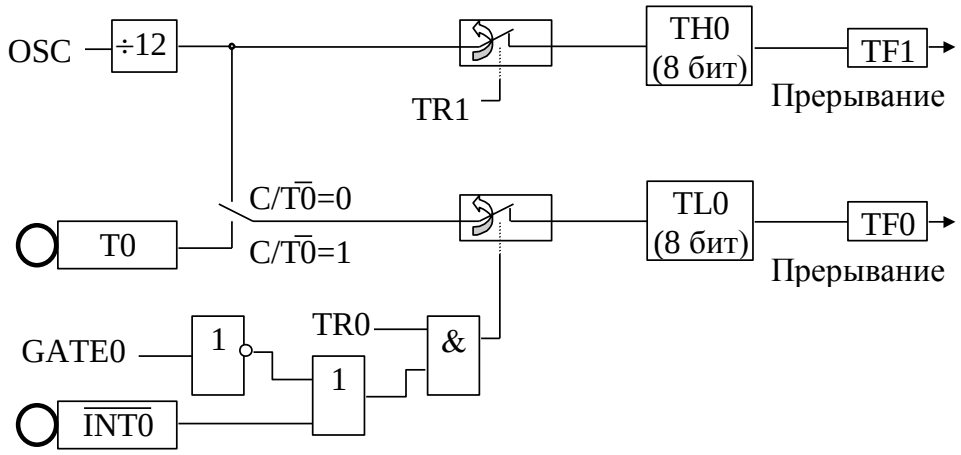

Рисунок 4.14 – Таймер/счетчик в режиме 3

Режим 3 применяется в тех приложениях, где требуется дополнительный 8-битный таймер или счетчик событий. Когда таймер 0 работает в режиме 3, таймер 1 может быть разрешен, запрещен, переведен в

свой собственный режим 3, использован последовательным портом для управления скоростью передачи или в любых других приложениях, не требующих прерывания.

4.8 Последовательный порт

Последовательный порт в MCS-51 – это универсальный асинхронный приемопередатчик (УАПП), через который выполняется прием и передача информации, представленной последовательным кодом. УАПП является полным дуплексным портом, что означает возможность осуществлять передачу и прием одновременно. В состав УАПП входят сдвигающие регистры приема и передачи и буферный регистр приемника/передатчика SBUF. На самом деле регистр SBUF состоит из двух раздельных регистров: буферного регистра приемника и буферного регистра передатчика.

Буферный регистр приемника организован таким образом, что прием нового байта можно начинать до считывания программой предыдущего принятого байта. Если же к моменту окончания приема текущего байта предыдущий байт не будет считан, он будет потерян.

Передача последовательного кода младшими разрядами вперед начинается из сдвигающего регистра передатчика сразу после записи байта в буферный регистр передатчика SBUF.

Для управления режимами и контроля состояниями используется регистр SCON (табл. 4.7).

Этот регистр содержит биты выборки режима (SM0 и SM1), бит SM2, предназначенный для мультипроцессорных режимов, бит разрешения приема (REN), 9-й бит данных для 9-битных режимов передачи и приема (TB8 и TR8) и биты прерывания последовательного порта (TI и RI).

Таблица 4.7 – Регистр управления SCON

<table>
<tr><td colspan="4">Регистр управления последовательным портом SCON Адрес 098h
Имеет битовую адресацию</td></tr>
<tr><td>Название бита</td><td>Адрес бита</td><td>Позиция бита</td><td>Имя и назначение</td></tr>
<tr><td>SM0
SM1</td><td>09FH
09EH</td><td>SCON.7
SCON.6</td><td>Режим УАПП. Сброс/установка программно
<table>
<tr><th>SM0</th><th>SM1</th><th>Ре-жим</th><th>Описа-ние</th><th>Скорость</th></tr>
<tr><td>0</td><td>0</td><td>0</td><td>Регистр сдвига</td><td>$F_{OSC}/12$</td></tr>
<tr><td>0</td><td>1</td><td>1</td><td>УАПП-8 бит</td><td>переменная</td></tr>
<tr><td>1</td><td>0</td><td>2</td><td>УАПП-9 бит</td><td>$F_{OSC}/64$ или $F_{OSC}/32$</td></tr>
<tr><td>1</td><td>1</td><td>3</td><td>УАПП-9 бит</td><td>переменная</td></tr>
</table></td></tr>
<tr><td>SM2</td><td>09DH</td><td>SCON.5</td><td>Бит управления режимом. При SM2=1 запрещается прием сообщений с нулевым девятым битом. Сброс/установка программно</td></tr>
<tr><td>REN</td><td>09CH</td><td>SCON.4</td><td>Бит разрешения приема. Установка /сброс программно для разрешения/запрета приема последовательных данных</td></tr>
<tr><td>TB8</td><td>09BH</td><td>SCON.3</td><td>9-тый бит данных для передачи в 2 и 3 режимах. Установка /сброс программно</td></tr>
<tr><td>RB8</td><td>09AH</td><td>SCON.2</td><td>9-тый принятый бит в режимах 2 и 3. Сброс аппаратно</td></tr>
<tr><td>TI</td><td>099H</td><td>SCON.1</td><td>Флаг прерывания передатчика. Установка аппаратно после передачи байта. Сброс программно</td></tr>
<tr><td>RI</td><td>098H</td><td>SCON.0</td><td>Флаг прерывания приемника. Установка аппаратно при приеме байта. Сброс программно</td></tr>
</table>

4.9 Режимы работы УАПП

4.9.1 Форматы режимов работы

Последовательный порт может работать в 4 режимах.

Режим 0. В этом режиме информация по 8 бит передается и принимается через внешний вывод входа приемника (RXD, порт P3.0). Че-

рез внешний вывод передатчика (TXD, порт P3.1) выдаются импульсы сдвига с фиксированной частотой $F_{OSC}/12$, которые синхронизируют каждый бит данных.

Передача начинается сразу после записи байта в SBUF. Прием начинается при условии (RI=0 & REN=1).

Режим 1. В этом режиме через TXD передаются и через RXD принимаются 10 бит: нулевой старт-бит, 8 бит данных, представленных младшими битами вперед и единичный стоп-бит (рис. 4.15).

Рисунок 4.15 – Кадр данных в режиме 1

Скорость передачи в этом режиме переменная. Для ее задания нужно использовать таймер T/C1. Прием данных начинается с приходом старт-бита при условии, что бит REN=1 (SCON.4).

Режим 2. В этом режиме через TXD передаются или через RXD принимаются 11 бит: нулевой старт-бит, 8 бит данных, программируемый 9-тый бит и единичный стоп-бит (рис. 4.16).

Рисунок 4.16 – Кадр данных в режиме 2

При передаче 9-тый бит данных считывается из бита TB8 (SCON.3), который можно изменять программно, например, копируя в него бит четности P (PSW.0). При приеме 9-тый принятый бит записывается в бит RB8 (SCON.2) с возможностью его дальнейшего программного анализа.

Скорость передачи фиксирована и выбирается из двух возможных вариантов: $F_{OSC}/32$ или $F_{OSC}/64$.

Режим 3. Режим соответствует режиму 2, но имеет переменную скорость передачи, определяемую частотой переполнения таймера/счетчика 1.

4.9.2 Скорость передачи

Сводка режимов (табл. 4.8) позволяет выбрать режим передачи с учетом требуемой скорости обмена информацией.

Таблица 4.8 – Скорость передачи в УАПП

Режим УАПП	Скорость передачи
0	$Fosc/12$
1	$2^{SMOD} \times F_{OVT1}/32$, F_{OVT1} – частота переполнения таймера 1
2	$2^{SMOD} \times Fosc/64$
3	$2^{SMOD} \times F_{OVT1}/32$, F_{OVT1} – частота переполнения таймера 1
1 или 3. Т/С1 в режиме с перезагрузкой	$2^{SMOD} \times \frac{F_{OSC}}{32 \times 12 \times [256 - (TH1)]}$

Стандартные скорости передачи (табл. 4.9) получаются с использованием таймера 1. В этом случае необходимо запретить прерывание TF1 (TCON.7). Таймер может работать либо как таймер, либо как счетчик событий в любом из трех режимов. В большинстве применений он используется как таймер в режиме 2 с автоперезагрузкой.

Таблица 4.9 – Типовые скорости передачи, задаваемые таймером 1

Режим УАПП	Скорость передачи, бит/с	F_{OSC}, МГц	SMOD	Таймер/счетчик 1		
				Бит С/Т1	Режим	Число
0	1 М	12	X	X	X	X
2	375 К	12	1	X	X	X
1,3	62,5 К	12	1	0	2	0FFH
–»–	19,2 К	11,059	1	0	2	0FDH
–»–	9,6 К	11,059	0	0	2	0FDH
–»–	4,8 К	11,059	0	0	2	0FAH
–»–	2,4 К	11,059	0	0	2	0F4H
–»–	1,2 К	11,059	0	0	2	0E8H
–»–	137,5	11,059	0	0	2	01DH
–»–	110	6	0	0	2	072H
–»–	110	12	0	0	1	0FEEBH

Скорость передачи в режимах 1, 2 или 3 дополнительно зависит от значения бита SMOD, расположенного в регистре управления энергопотреблением PCON (таблица 4.10). Кроме этого, в регистре имеются два бита, определяемые пользователем GF0 и GF1, бит сна PD и бит холостого хода IDL.

Таблица 4.10 – Регистр PCON

Регистр управления энергопотреблением PCON Адрес 087H Не имеет битовой адресации		
Название бита	Адрес бита	Имя и назначение битов
SMOD	PCON.7	Удвоенная скорость передачи. В режимах 1, 2 и 3 работы УАПП при SMOD=1 скорость передачи вдвое выше, чем при SMOD=0.
–	PCON.6	Не используются
–	PCON.5	Не используются
–	PCON.4	Не используются
GF1	PCON.3	Флаг пользователя GF1. Установка/сброс программно по желанию.
GF0	PCON.2	Флаг пользователя GF0. Установка/сброс программно по желанию.
PD	PCON.1	Бит сна. При PD =1 ОЭВМ работает в режиме минимальной мощности потребления (режим сна).
IDL	PCON.0	Бит холостого хода. При IDL=1 ОЭВМ работает в режиме ожидания (режим холостого хода).

4.10 Режимы пониженного энергопотребления

Для применений, где потребление энергии критично, например, при питании от батарей, MCS-51 имеет два режима пониженного энергопотребления: режим ожидания Idle (холостого хода) и режим сна PD (Power Down). Для активизации этих режимов требуется установить соответствующие биты PCON.0 (IDL) или PCON.1 (PD) (табл. 4.10).

Режим холостого хода. При поступлении команды, которая установит бит IDL, МК переходит в режим ожидания (холостого хода). В контроллере работает внутренний тактовый генератор, работает схема прерываний, таймеры и последовательный порт. Останавливается процесс выборки команд, сохраняются данные в регистрах и на разрядах портов.

Для выхода из режима ожидания можно воспользоваться одним из двух способов:

a) аппаратный сброс. Используется сигнал сброса $\overline{\text{RESET}}$. При этом переопределяются многие регистры блока специальных функций, но сохраняется содержимое РПД;

b) прерывание. Активизация любого разрешенного прерывания приведет к сбросу бита IDL, вызову программы обслуживания прерывания и возврату в основной программе к команде, следующей за командой, которая включила режим ожидания.

Режим сна. При выполнении команды, которая установит бит PD, МК перейдет в режим сна (низкого потребления) Power Down. В этом режиме выключается работа тактового генератора, все функции МК замораживаются, но РПД и блок регистров специальных функций сохраняют свои значения. Напряжение питания схемы при этом можно понизить до +2 В. Важно не понижать напряжение до самого момента включения режима Power Down, иначе может произойти аппаратный сброс.

Для выхода из данного режима можно использовать аппаратный сброс, или разрешенное прерывание от внешнего источника. Перед выходом из режима необходимо восстановить нормальный уровень питания и удержать его не менее 10^{-3} с. В этом случае успеет запуститься тактовый генератор, и его частота станет стабильной к моменту активизации МК.

4.11 Система прерываний

4.11.1 Источники прерываний

MCS-51 имеет пять источников прерываний (внешние прерывания $\overline{\text{INT0}}$, $\overline{\text{INT1}}$, прерывания от таймеров T0, T1 и прерывания от последовательного порта).

Внешние прерывания. Прерывания $\overline{\text{INT0}}$ и $\overline{\text{INT1}}$ могут быть вызваны либо уровнем, либо отрицательным перепадом на одноименных входах МК. Выбор определяется состоянием программно изменяемых бит IT0 и IT1 регистра TCON (TCON.0, TCON.2). При значении соответствующего бита «1», данное внешнее прерывание вызывается отрицательным перепадом.

Для генерации внешних прерываний используются флаги IE0 и IE1, находящиеся в регистре TCON (TCON.1, TCON.3). Если прерывание было вызвано отрицательным перепадом, то биты прерывания

сбрасываются при вызове программы обслуживания прерываний. Если прерывание вызвано низким уровнем внешнего сигнала, то сброс флагов прерывания связан со снятием активного уровня запроса прерывания.

При формировании запроса прерывания по перепаду высокий и низкий уровни входных сигналов должны удерживаться не менее одного машинного цикла.

Если внешние прерывания $\overline{\text{INT0}}$ или $\overline{\text{INT1}}$ вызываются низким уровнем сигнала, внешний источник должен удерживать активный уровень до вызова программы обслуживания прерывания. Запрос необходимо снять до завершения программы прерывания.

Прерывания таймеров. Прерывания вызываются флагами TF0 и TF1 регистра TCON (TCON.5, TCON.7). Флаги устанавливаются при переполнении соответствующего таймера/счетчика. Флаги сбрасываются аппаратно при вызове программы прерывания.

Прерывания последовательного порта. Прерывание от последовательного порта вызывается при единичном значении одного из флагов RI или TI регистра SCON (SCON.0, SCON.1). Значения флагов проверяются по логическому условию ИЛИ. Программа обслуживания прерывания от последовательного порта должна сама определить флаг, вызвавший прерывание, и сбросить его программно.

Все указанные флаги прерываний программно доступны, поэтому программа пользователя может вызвать любое прерывание и сбросить его флаг аналогично аппаратной реакции и с тем же результатом. Таким способом можно отлаживать соответствующие программы обслуживания прерываний или выполнять их для нужд решаемой задачи.

Адреса векторов прерываний (таблица 4.11) имеют фиксированные адреса, по которым должны находиться команды безусловного перехода на начало соответствующей программы обслуживания прерывания.

Таблица 4.11 – Адреса векторов прерываний

Источник прерывания	Бит запроса прерывания	Сбрасывается аппаратно	Адрес вектора
$\overline{\text{INT0}}$	IE0	Нет (уровень) Да (переход)	0003H
Таймер 0	TF0	Да	000BH
$\overline{\text{INT1}}$	IE1	Нет (уровень) Да (переход)	0013H

Источник прерывания	Бит запроса прерывания	Сбрасывается аппаратно	Адрес вектора
Таймер 1	TF1	Да	001BH
Последовательный порт	RI, TI	Нет	0023H

В системе прерываний MCS-51 имеется возможность индивидуально разрешать и запрещать реакцию системы на любое прерывание. Кроме этого, имеется возможность группового запрета всех прерываний, независимо от индивидуального разрешения/запрета.

Для управления процессом разрешения и запрета используется специальный регистр масок прерываний IE (таблица 4.12).

Таблица 4.12 – Регистр IE

Регистр масок прерываний IE Адрес 0A8h Имеет битовую адресацию			
Название бита	Адрес бита	Позиция бита	Имя и назначение
EA	0AFH	IE.7	Бит общего запрета. При EA=0 все прерывания запрещены. При EA=1 каждое прерывание разрешается/запрещается индивидуально
–	0AEH	IE.6	Не используется
–	0ADH	IE.5	Не используется
ES	0ACH	IE.4	Бит разрешения прерывания от УАПП. Установка /сброс программно для разрешения/запрета прерываний от флагов TI или RI
ET1	0ABH	IE.3	Бит разрешения прерывания от таймера 1. Установка/сброс программно для разрешения/запрета прерываний от таймера 1
EX1	0AAH	IE.2	Бит разрешения внешнего прерывания $\overline{\text{INT1}}$. Уст./сброс программно для разрешения/запрета прерываний от контакта $\overline{\text{INT1}}$
ET0	0A9H	IE.1	Бит разрешения прерывания от таймера 0. Установка/сброс программно для разрешения/запрета прерываний от таймера 0
EX0	0A8H	IE.0	Бит разрешения внешнего прерывания $\overline{\text{INT0}}$. Уст./сброс программно для разрешения/запрета прерываний от контакта $\overline{\text{INT0}}$

Для разрешения некоторого прерывания требуется отменить запрет всех прерываний (EA=1) и установить нужный бит. Если сбросить бит (EA=0), все прерывания запрещаются, независимо от индивидуального разрешения.

Способ группового запрета по биту EA позволяет быстро и просто в заданное время сделать систему нечувствительной к запросам прерываний (например, для выполнения важной процедуры, имеющей критическое время исполнения).

4.11.2 Приоритеты прерываний

Есть возможность индивидуально управлять приоритетом источников прерываний при помощи регистра приоритетов IP (табл. 4.13).

Таблица 4.13 – Регистр IP

Регистр приоритетов прерываний IP Адрес 0B8h **Имеет битовую адресацию**			
Название бита	Адрес бита	Позиция бита	Имя и назначение
–	0BFH	IP.7	Не используется
–	0BEH	IP.6	Не используется
–	0BDH	IP.5	Не используется
PS	0BCH	IP.4	Бит приоритета УАПП. Установка/сброс программно для присвоения прерыванию от УАПП высшего/низшего приоритета
PT1	0BBH	IP.3	Бит приоритета таймера 1. Установка/ сброс программно для присвоения прерыванию от таймера 1 высшего/низшего приоритета
PX1	0BAH	IP.2	Бит приоритета внешнего прерывания 1. Уст./сброс програм. для назначения прерыванию от контакта $\overline{INT1}$ высшего/низшего приоритета
PT0	0B9H	IP.1	Бит приоритета таймера 0. Установка/ сброс программно для присваивания прерыванию от таймера 0 высшего/низшего приоритета
PX0	0B8H	IP.0	Бит приоритета внешнего прерывания 0. Уст./сброс програм. для назначения прерыванию от контакта $\overline{INT0}$ высшего/низшего приоритета

Каждому из прерываний можно индивидуально присвоить один из двух приоритетов. При этом сначала обслуживаются запросы от прерываний с высшим приоритетом, а затем с низшим. Это делает систему прерываний очень гибкой и позволяет ее легко перестраивать под нужды решаемой задачи.

Прерывание с низким приоритетом может быть прервано прерыванием с более высоким приоритетом. Прерывание с наивысшим приоритетом не может быть прервано никаким другим источником прерывания.

Когда два или более запросов с разными уровнями приоритетов принимаются одновременно, запрос высокого уровня приоритета обслуживается первым.

Если одновременно принимаются запросы с одинаковыми уровнями приоритета, то порядок обслуживания определяется внутренней последовательностью опроса. Если все прерывания имеют одинаковый приоритет, то обслуживание запросов предполагает следующий порядок: $\overline{\text{INT0}}$, T0, $\overline{\text{INT1}}$, T1, TI+RI. Данный порядок соблюдается внутри каждой группы запросов с одина-ковым приоритетом.

4.11.3 Процесс прерывания

Флаги прерываний опрашиваются в фазе S5P2 каждого машинного цикла. При обнаружении запроса прерывания, в следующем машинном цикле выполняется опрос по уровням приоритетов.

Если один из флагов прерывания равен «1», система аппаратно генерирует команду вызова подпрограммы по фиксированному адресу вектора прерывания. При этом в стек загружается адрес следующей команды, после текущей, а в программный счетчик грузится вектор прерываний (адрес программы обслуживания).

Вызов подпрограммы может быть блокирован одним из условий:

1. В данный момент обслуживается прерывание равного или более высокого уровня приоритета.

2. Текущий цикл опроса не является последним циклом выполняемой команды.

3. Выполняется любая команда записи в регистр масок IE или регистр приоритетов IP.

Условие 2 гарантирует, что выполняемая команда будет завершена полностью. Условие 3 гарантирует, что если выполняется любая ко-

манда записи в указанные регистры, то по крайней мере еще одна команда будет выполнена после нее до вызова любой программы обслуживания.

Цикл опроса повторяется в каждом машинном цикле. Каждый цикл опроса является новым, т.е. флаг прерывания, не дождавшийся обслуживания, не запоминается. Для возврата из подпрограммы обслуживания прерывания должна использоваться команда RETI, которая считывает из стека адрес возврата, записывает его в программный счетчик и информирует систему, что завершена программа обслуживания прерывания, что восстанавливает всю систему прерываний.

В случае использования для возврата из программы обслуживания команды RET система прерываний будет считать, что продолжается обслуживание данного прерывания.

4.12 Запись в память программ MCS-51

В резидентную память программ MCS-51 записывают объектные коды прикладной программы. Для программирования РПП необходимо понизить частоту кварцевого резонатора до 4 – 6 МГц.

Схема подключения информации в режиме программирования (рис. 4.17) предусматривает подачу младшего байта адреса записываемого байта РПП через порт Р1. Четыре старших бита адреса РПП поступают через разряды Р2.0 – Р2.3. Через порт Р0 подается записываемый байт команды.

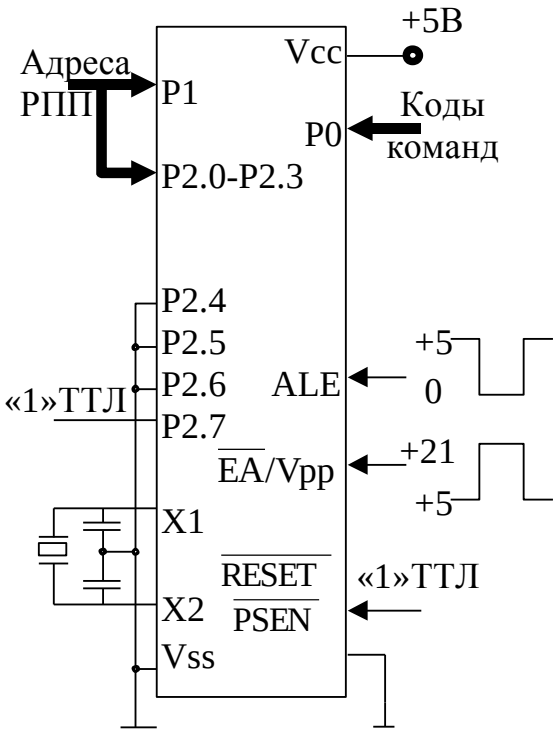

Рисунок 4.17 – Программирование MCS-51

Выводы P2.4 – P2.6 и $\overline{\text{PSEN}}$ должны быть заземлены. На выводы P2.7 и $\overline{\text{RESET}}$ необходимо подать уровень логической «1». На входы ALE и $\overline{\text{EA}}$/Vpp подается напряжение + 5 В.

В момент записи байта необходимо на выводе $\overline{\text{EA}}$/Vpp повысить напряжение до +21 Вольт. При этом на выводе ALE необходимо напряжение понизить до 0 В. Длительность сигналов должна превышать $50 \cdot 10^{-3}$ с. После истечения времени оба напряжения необходимо восстановить до исходного значения. После этого увеличивают адрес РПП, задают новый байт команды и повторяют процесс до завершения записи всей программы.

Для повышения достоверности записи в каждом цикле записи байта повторяют до 5 импульсов программирования.

В системе MCS-51 для защиты от несанкционированного копирования программ имеется возможность запретить чтение РПП любыми внешними средствами. Для этого необходимо установить бит защиты. Для его записи нужно в схеме (рис. 4.17) подать P2.6=1 и выполнить произвольный цикл записи.

Для сброса бита защиты потребуется полностью очистить память программ. Для стирания РПП микроконтроллер следует поместить под источник ультрафиолетового излучения с определенной длиной волны. После стирания в матрице РПП содержатся все единицы.

4.13 Система команд ОЭВМ КР1816ВЕ51

4.13.1 Форматы команд

Система команд MCS-51 содержит 111 базовых команд. По функциональному признаку команды можно разделить на пять групп:

a) команды передачи данных;

b) арифметические команды;

c) логические команды;

d) команды передачи управления;

e) команды операций с битами.

Большинство команд (94) имеют формат один или два байта и выполняются за один или два машинных цикла. Остальные команды – трехбайтные и выполняются за два машинных цикла. Только на выполнение команд умножения и деления требуется четыре машинных цикла.

Первый байт всех 13 типов команд (рис. 4.18) содержит код операции (КОП). Второй и третий байты содержат либо адреса операндов, либо непосредственные данные.

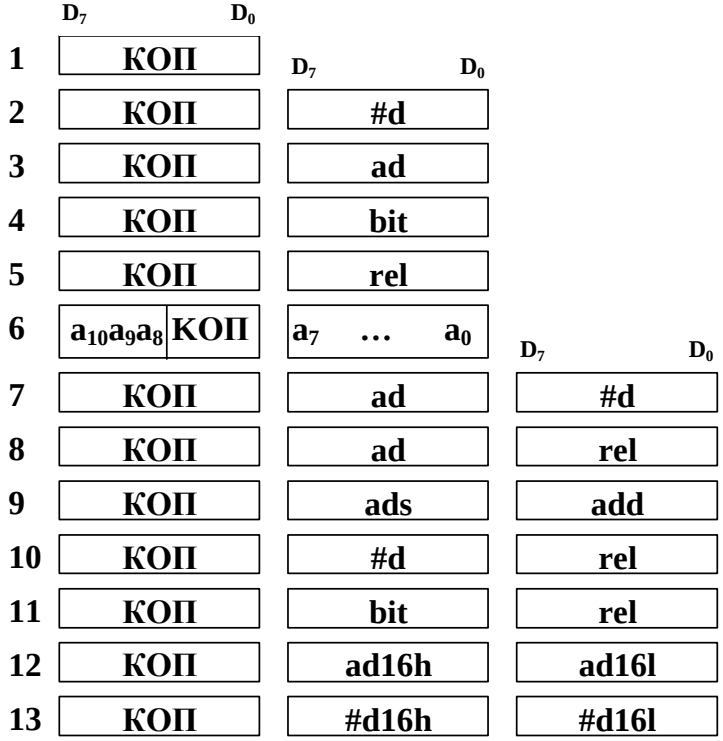

Рисунок 4.18 – Типы операндов в MCS-51

В системе команд MCS-51 используется четыре способа адресации данных.

1. Регистровая адресация - в команде указано символическое название регистра.

2. Прямая адресация - в команде явно указан адрес операнда в памяти.

3. Непосредственная адресация - в команде задан сам операнд.

4. Косвенная адресация - в команде задано имя регистра или регистровой пары, где содержится адрес, по которому и происходит обращение к операнду в памяти.

В системе команд используются следующие обозначения (в скобках показаны варианты обозначений, встречающиеся в командах, с буквой h обозначаются старшие байты слов, с буквой l – младшие байты слов):

Rn, n=0÷7 – регистры R0-R7 текущего банка регистров;

direct (ad, add, ads) – 8-битный прямой адрес ячейки резидентной памяти данных (0-127) или регистра специальных функций (128-255);

@Ri, i=0,1 – 8-битная ячейка РПД или регистра специальных функций, косвенно адресуемая через регистр R0 или R1;

#data (#d) – 8-битная константа, входящая в состав команды;

#data16 (#d16h, #d16l) – 16-битная константа, входящая в состав команды (непосредственная адресация);

addr16 (ad16h, ad16l) – 16-битовый адрес назначения в командах длинного перехода LJMP и вызова подпрограмм LCALL, позволяющий охватить полное адресное пространство в 64 Кбайт;

addr11 – 11-битовый адрес назначения в командах абсолютного перехода AJMP и вызова подпрограмм ACALL, позволяющий выполнить переход в пределах 2 Кбайт страницы от адреса текущей команды;

rel – 8-битное смещение со знаком. Смещение с учетом знака суммируется с текущим содержимым счетчика команд и позволяет выполнить переход в пределах –128 +127 байт относительно текущего адреса;

bit – 8-битный прямой адрес бита, находящегося в РПД (0-127) или в регистре специальных функций (128-255).

Микроконтроллер оперирует данными четырех типов: биты, тетрады (4-битные цифры), байты и 16-битные слова.

Биты. В MCS-51 определены 256 битов, из них 128 находятся в РПД и используются как флаги пользователя, а остальные 128 битов расположены в блоке регистров специальных функций и входят в со-

став некоторых регистров этого блока. Для обращения к битам используется прямой 8-битный адрес (bit).

Тетрады. Операции с тетрадами определены только в командах обмена тетрад в аккумуляторе и между регистром и аккумулятором.

Байты. Байтовым операндом может быть адрес внутренней и внешней памяти данных и программ, непосредственный операнд, константа, адрес регистра специальных функций или порта ввода/вывода.

Слова. Двухбайтовые операнды могут быть или константами, или прямыми адресами памяти и занимают второй и третий байты команды.

В слове состояния процессора PSW (табл. 4.3) имеются четыре флага, отражающие результат операции в АЛУ: C – перенос, AC – вспомогательный перенос, OV – переполнение и P – паритет. Только некоторые команды изменяют флаги результата (табл. 4.14).

Таблица 4.14 – Команды, изменяющие флаги результата

Команды	Флаги	Команды	Флаги	Команды	Флаги
ADD	C, OV, AC	DA	C	ANL C,b	C
ADDC	C, OV, AC	RRC	C	ANL C,/b	C
SUBB	C, OV, AC	RLC	C	ORL C,b	C
MUL	C=0, OV	SETB C	C=1	ORL C,/b	C
DIV	C=0, OV	CLR C	C=0	MOV C,b	C
		CPL C	C=$\overline{c}$	CJNE	C

Флаги C, AC, OV связаны с состоянием АЛУ и не изменяются в операциях, минующих использование АЛУ. Состояние этих флагов можно изменять и прямой записью соответствующих битов в регистре PSW. Флаг паритета P изменяется любой командой, изменяющей аккумулятор. Прямое изменение флага P не вызовет сообщения об ошибке, но его значение останется соответствующим состоянию аккумулятора.

4.13.2 Команды передачи данных

Команды передачи данных не влияют на флаги результата.

На графе возможных путей передачи данных в МК (рис 4.19) аккумулятор представлен отдельной вершиной, поскольку многие команды используют его при передаче данных .

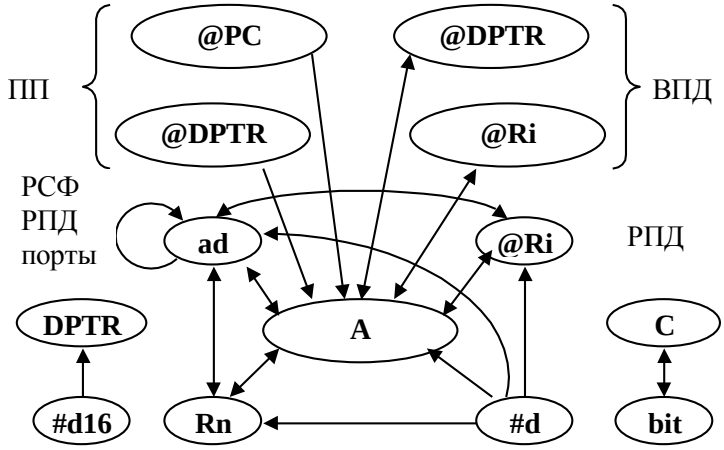

Рисунок 4.19 – Граф путей передачи данных в MCS-51

Большую часть команд данной группы составляют команды передачи и обмена байтами.

Столбец «Байты» (табл. 4.15) показывает количество байтов в команде, а «Циклы» – это количество машинных циклов на выполнение команды.

Таблица 4.15 – Команды передачи данных

Команда	Байты	Циклы	Описание команды
MOV A,#d	2	1	Загрузка в аккумулятор константы
MOV A,@Ri	1	1	Пересылка в аккумулятор байта из РПД (i=0,1)
MOV A,ad	2	1	Пересылка в аккумулятор байта с адреса ad
MOV A,Rn	1	1	Пересылка в аккумулятор регистра (n=0÷7)
Команды, пересылающие байт данных в аккумулятор с разными методами адресации, работают очень быстро			
MOV Rn,#d	2	1	Загрузка в регистр (n=0÷7) константы
MOV Rn,A	1	1	Пересылка в регистр (n=0÷7) аккумулятора
MOV Rn,ad	2	2	Пересылка в регистр (n=0÷7) байта, определяемого прямым 8-битным адресом
Пересылка байта данных в любой из 8 (R0-R7) регистров текущего банка с разными методами адресации			
MOV ad,#d	3	2	Пересылка по прямому адресу константы
MOV ad,@Ri	2	2	Пересылка по прямому адресу байта из РПД
MOV ad,A	2	1	Пересылка по прямому адресу аккумулятора
MOV ad,Rn	2	2	Пересылка по прямому адресу регистра (n=0÷7)
MOV add,ads	3	2	Пересылка прямоадресуемого байта ads по прямому адресу add

Команда	Байты	Циклы	Описание команды
Команды пересылки по прямому адресу байта, получаемого из различных источников, значительно повышают эффективность программ			
MOV @Ri,#d	2	1	Пересылка в РПД константы по адресу из R0 или R1 (i=0,1)
MOV @Ri,A	1	1	Пересылка в РПД байта из аккумулятора (i=0,1)
MOV @Ri,ad	2	2	Пересылка в РПД байта, взятого по адресу ad
Пересылка байта во внутренней памяти с использованием косвенного адреса, расположенного в регистре R0 или R1			
MOV DPTR,#d16	3	2	Загрузка указателя данных
Загрузка 16-битовым операндом регистра указателя данных			
MOVC A,@A+DPTR	1	2	Пересылка в аккумулятор байта из ПП (косв.адрес)
MOVC A,@A+PC	1	2	Пересылка в аккумулятор байта из ПП (косв.адрес)
Единственные команды для чтения байтов данных из памяти программ, например, констант, кодов инициализации			
MOVX @DPTR,A	1	2	Пересылка байта в ВПД из аккумулятора с 16-битовым адресом
MOVX A,@DPTR	1	2	Пересылка в аккумулятор байта из расширенной ВПД, адресуемого 16-битовым адресом
MOVX @Ri,A	1	2	Пересылка байта в ВПД из аккумулятора с 8-битовым адресом из R0 или R1 (i=0,1)
MOVX A,@Ri	1	2	Пересылка в аккумулятор байта из ВПД (i=0,1)
Команды с косвенной адресацией для чтения и записи байтов во внешней памяти данных с использованием 16-битового и 8-битового адресов			
PUSH ad	2	2	Загрузить в стек байт, взятый по адресу ad
POP ad	2	2	Извлечь из стека байт, и записать по адресу ad
Команды записи и чтения прямо адресуемого байта в область стека			
XCH A,@Ri	1	1	Обмен аккумулятора с байтом из РПД (i=0,1)
XCH A,ad	2	1	Обмен аккумулятора с байтом по адресу ad
XCH A,Rn	1	1	Обмен аккумулятора с регистром (n=0÷7)
Команды обмена аккумулятора и байта с различной адресацией			
XCHD A,@Ri	1	1	Обменять младшую тетраду аккумулятора с младшей тетрадой байта из РПД (i=0,1)
SWAP A	1	1	Обменять тетрады в аккумуляторе между собой
Команды обмена 4-битовых операндов используют аккумулятор			

4.13.3 Арифметические команды

В данную группу команд (табл. 4.16) входят команды сложения, вычитания, умножения и деления байтов. Имеются команды десятичной коррекции результата сложения, увеличения на единицу как 8-битовых, так и 16-битовых операндов, уменьшения на единицу для 8-битовых операндов.

Все команды этой группы используют АЛУ и влияют на состояние флагов.

Таблица 4.16 – Группа арифметических операций

Команда	Байты	Циклы	Описание команды
ADD A,#d	2	1	Сложить аккумулятор с константой
ADD A,@Ri	1	1	Сложить аккумулятор с байтом из РПД (i=0,1)
ADD A,ad	2	1	Сложить аккумулятор с байтом по адресу ad
ADD A,Rn	1	1	Сложить аккумулятор с регистром
Команды сложения используют аккумулятор как источник одного слагаемого и как приемник суммы. Другой байт берется из команды или из РПД с различными способами адресации			
ADDC A,#d	2	1	Сложить аккумулятор с константой и с переносом
ADDC A,@Ri	1	1	Сложить аккумулятор с байтом из РПД (i=0,1) и с переносом
ADDC A,ad	2	1	Сложить аккумулятор с байтом по адресу ad и с переносом
ADDC A,Rn	1	1	Сложить аккумулятор с регистром (n=0÷7) и с переносом
Аналогично предыдущей подгруппе, но к результату прибавляется бит переноса. Так организуются многобайтовые сложения			
DA A	1	1	Десятичная коррекция аккумулятора
Десятичная коррекция нужна при работе с двоично-десятичными числами			
SUBB A,#d	2	1	Вычесть из аккумулятора константу и заем
SUBB A,@Ri	1	1	Вычесть из аккумулятора байт из РПД (адрес в R0 или R1) и заем
SUBB A,ad	2	1	Вычесть из аккумулятора байт по адресу ad и заем
SUBB A,Rn	1	1	Вычесть из аккумулятора регистр (n=0÷7) и заем
Команды вычитают из результата бит заема, что позволяет выполнять многобайтовое вычитание, начиная с младших пар байтов			
INC @Ri	1	1	Инкремент байта в РПД по адресу в R0 или R1
INC A	1	1	Инкремент аккумулятора
INC ad	2	1	Инкремент байта по адр. Ad

Команда	Байты	Циклы	Описание команды
INC Rn	1	1	Инкремент регистра R0-R7, (n=0÷7)
INC DPTR	1	2	Инкремент указателя данных
Команды увеличения на единицу как 8-битных, так и 16-битных операндов (DPTR), расположенных в РПД, с разными способами адресации. Удобно использовать для организации последовательного доступа к элементам данных			
DEC @Ri	1	1	Декремент байта в РПД по адресу в R0 или R1
DEC A	1	1	Декремент аккумулятора
DEC ad	2	1	Декремент байта по адресу ad
DEC Rn	1	1	Декремент регистра R0-R7, (n=0÷7)
Команды уменьшения на единицу 8-битных операндов из РПД с разными способами адресации. Удобно использовать для организации последовательного доступа к элементам данных, в качестве счетчиков			
MUL AB	1	4	Умножение аккумулятора на регистр B
DIV AB	1	4	Деление аккумулятора на регистр B
Команда умножения использует для приема 16-битового результата умножения регистры B и A. Команда деления возвращает в аккумуляторе целую часть частного, в регистре B записан остаток от деления			

4.13.4 Логические команды

В данную группу (табл. **4.17**) входят команды, выполняющие логические операции вида «И», «ИЛИ», «Исключающее ИЛИ» над операндом в аккумуляторе и в любом байте РПД или в регистрах специальных функций. Сюда же входят команды сдвига аккумулятора влево и вправо, а также сброса и инверсии аккумулятора.

Все команды этой группы используют АЛУ и влияют на состояние флагов.

Таблица 4.17 – Группа логических операций

Команда	Байты	Циклы	Описание команды
ANL A,#d	2	1	Логическое И аккумулятора с 8-битовой константой
ANL A,@Ri	1	1	Логическое И аккумулятора с байтом из РПД (i=0,1), адрес которого записан в R0 или R1

Команда	Байты	Циклы	Описание команды
ANL A,ad	2	1	Логическое И аккумулятора с байтом по адресу ad
ANL A,Rn	1	1	Логическое И аккумулятора с регистром (n=0÷7)
ORL A,#d	2	1	Логическое ИЛИ аккумулятора с константой
ORL A,@Ri	1	1	Логическое ИЛИ аккумулятора с байтом из РПД (i=0,1)
ORL A,ad	2	1	Логическое ИЛИ аккумулятора с байтом по адресу ad
ORL A,Rn	1	1	Логическое ИЛИ аккумулятора с регистром (n=0÷7)
XRL A,#d	2	1	Исключающее ИЛИ аккумулятора с константой
XRL A,@Ri	1	1	Исключающее ИЛИ аккумулятора с байтом из РПД (i=0,1)
XRL A,ad	2	1	Исключающее ИЛИ аккумулятора с байтом по адресу ad
XRL A,Rn	1	1	Исключающее ИЛИ аккумулятора с регистром (n=0÷7)
При выполнении логических операций результат помещается в аккумулятор. Он же является источником первого операнда. Другой операнд находится либо в команде, либо в РПД с разными методами доступа			
ANL ad,#d	3	2	Логическое И байта по адресу ad с константой
ANL ad,A	2	1	Логическое И байта по адресу ad с аккумулятором
ORL ad,#d	3	2	Логическое ИЛИ байта по адресу ad с константой
ORL ad,A	2	1	Логическое ИЛИ байта по адресу ad с аккумулятором
XRL ad,#d	3	2	Исключающее ИЛИ байта по адресу ad с константой
XRL ad,A	2	1	Исключающее ИЛИ байта по адресу ad с аккумулятором
Команды позволяют любой байт из РПД и регистров специальных функций подвергнуть логической операции, используя в качестве маски константу или аккумулятор. Можно изменять значения портов, регистров управления и статуса таймеров, последовательного порта и т.д.			
CLR A	1	1	Сброс аккумулятора
CPL A	1	1	Инверсия аккумулятора
Короткие команды изменения содержимого аккумулятора			
RL A	1	1	Сдвиг аккумулятора влево по циклу
RR A	1	1	Сдвиг аккумулятора вправо по циклу
RLC A	1	1	Сдвиг аккумулятора влево через перенос
RRC A	1	1	Сдвиг аккумулятора вправо через перенос
Команды сдвига на разряд влево и вправо работают по циклу. Сдвиг через перенос включает в цикл флаг переноса и используется при арифметических сдвигах			

4.13.5 Команды операций с битами

Данные команды (табл. 4.18) оперируют с однобитовыми операндами. Биты расположены в области битовой адресации (128 битов пользователя) и в области регистров специальных функций. Для адресации бит используется прямой 8-битовый адрес (bit). Косвенная адресация бит невозможна.

Команды не затрагивают флагов, кроме флага переноса в отдельных случаях.

Таблица 4.18 – Битовые операции

Команда	Байты	Циклы	Описание команды
CLR C	1	1	Сброс переноса
SETB C	1	1	Установка переноса
CPL C	1	1	Инверсия переноса
Команды смены состояния флага переноса. Полезны для операций сдвига, команд вычитания и сложения с учетом переноса			
CLR bit	2	1	Сброс бита
SETB bit	2	1	Установка бита
CPL bit	2	1	Инверсия бита
Команды смены состояния произвольного бита из области битов пользователя и регистров специальных функций. Полезны для управления внутренними устройствами: портами, таймерами и т.д.			
MOV bit,C	2	1	Пересылка переноса в бит
MOV C,bit	2	1	Пересылка бита в перенос
Команды пересылки битов во флаг переноса и обратно увеличивают эффективность битовой обработки в MCS-51 и функций управления устройств			
ANL C,bit	2	2	Логическое И бита и переноса
ANL C,/bit	2	2	Логическое И инверсии бита и переноса
ORL C,bit	2	2	Логическое ИЛИ бита и переноса
ORL C,/bit	2	2	Логическое ИЛИ инверсии бита и переноса
Команды логических функций флага переноса и заданных битов расширяют систему битовых команд			

4.13.6 Команды передачи управления

Данная группа команд (табл. 4.19) содержит команды условных и безусловных переходов, вызовы подпрограмм и возврат из них, коман-

ды сравнения и цикла. Для выполнения условных переходов можно проверять разнообразные условия, а безусловные переходы и вызовы подпрограмм различаются по предельной удаленности перехода.

Таблица 4.19 – Группа команд передачи управления

Команда	Байты	Циклы	Описание команды
LJMP ad16	3	2	Длинный переход в пределах 64 Кбайт
AJMP ad11	2	2	Абсолютный переход в странице 2 Кбайт
JMP @A+DPTR	1	2	Косвенный относительный переход
Команды безусловного перехода загружают счетчик команд адресом, заданным в команде 16-битовым, 11-битовым числом или суммой байта аккумулятора с 16-битовым числом в регистре указателя данных			
SJMP rel	2	2	Короткий относительный переход
Команда безусловного перехода суммирует счетчик команд с учетом знака с байтом rel и выполняет переход в пределах –128 +127 байтов			
JZ rel	2	2	Переход, если аккумулятор равен нулю
JNZ rel	2	2	Переход, если аккумулятор не равен нулю
JC rel	2	2	Переход, если перенос равен единице
JNC rel	2	2	Переход, если перенос равен нулю
JB bit,rel	3	2	Переход, если бит равен единице
JNB bit,rel	3	2	Переход, если бит равен нулю
JBC bit,rel	3	2	Переход, если бит установлен, с последующим сбросом бита
Команды условного перехода проверяют заданное условие и выполняют, при необходимости, переход в пределах –128 +127 байтов. Последняя команда дополнительно сбрасывает проверяемый бит			
DJNZ ad,rel	3	2	Декремент байта по адресу ad и переход, если не нуль
DJNZ Rn,rel	2	2	Декремент регистра и переход, если не нуль
Команды условного перехода вычитают «1» из данного байта и выполняют переход в пределах –128 +127 байтов при ненулевом результате. Иначе перехода нет. Команды используются для организации циклов			
CJNE A,ad,rel	3	2	Сравнение аккумулятора с байтом по адресу ad и переход, если не равно
CJNE @Ri,#d,rel	3	2	Сравнение байта из РПД с константой и переход, если не равно
CJNE A,#d,rel	3	2	Сравнение аккумулятора с константой и переход, если не равно
CJNE Rn,#d,rel	3	2	Сравнение регистра с константой и переход, если не равно

Команда	Байты	Циклы	Описание команды
Команды сравнивают байт с константой или другим байтом. При неравенстве выполняют переход на –128 +127 байтов. Применяются при ожидании события, например, определенного значения порта. Команды изменяют флаг C в соответствии с результатом операции сравнения			
LCALL ad16	3	2	Длинный вызов подпрограммы в пределах 64 Кбайт
ACALL ad11	2	2	Абсолютный вызов подпрограммы в странице 2 Кбайта по 11-битовому адресу
RET	1	2	Возврат из подпрограммы
RETI	1	2	Возврат из подпрограммы обработки прерывания
Команды вызова подпрограмм по 16 и 11-битовому адресу записывают в стек адрес следующей команды и выполняют переход на данный адрес. По командам возврата в счетчик команд загружается адрес из стека. Команда RETI сообщает системе прерываний о завершении прерывания			
NOP	1	1	Нет операции
Пустая команда используется для резервирования места при отладке программ, формировании задержки и других целей			

Команды условных переходов используют состояние флагов C, bit и состояние аккумулятора.

В приложении А приведена система команд MCS-51, разделенная на функциональные группы и представленная в списочной форме с описанием команд.

4.14 Пример составления простейших программ для ОЭВМ КР1816ВЕ51

Составление простейших программ и фрагментов для решения типовых задач пользователя способствует лучшему усвоению методики программирования микропроцессорных систем на основе MCS-51. Программы сопровождаются подробными комментариями и содержат список начальных условий и результатов.

Использование команд передачи данных. Одной из типовых задач является инициализация некоторой области резидентной памяти данных, например, ее очистка. Примером такой задачи является фрагмент программы (табл. 4.20), выполняющей сброс всех 128 флагов пользователя, расположенных в резидентной памяти данных.

Таблица 4.20 – Фрагмент программы сброса флагов пользователя

Начальные условия: Флаги пользователя расположены в РПД в пределах адресов с 20H по 2FH **Результаты:** Сброшены все 128 флагов пользователя в области РПД			
Метка	**Команда**		**Комментарий**
	MOV	R0,#20H	; Начальный адрес области флагов
	MOV	R1,#0FH	; Счетчик длины области флагов
MET0:	MOV	@R0,#00	; Сброс очередных 8 флагов
	INC	R0	; Переход к следующему байту флагов
	DJNZ	R1,MET0	; Делаем цикл до обнуления счетчика R1
	…	…	; Следующие команды программы

Ввод информации с внешних датчиков. Одной из типовых для микропроцессорной системы является задача ввода поступающей с внешних источников информации.

Допустим, во внешнюю память данных с некоторого датчика порциями по 60 байтов записывается информация. После записи последнего байта, на P1.2 разряд порта P1 поступает положительный импульс длительностью около $10*10^{-6}$ с, что гарантирует его обнаружение в МК. Требуется обнаружить входной импульс и по его окончании, перенести массив из ВПД в РПД с очисткой исходной области во внешней памяти.

Для обнаружения импульса готовности данных в программе (табл. 4.21) сначала выполняется поиск положительного фронта импульса. Команда условного перехода опрашивает значение заданного бита и выполняет переход на саму себя. Другая условная команда предназначена для обнаружения спада импульса. Две условные команды последовательно позволят обнаружить начало и конец импульса управления.

Таблица 4.21 – Перенос массива из ВПД в РПД

Начальные условия: 1235H – начальный адрес массива в ВПД. 60 байтов – длина массива. 40H – начальный адрес приемника в РПД **Результаты:** Весь массив из ВПД перенесен в РПД. Массив в ВПД очищен			
Метка	**Команда**		**Комментарий**
	MOV	DPTR,#1235H	; Начальный адрес массива в ВПД
	MOV	R1,#40H	; Начальный адрес приемника в РПД
	MOV	R2,#60D	; Счетчик длины массива

```
MET1:  JNB    P1.2,MET1     ; Ожидаем фронт импульса
MET2:  JB     P1.2,MET2     ; Ожидаем спад импульса
MET3:  MOVX   A,@DPTR       ; Читаем текущий элемент массива
       MOV    @R1,A         ; Записываем его в РПД
       CLR    A             ; Очищаем аккумулятор
       MOVX   @DPTR,A       ; Чистим текущий элемент массива
       INC    DPTR          ; Идем к следующему элементу в ВПД
       INC    R1            ; Идем к следующему байту приемника
       DJNZ   R2,MET3       ; Делаем цикл до обнуления счетчика R2
         …        …         ; Следующие команды программы
```

Использование арифметических и битовых команд. Использование арифметических и битовых команд удобно показать на примере подпрограммы табличного вычисления функции sin(x). Угол должен быть в пределах 0 – 360º. Значение угла x в градусах задается старшей частью в регистре B, а младшей – в аккумуляторе.

В памяти программ записана таблица значений синуса от 0º до 89º в виде байта дробных значений. Дробные величины записываются в виде целочисленного значения, равного числителю дроби, а знаменатель дроби подразумевается и равен 256.

Значения синуса угла в 3 и 4 квадранте отличаются только знаком от значения угла 1 и 2 квадрантов. Значения из 2 квадранта соответствуют дополнительным значениям из 1 квадранта. Поэтому при преобразовании значения угла использовались следующие формулы:

$Y = \sin(x),$ при $0 \le x < 90$ 1 квадрант

$Y = \sin(180 - x),$ при $90 \le x < 180$ 2 квадрант

$Y = -\sin(x - 180),$ при $180 \le x < 270$ 3 квадрант

$Y = -\sin(360 - x),$ при $270 \le x < 360$ 4 квадрант

При вычислении функции (табл. 4.22) для определения значения угла, превышающего 180º, требуется сравнивать значения двухбайтовых чисел. Поскольку максимальное значение не превышает числа 360, можно хитро уменьшить программу. Если временно разделить значения угла пополам и сравнивать половинные значения углов, числа не будут превышать одного байта. Это упростит процедуру сравнения. Деление величины угла на 2 выполняется операцией сдвига право.

Таблица 4.22 – Табличное вычисление sin(x)

Начальные условия: Рег B – старшая часть значения угла, A – мл. часть значения угла в градусах. В памяти программ – таблица значений синуса в виде байта **Результаты:** A – байт дробной части значения синуса, рег. B = 0 для положительного результата, и B=0FFH – для отрицательного результата			
Метка	**Команда**		**Комментарий**
SINUS:	MOV	R7,A	; Копия младшей части значения угла
	MOV	C,B.0	; Определяем старший бит значения угла
	RRC	A	; Для работы делим угол пополам, с учетом C
	CJNE	A,#90,MET4	; Угол относится к 3 или 4 квадранту?
MET4:	MOV	A,R7	; Восстановим правильное значение угла
	JNC	POS	; Значение синуса положительно! Переход.
	CLR	C	; Иначе, для вычитания подготовимся
	SUBB	A,#180	; Корректируем значение угла на 180º
	MOV	B,#0FFH	; Значение синуса отрицательно
POS:	CJNE	A,#90,MET5	; Угол относится к 1 квадранту или 2?
MET5:	JNC	SIN	; если 1 квадрант, можно вычислять функцию
	CPL	A	; Для 2 квадранта нужно перенести
	INC	A	; значение угла в 1 квадрант
	ADD	A,#180	; Формируем (180 – угол)
SIN:	INC	A	; Коррекция для обращения к таблице
	MOVC	A,@A+PC	; Читаем значение функции из таблицы
	RET		; Возврат в вызывающую программу
T_SIN:	DB	00000000B	; Значение sin(0º) = 0.000
	DB	00000100B	; Значение sin(1º) = 0.017
	DB	00001001B	; Значение sin(2º) = 0.035
	DB	00001101B	; Значение sin(3º) = 0.052
	DB	00010010B	; Значение sin(4º) = 0.070
	…	…	; Все значения от sin(5º) до sin(88º)
	DB	11111111B	; Значение sin(89º) = 0.999

Поэтому старший бит значения угла из регистра B копируется в флаг переноса. Затем этот бит вдвигается в аккумулятор. Полученное половинное значение угла сравнивается с половинным значением от 180. Далее, при необходимости, вычитаем 180 для перехода в 1, 2 квадранты.

Аналогично происходит переход в 1 квадрант. Для 2-го квадранта считаем величину 180-x. Когда значение угла соответствует 1 квадранту, значение sin(x) берется из таблицы. Для положительного значения функции в регистре B будет 0 и 0FFH – для отрицательного значения sin(x).

Использование подпрограмм. Одной из типовых задач является процедура формирования программной задержки. Примером такой задачи является фрагмент программы (табл. 4.23), выполняющей задержку на 1 секунду. Во фрагменте задана подпрограмма DELAY01, выполняющая задержку примерно на 0,1 сек.

Таблица 4.23 – Фрагмент программы формирования задержки

Начальные условия: Подпрограмма DELAY01 дает задержку около 0,1 секунды. Она вызывается 10 раз подряд **Результаты:** Программа выполнит задержку на 1 секунду			
Метка	**Команда**		**Комментарий**
	MOV	R4,#10	; Количество вызовов задержки в 0,1 сек
SNOV:	LCALL	DELAY01	; Вызов задержки в 0,1 сек
	DJNZ	R4, SNOV	; Цикл на 10 повторений
	…	…	
	…	…	; Следующие команды программы
; Подпрограмма задержки на 0,1 секунды на основе двух счетчиков			
DELAY01:	MOV	R3,#216	; R3 – Счетчик внешнего цикла = 0D8H
HAP:	MOV	R2,#230	; R2 – Счетчик внутреннего цикла = 0E6H
BH:	DJNZ	R2,BH	; В цикле обнуляем внутренний счетчик R2
	DJNZ	R3,HAP	; Цикл до обнуления внешнего счетчика R3
	RET		; Возврат из подпрограммы

Коэффициенты для счетчиков подбирались исходя из требуемого числа машинных циклов. При тактовой частоте 12 МГц длительность машинного цикла равна $1 \cdot 10^{-6}$ с. С учетом длительности команд и повторения циклов получается около 100 тысяч машинных циклов. Это приближенно дает задержку в 0,1 секунды. За счет повторных вызовов подпрограммы задержка доводится до 1 секунды. Максимальная задержка в данном примере может достигать 25 секунд.

Использование таймера. Задачу формирования задержки можно решить с использованием таймера. В табл. 4.24 приведен фрагмент программы, выполняющей задержку на 1 секунду с помощью таймера. В 16-битовом режиме таймера, при тактовой частоте 12 МГц, максимальное время задержки составит $65536 * 10^{-6}$ сек (65,536 мс). Для удлинения общей задержки потребуются повторные запуски таймера. Для кратного счета удобнее обеспечить задержку в 50 мс (0,05 сек).

Таблица 4.24 – Фрагмент программы задержки с таймером

Начальные условия:
Подпрограмма DELAY005 дает задержку около 0,05 секунды.
Она вызывается 20 раз подряд
Результаты:
Программа выполнит задержку на 1 секунду

Метка	Команда		Комментарий
	MOV	R4,#20	; Счетчик вызовов задержки в 0,05 сек
SNOV2:	LCALL	DELAY005	; Вызов задержки в 0,05 сек
	DJNZ	R4, SNOV2	; Цикл на 20 повторений
	…	…	
	…	…	; Следующие команды программы
; Подпрограмма задержки на 0,05 секунды на основе таймера			
DELAY005:	MOV	TMOD,#00000001B	; Выбран таймер 0, режим 1,
			; (16-бит делитель), программный запуск
	CLR	TR0	; Остановка T/C0 (бит TCON.4)
	CLR	TF0	; Сброс готовности T/C0 (бит TCON.5)
	MOV	TH0,#3CH	; Начальное значение T/C0 равно (–50000)
	MOV	TL0,#0BEH	; расчет:(65536-50000= =15536=3CB0H)
			; Плюс 14=0EH, учтем остальные команды
	SETB	TR0	; Запуск T/C0 на счет
WAIT:	JNB	TF0, WAIT	; Ожидаем переполнения T/C0
	RET		; Возврат из подпрограммы

В приведенном фрагменте подпрограмма DELAY005 выполняет задержку на 0,05 сек. За счет повторных вызовов заново запускается таймер и задержка доводится до 1 сек. Число в таймере равно (–50000+14). Эти 14 единиц соответствуют 14 тактам для выполнения всех предыдущих команд перед запуском таймера.

Максимальный интервал задержки в данной программе можно получить около 13 сек.

Если на внешний вход таймера подавать низкочастотные импульсы, можно с помощью больших коэффициентов деления получать огромные длительности задержки. Это решение потребует наличия внешнего генератора, что весьма неудобно.

4.15 Микропроцессорная система на основе ОЭВМ КР1816ВЕ51

Система на базе MCS-51 может быть реализована без использования внешней памяти программ и/или данных. Это наиболее тривиальный случай системы, он не требует дополнительных пояснений для реализации. Наиболее сложным является случай построения системы с использованием внешней памяти программ и внешней памяти данных.

Система, имеющая полный объем внешней памяти программ и данных [6] (рис. 4.20), может использоваться для решения огромного числа прикладных задач в различных областях.

Буферные усилители (BF) в системе предназначены для усиления сигналов, имеющих слабую нагрузочную способность. Буферный усилитель, формирующий шину данных, должен работать в обоих направлениях. Поэтому для его переключения формируется смесь сигналов чтения PSEN и RD, по каждому из которых буфер включается на передачу сигналов в МК.

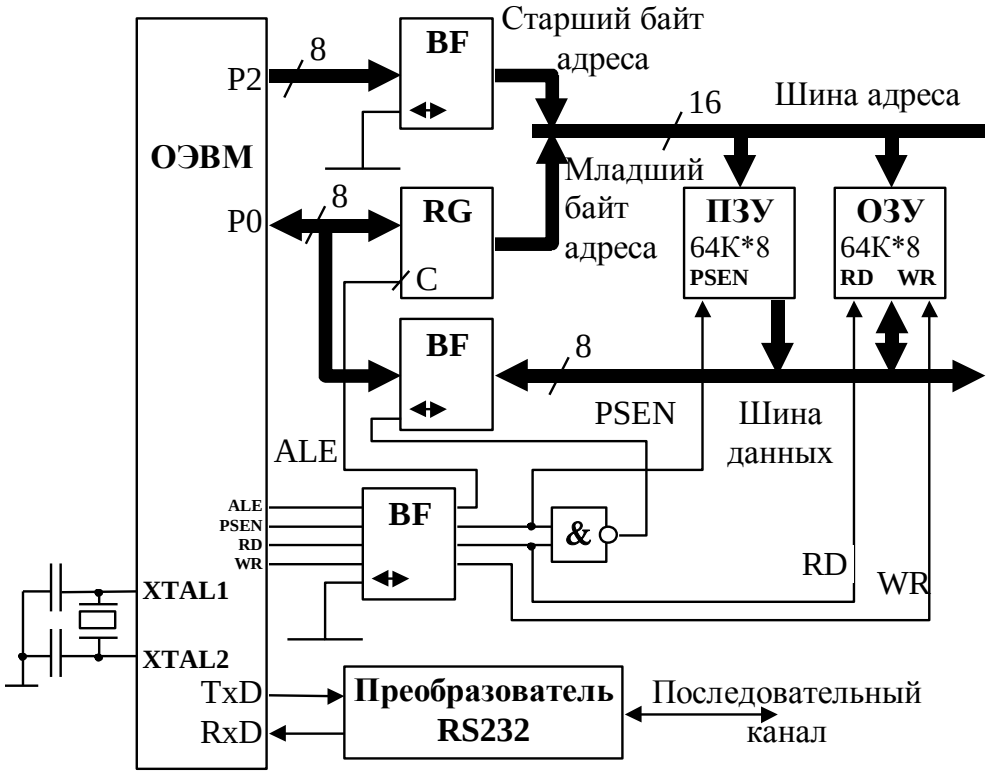

Рисунок 4.20 – Структурная схема системы с внешней памятью

Другие буферные усилители стационарно работают в одном направлении. Регистр RG по сигналу ALE записывает младший байт адреса внешней памяти, который вместе со старшим байтом адреса, передаваемого через порт P2, формирует 16-битную шину адреса. Принципиальная схема системы на основе MCS-51 (рис. 4.21) с 2 Кбайт ОЗУ данных и 2 Кбайт ПЗУ программ содержит 7 интегральных микросхем средней и большой степени интеграции. В системе объем памяти ОЗУ и ПЗУ можно увеличить до 4 Кбайт без изменения схемы.

Микропроцессорная система, не имеющая внешней памяти программ или внешней памяти данных, будет иметь более простой вид за счет исключения соответствующих компонентов (ПЗУ или ОЗУ) и их сигналов управления.

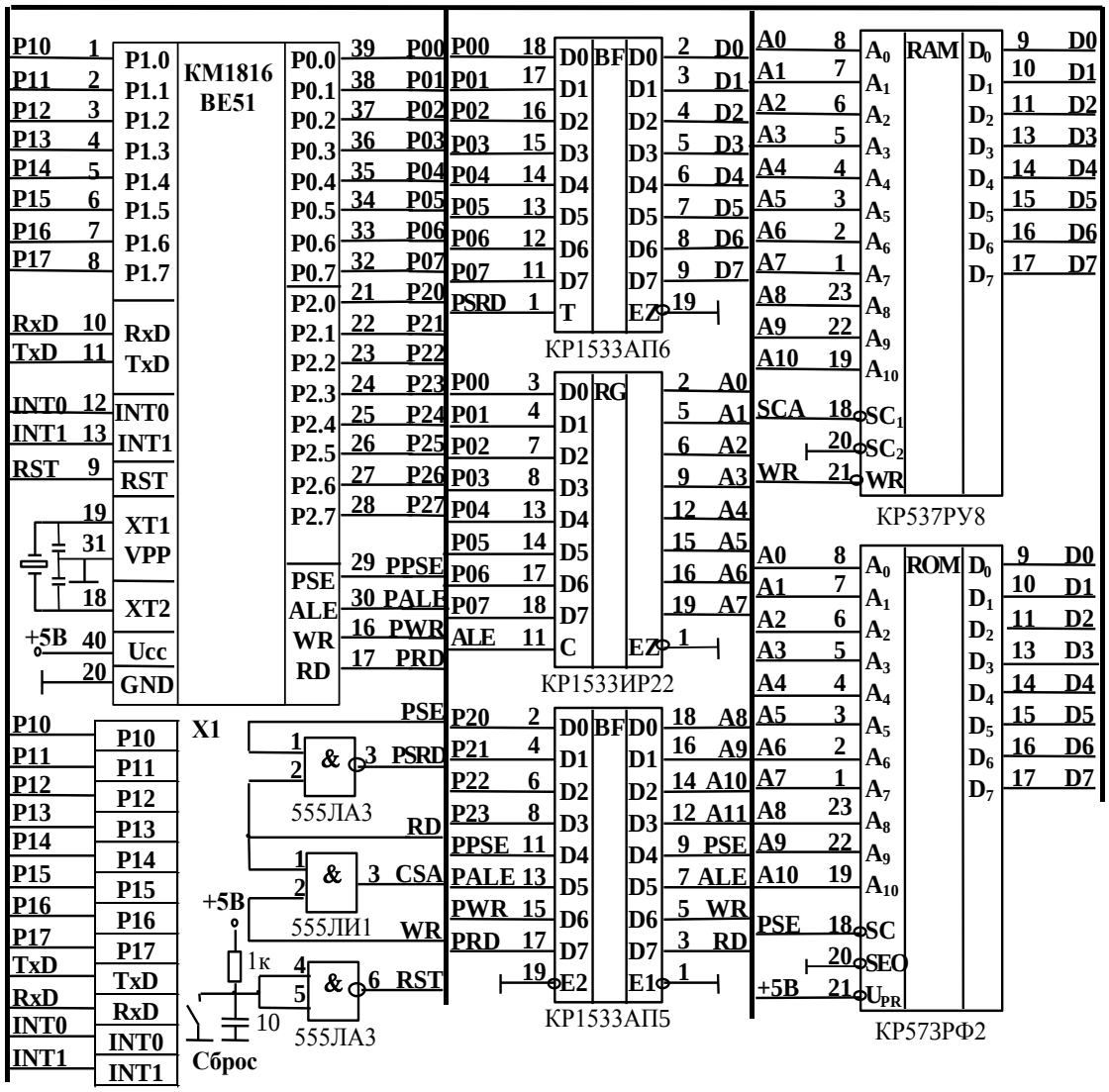

Рисунок 4.21 – Принципиальная схема системы с внешней памятью

4.16 Развитие архитектуры MCS-51

Микроконтроллеры архитектуры MCS-51 завоевали заслуженное признание во всем мире. Безусловно, данная архитектура не является единственно возможной. Но можно привести ряд факторов, свидетельствующих в пользу ее популярности и современности.

Фирма Cygnal (США) выпустила 4 новых семейства микроконтроллеров, совместимых с MCS-51.

Основным в новых микроконтроллерах является высокопроизводительное x8051 – совместимое ядро. 70% инструкций выполняется за 1 или 2 такта. На кристалл интегрирован 12 или 10 битовый Аналогово-Цифровой Преобразователь с входным 8-ми канальным аналоговым мультиплексором и программируемым входным усилителем, имеющим коэффициенты усиления от 0,5 до 16. На кристалле имеется два 12 разрядных Цифро-Аналоговых Преобразователя с временем реакции не более 10 мкс.

Питающее напряжение снижено до 2,7 – 3,6 В. Токи потребления не превышают 10 – 12 мА. Такие характеристики очень полезны для систем на базе ОЭВМ, построенных с автономным питанием. Тактовая частота ОЭВМ доведена до 25 МГц.

Встроенная аппаратная система отладки программ JTAG обеспечивает простоту разработки и внутрикристальной отладки программ с точками останова, пошаговым режимом, режимом остановки по времени с контролем ячеек памяти и регистров. Предлагается фирменное программное обеспечение процесса отладки.

На кристалле имеется до 32 Кбайт Flash-памяти с внутрисистемным программированием. Такая память сохраняет данные и при выключении питания. Кроме этого, оперативная память на кристалле увеличена до 1 Кбайт. Есть варианты с 2,25 Кбайт и даже с 4,25 Кбайт оперативной памяти на самом кристалле.

Таким образом, описанные микроконтроллеры компании Cygnal являются мощными интегрированными системами сбора и обработки аналоговых сигналов, сочетающими высокопроизводительное «традиционное» ядро семейства MCS-51 с аналогово-цифровыми и цифро-аналоговыми узлами. Микросхемы оснащены большими объемами Flash-памяти и большими объемами внутренней оперативной памяти. Пониженное напряжение питания с низкими токами потребления хорошо сочетается с высокими тактовыми частотами. Такие микро-

контроллеры могут эффективно сочетать задачи управления системой с задачами обслуживания большого числа входных датчиков, часть из которых даже не имеет цифровых выходов, и с задачами формирования, даже в аналоговой форме, необходимых управляющих воздействий.

Наличие энергонезависимой Flash-памяти, запись в которую можно выполнять при работе программ, позволяет протоколировать работу системы управления с целью дальнейшего выявления сбоев и не обслуженных запросов от внешних устройств.

Рассмотрение микроконтроллера MCS-51 было бы не полным, если не упомянуть о развитии архитектуры MCS-51. Следующим поколением архитектуры фирмы INTEL является архитектура MCS-251.

Новые продукты семейства MCS-251 совместимы по выводам с микросхемами INTEL 80C51 и могут устанавливаться прямо в панельку ОЭВМ. Эти ОЭВМ совместимы по системе команд и двоичному коду с MCS-51. При выполнении кода, написанного для MCS-51, обеспечивается повышение производительности в 5 раз.

В системе команд MCS-251 появилось много новых команд, учитывающих новые возможности данной архитектуры. Если переписать код программ, используя инструкции новой архитектуры, можно увеличить производительность системы до 15 раз.

Некоторые новые параметры архитектуры MCS-251 указаны ниже:

- машинный цикл состоит из одного состояния и длится 2 такта;
- большинство команд выполняется за 1 машинный цикл;
- процессор имеет 3-уровневую конвейерную архитектуру;
- доступны 16 8-ми разрядных, 16 16-ти разрядных и 10 32-х разрядных регистра общего назначения;
- при одинаковой тактовой частоте рост производительности от MCS-51 достигает от 5 до 15 раз;
- внутренняя шина команд содержит 16 бит, 2 байта команды выбираются за одно состояние;
- для адресации внешней памяти программ и данных предусмотрено 24 бита адреса, что расширяет объем внешней памяти данных и программ до 16 Мбайт;
- объем стека увеличен до 64 Кбайт;
- введены новые команды и режимы адресации;
- добавлена 32-х битовая передача данных;

- введена 16-ти и 32-х битовая арифметика и логические операции;
- к традиционным добавлены сдвиговые, относительные и битовые режимы адресации;
- в системе прерываний появилось 2 немаскируемых прерывания с наивысшим приоритетом;
- число источников прерываний достигло 64;
- имеются 4 уровня приоритетов, вместо 2, имеющихся ранее.

4.17 Контрольные вопросы

1. Что такое Гарвардская архитектура микро-ЭВМ?
2. Расшифруйте аббревиатуры ВПП, РПП, РПД, ВПД.
3. Укажите максимальный суммарный объем памяти программ с использованием внутренней и внешней памяти в МК-51.
4. Укажите максимальный суммарный объем памяти данных с использованием внутренней и внешней памяти в MCS-51.
5. Укажите разрядность регистров, ячеек памяти данных, регистра DPTR в MCS-51.
6. Как в MCS-51 настроить порт на ввод информации?
7. Укажите назначение в MCS-51 сигнала ALE.

5 КОМПОНЕНТЫ МИКРОПРОЦЕССОРНЫХ СИСТЕМ

5.1 Датчики

Датчиками называют устройства, формирующие электрические сигналы под воздействием внешних раздражающих факторов. По формируемым датчиками сигналам можно опознавать и измерять характеристики этих факторов. Существует множество явлений и эффектов, видов преобразования свойств и энергии, которые можно использовать для создания датчиков.

Для датчиков технических систем необходимо преобразование интересующих явлений и эффектов в электрические сигналы. Некоторые датчики выполняют такое преобразование напрямую, за счет соответствующего физического явления. Терморезистор, например, изменяет сопротивление в зависимости от окружающей температуры.

Другие датчики требуют промежуточного преобразования. Примером такого датчика можно назвать датчик обледенения, выполненный на основе оптического элемента. Осаждение инея приводит к изменению освещенности, которое и преобразуется в электрический сигнал.

Из всего многообразия подобных устройств рассмотрим несколько наиболее важных типов датчиков.

Температурные датчики. С температурой мы сталкиваемся ежедневно, это наиболее знакомая нам физическая величина. Температурные датчики отличаются особенно большим разнообразием типов и являются самыми распространенными (таблица 5.1).

Таблица 5.1 – Температурные датчики

Тип датчика	Принцип действия	Примерный диапазон температур
Биметаллический датчик	Тепловое расширение	-50° - +500° С
Платиновый термометр сопротивления	Изменение электрического сопротивления	-50° - +300° С
Термопара	Генерация термо-ЭДС	-100° - +1300° С
Термоферрит	Изменение магнитной проницаемости	-20° - +150° С
Диод, транзистор, тиристор	Изменение проводимости	-20° - +100° С
Инфракрасный пироэлек-	Тепловое излучение	-150° - +1300° С

Тип датчика	Принцип действия	Примерный диапазон температур
трический детектор		
Кварцевый резонатор	Изменение частоты	-20° - +200° С
Плавкий предохранитель	Деформация, разрушение	+40° - +400° С

Как можно заключить из приведенной таблицы, различные температурные датчики имеют различную чувствительность и разный диапазон рабочих температур. В последнее время практическое применение нашли интегральные температурные датчики, имеющие на одном кристалле термочувствительный диод, усилитель и периферийные схемы для связи с микро-ЭВМ.

Оптические датчики. Подобно температурным, оптические датчики отличаются большим разнообразием и массовостью применения. По принципу преобразования «свет - электрический сигнал» в оптических датчиках можно выделить типы, основанные на следующих явлениях:

- эффект фотоэлектронной эмиссии;
- эффект фотопроводимости;
- фотогальванический эффект;
- пироэлектрический эффект.

Фотоэлектронная эмиссия – это испускание электронов при падении света на физическое тело.

Эффект фотопроводимости связан с изменением электрического сопротивления физического тела при облучении его светом.

Фотогальванический эффект – это возникновение ЭДС на выводах p-n перехода в облучаемом светом полупроводнике.

Пироэлектрический эффект связан с появлением на поверхности физического тела электрических зарядов при изменении уровня освещенности.

Преимущества оптических датчиков перед датчиками других типов заключаются в следующем:

- возможность бесконтактного обнаружения;
- возможность за счет изменения оптики работать с объектами чрезвычайно больших или очень малых размеров;
- высокая скорость реакции;
- обширная сфера использования, например, от измерения перемещения до определения формы и распознавания предметов.

Однако среди оптических датчиков практически нет датчиков, обладающих достаточной чувствительностью во всем световом диапазоне. Большинство датчиков имеют оптимальную чувствительность в довольно узкой зоне ультрафиолетовой, видимой или инфракрасной части спектра.

К недостаткам оптических датчиков следует также отнести чувствительность к загрязнению, уровням посторонней фоновой засветки.

Датчики давления. Такие датчики используются в задачах измерения давления газов, массы и положения твердых тел, уровня и расхода жидкости.

Обычным способом реакции на изменение давления в таких датчиках является эффект деформации упругих тел (диафрагмы, мембраны и т.д.). Но при использовании эффекта деформации трудно получить электрический сигнал. В датчиках давления типа потенциалометрических (реостатных), емкостных, индукционных и ультразвуковых на выходе имеется электрический сигнал, но сами датчики более дорогие и сравнительно сложны в изготовлении.

Более перспективным является использование тензометров. Полупроводниковые диффузионные тензометры обладают высокой чувствительностью, малыми размерами, легко сопрягаются с периферийными схемами.

При изготовлении тензометра на поверхности кристалла кремния с n-проводимостью по тонкопленочной технологии формируется круглая диафрагма. На ее краях (рисунок 5.1) наносят пленочные резисторы, имеющие p-проводимость и ориентированные вдоль осей.

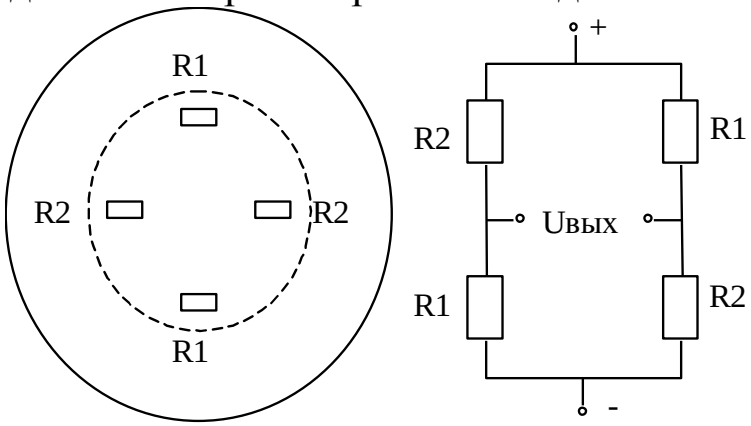

Рисунок 5.1 – Конструкция и схема тензометрического датчика

При приложении давления сопротивление одной пары резисторов увеличивается, а другой – уменьшается. С помощью мостовой схемы

формируется выходной сигнал датчика. Недостатками этого типа датчика является температурная зависимость, небольшой срок службы.

Полупроводниковые датчики давления широко используются в автомобильной электронике, компрессорах, системах измерения артериального давления крови и т.д.

Датчики влажности и газоанализаторы. Влажность – физический параметр, с которым человечество имеет дело очень давно. Раньше датчиком влажности выступал человеческий или конский волос, изменяющий свою длину при изменении влажности. Позднее применяли специальную полимерную пленку, разбухающую от воды. Однако, датчики на этой основе обладают нестабильностью характеристик во времени, узким рабочим диапазоном, наличием гистерезиса чувствительности.

В настоящее время для датчиков используется специальная пористая керамика и твердые электролиты. В этих датчиках устранены описанные выше недостатки. Датчики на основе керамики используются в видеомагнитофонах и видеокамерах для обнаружения повышенной влажности с последующим переводом аппаратуры в нерабочее состояние до высыхания. Другой областью применения этих датчиков можно назвать схемы управления электронными кухонными плитами. В плитах по уровню влажности можно судить о степени готовности блюда.

Газовые датчики или газоанализаторы используются в бытовых случаях для обнаружения утечек горючего газа. Датчики могут использовать свойства явления катализа в твердых электролитах, интерференции и поглощения инфракрасных лучей, свойств полупроводниковой керамики, работающей по принципу каталитического горения.

Однако газовые датчики обладают большой избирательной характеристикой относительно газовой среды.

Магнитные датчики. Магнитные датчики, как и оптические, имеют возможность бесконтактного измерения и обнаружения, обладают высоким быстродействием. Однако для этих датчиков важен фактор расстояния, требуется достаточная близость к источнику магнитного поля. По классификации магнитных датчиков можно выделить датчик Холла, магниторезистор, датчик Джозефсона.

Среди магнитных датчиков широко известен *датчик Холла*. Принцип эффекта Холла основан на возникновении разности потенциалов на гранях твердого тела при протекании тока в условиях приложения магнитного поля перпендикулярно направлению электрического тока. Датчики Холла в интегральном исполнении широко применяются в двига-

телях видеомагнитофонов и видеокамер для определения положения, угла поворота и управления частотой вращения двигателя.

Магниторезистивные датчики изменяют свое сопротивление в магнитном поле. Более ранними являются магниторезисторы на основе полупроводников. Сейчас активно разрабатываются магниторезисторы на основе ферромагнетиков. Для последнего вида характерна высокая чувствительность и технологичность производства. Датчики применяются в магнитных головках многодорожечных цифровых магнитофонов. Недостатком отмечают узкий динамический диапазон обнаружения изменений магнитного поля.

Перспективными являются магнитные *датчики на эффекте Джозефсона* – явлении низкотемпературной сверхпроводимости, зависящей от магнитных полей. Датчики на этом принципе отличаются сверхвысокой чувствительностью к изменению магнитного поля. В качестве недостатка можно назвать относительную сложность изготовления и применения, особенно за счет необходимости использования очень низких температур при работе.

Перечисленные типы датчиков являются очень интересными и распространенными. Можно также назвать и другие типы датчиков: звуковые, радиационные, рентгеновские, СВЧ-датчики, датчики вибрации, скорости вращения, датчики вкуса, запаха и т.п.

К современным датчикам предъявляются довольно жесткие требования:

- Высокие качественные характеристики: чувствительность, точность, линейность, скорость отклика, взаимозаменяемость.
- Высокая надежность: длительный срок службы, устойчивость к влияниям внешней среды, безотказность в работе.
- Технологичность: малые габариты, масса, низкая себестоимость, простота конструкции.

При подключении датчиков к устройству управления, выполненного на основе микро-ЭВМ, появляются дополнительные возможности и для самих датчиков. Частично удается скомпенсировать недостатки датчиков. При этом:

- линеаризуется нелинейная характеристика конкретного типа датчиков;
- подавляются шумы датчика;
- корректируются чувствительность датчика и точка нуля, которые могут изменяться при длительной эксплуатации;

- производится автоматическая диагностика датчиков.

В дальнейшем для разработки датчиков будут ужесточены требования:

- Интегральное исполнение. В датчиках обязательными будут интерфейсные схемы, усилители, АЦП и.т.д.
- Комбинирование. В одном корпусе нужно объединять несколько датчиков, либо объединять датчики с исполнительными устройствами.
- «Интеллектуализация». На кристалле датчиков необходим микропроцессор, обрабатывающий и контролирующий состояние датчиков. Микропроцессор также должен принимать решение относительно полученных данных. Такая необходимость актуальна в системах обеспечения безопасности, где недопустима ложная информация и важна малая протяженность и недоступность линии передачи данных к устройству управления.

5.2 Отображение информации для МП

5.2.1 Термины и определения

Для отображения информации в МП применяются различные индикаторы. В основе принципов действия индикаторов лежат весьма разнообразные физические явления, такие как низковольтная катодолюминесценция, инжекционная и предпробойная электролюминесценция, излучение газового разряда, различные электрооптические эффекты и т. п. Индикаторы различаются функциональными возможностями и назначением, конструктивным и технологическим исполнением. В литературе по индикаторам особенности конкретных типов индикаторов отражаются рядом специальных терминов, а также условными обозначениями.

В таблице 5.2 приводятся основные термины индикаторов и их определения (согласно ГОСТ 25066-81).

Таблица 5.2 – Основные термины индикаторов и их определения

Индикатор	Выходное устройство информационного прибора или системы, обеспечивающее визуальное (видимое) отображение информации, воспринимаемое человеком в удобном для наблюдения виде.

Активный индикатор	Индикатор, преобразующий электрическую энергию в световую
Пассивный индикатор	Индикатор, преобразующий (модулирующий) внешний световой поток под действием электрического поля или тока
Вакуумный накаливаемый индикатор	Активный индикатор, в котором используется свечение тел накала в вакууме
Вакуумный люминесцентный индикатор (ВЛИ)	Активный индикатор, в котором используется явление низковольтной катодолюминесценции
Полупроводниковый индикатор	Активный индикатор, в котором используется явление инжекционной электролюминесценции
Жидкокристаллический индикатор (ЖКИ)	Пассивный индикатор, в котором используются электрооптические эффекты в жидких кристаллах
Электролюминесцентный индикатор	Активный индикатор, в котором используется явление предпробойной электролюминесценции
Газоразрядный индикатор	Активный индикатор, в котором используется видимое излучение газового разряда
Индикатор индивидуального, коллективного пользования	Индикатор, конструктивное исполнение, параметры и характеристики которого обеспечивают возможность безошибочного считывания информации с расстояния: • менее 1,5 м (индивидуального пользования); • от 1,5 до 4 м (группового пользования); • более 4 м (коллективного пользования)
Информационное поле	Конструктивная часть индикатора, в пределах которой возможно формирование изображения
Знак	Условное обозначение букв алфавита, цифр, математических знаков, знаков препинания, предметов, явлений, событий
Знакоместо	Информационное поле или его часть, необходимая и достаточная для изображения одного знака
Элемент (сегмент) отображения	Часть информационного поля, имеющая самостоятельное управление
Структурный рисунок индикатора	Изображение, возникающее при включении всех элементов индикатора и показывающее их число, форму, размеры и возможное расположение при

	отображении информации
Знакосинтезирующий индикатор (ЗСИ)	Прибор отображения информации, в котором видимое изображение создается из одного или совокупности дискретных элементов отображения
Единичный индикатор	Индикатор, состоящий из одного элемента отображения и предназначенный для отображения информации в виде точки, линии или поля
Одноразрядный индикатор	Цифровой или буквенно-цифровой индикатор, имеющий одно знакоместо
Многоразрядный индикатор	Цифровой или буквенно-цифровой индикатор, имеющий несколько фиксированных знакомест
Матричный индикатор	ЗСИ, элементы отображения которого имеют вид точек и сгруппированы по строкам и столбцам, что позволяет отображать информацию произвольного характера
Экран	Матричный индикатор с числом элементов отображения не менее 10000 и не имеющий фиксированных знакомест
ЗСИ со встроенным управлением	ЗСИ, конструктивно выполненный совместно с элементами системы управления
Шкальный индикатор	Индикатор, предназначенный для отображения в виде шкал
Графический индикатор	Индикатор, предназначенный для отображения букв, цифр, символов, графиков и другой сложной графической информации
Мнемонический индикатор	Индикатор, предназначенный для отображения мнемосхемы или части мнемосхемы

Рассмотрим особенности строения, условия применения и схемы управления наиболее распространенных индикаторов, применяемых в бытовой РЭА.

5.2.2 Вакуумные люминесцентные индикаторы

Вакуумные люминесцентные индикаторы (В Л И) относятся к активным индикаторам, преобразующим электрическую энергию в световую. По виду отображаемой информации ВЛИ можно разделить на единичные, цифровые, буквенно-цифровые, шкальные, мнемонические и

графические. По виду информационного поля индикаторы делятся на сегментные и матричные, одноразрядные и многоразрядные, а также матрицы без фиксированных знакомест.

К числу достоинств ВЛИ следует отнести:

- высокую яркость, обеспечивающую хорошую видимость воспроизводимых знаков;
- низкие рабочие напряжения, допускающие возможность их применения с формирователями на МОП-микросхемах;
- малое потребление энергии, что позволяет использовать их в устройствах, питаемых от батарей и автономных источников питания.

Необходимость использования источника питания накала индикатора может оказаться его недостатком. В ряде случаев трудно исключить мешающие восприятию изображения блики, создаваемые отражением света от стеклянных баллонов индикаторов.

Вакуумные люминесцентные индикаторы используют для отображения информации в устройствах самого различного назначения: в микрокалькуляторах и ЭВМ, кассовых аппаратах и станках с числовым и программным управлением, электронных часах, электро- и радиоизмерительных приборах (цифровых ампервольтомметрах, частотомерах), диспетчерских пультах управления энергетическими установками и воздушным движением, медицинских приборах и т. п.

Вакуумный люминесцентный индикатор представляет собой электронную диодную или триодную систему, в которой под воздействием электронной бомбардировки высвечиваются покрытые низковольтным катодолюминофором аноды-сегменты.

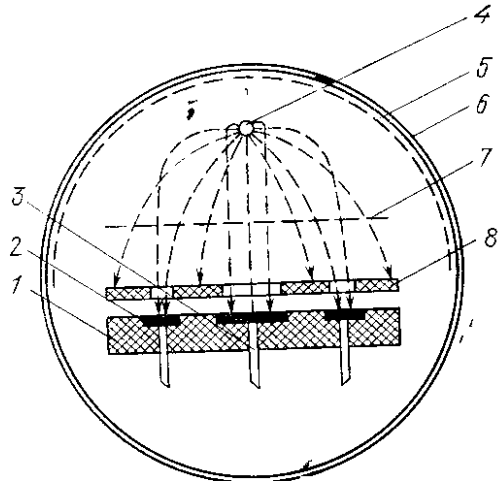

Рисунок 5.2 – Конструктивная схема ВЛИ

Конструктивная схема одноразрядного индикатора показана на рис.5.2**Ошибка! сточник ссылки не найден.**. Детали индикатора монтируются на керамической или стеклянной плате 1. Участки платы, на которые нанесен люминофор, образуют аноды-сегменты 2; под люминофором имеется токопроводящий слой. Каждый из анодов имеет определенный вывод *3.* Источником электронов служит оксидный катод прямого накала *4.* Управление электронным потоком осуществля-

ется сеткой 7. Электронный поток, высвечивающий сегменты, ограничивается экранирующим электродом-маской *8.* Вся арматура индикатора заключена в стеклянный баллон 6, в котором создан вакуум. Штриховой линией показаны примерные траектории электронов. На внутреннюю поверхность баллона нанесено токопроводящее покрытие 5, прозрачное для всей области спектра излучения индикатора. Электрически оно соединено с отдельным выводом или катодом; покрытие обеспечивает стекание с поверхности баллона электрических зарядов, способных исказить траектории электронов.

Катод ВЛИ представляет собой отрезок тонкой вольфрамовой проволоки. Рабочая температура катода выбирается низкой, чтобы нить, находящаяся по направлению наблюдения перед анодами, не мешала наблюдению светящихся символов.

Сетка ВЛИ управляет электронным потоком. Она имеет положительный относительно катода потенциал, рассеивает электроны и ускоряет их в направлении анодных сегментов. Сетки изготовляются из полотна, «тканого» из вольфрамовой проволоки или электрохимическим фрезерованием тонкой никелевой фольги.

Изображение букв, цифр и других символов во ВЛИ формируется высвечиванием необходимой комбинации анодов-сегментов. Смена изображений достигается путем соответствующей коммутации анодов-сегментов. Аноды-сегменты представляют собой покрытые люминофором слои токопроводящего материала заданной конфигурации, нанесенные на стеклянную или керамическую плату. Сегменты выполнены в виде точек или протяженных участков различной формы, символов и трафаретов. Количество, конфигурация и взаимное расположение сегментов образует структурный рисунок индикатора, по которому различают цифровые, буквенно-цифровые, матричные и шкальные индикаторы.

Люминофор включенных сегментов, имеющих в данный момент положительный относительно катода потенциал, светится под воздействием попадающего на них электронного потока. Ток катода индикатора и токи сегментов практически не зависят от числа включенных сегментов. Электроны, попадающие на включенные сегменты, заряжают их отрицательно и отражаются. Вторичные электроны перехватываются экранирующим электродом.

Изображение, формируемое ВЛИ, очень высококонтрастное, яркость достигает 500 кд/м^2 и более; для сравнения можно напомнить, что яркость экрана современного цветного кинескопа не превышает 300 кд/м^2. В ВЛИ используется явление низковольтной катодолюминесцен-

ции (НВК), при котором свет излучается кристаллофосфором, бомбардируемым электронами с относительно низкой энергией (около 10... 100 эВ). Для веществ, у которых наблюдается этот эффект, потенциал начала НВК составляет всего несколько вольт.

В ВЛИ в качестве люминофора широко используется окись цинка, активированная цинком (ZnO:Zn), обеспечивающая интенсивное сине-зеленое свечение. Применяя светофильтры, можно получить цвета символов от синего до красного. Существует достаточно широкая номенклатура люминофоров, имеющих различные цвета свечения (синий, зеленый, желтый, красный).

Матричные ВЛИ предназначены для синтеза цифр, букв любого алфавита, различных символов и знаков. Формирование изображения на информационном поле ВЛИ можно осуществлять статическим или мультиплексным способом. При статическом способе возбуждающие сигналы подаются на необходимые для получения заданного изображения аноды-сегменты, и все изображение знака формируется одновременно.

Формирование изображения мультиплексным способом осуществляется применительно к ВЛИ, имеющим два канала управления, например, многоразрядные индикаторы с параллельно соединенными анодами-сегментами и раздельными для каждого знакоместа сетками. Так же управляют матричными и графическими ВЛИ.

При мультиплексном управлении в течение каждого момента времени формируется не полное изображение, а его отдельные элементы. За счет инерционности человеческого зрения наблюдатель видит цельную картинку. Такое управление еще называют динамической индикацией.

Долговечность работы ВЛИ определяется сохранением работоспособности люминофора и оксидного катода. В течение первых сотен часов яркость претерпевает спад на 10…20 %. После этого следует довольно длительный период, составляющий десятки тысяч часов, в течение которых яркость практически не изменяется.

5.2.3 Жидкокристаллические индикаторы

Принцип действия и конструктивные модификации

Жидкокристаллические индикаторы (ЖКИ) являются пассивными индикаторами, преобразующими падающий на них свет. Они обладают рядом достоинств, к числу которых относятся:

- малая потребляемая мощность (для ЖКИ на основе твист-эффекта удельная мощность потребления составляет несколько единиц мкВт/см^2);
- низкие рабочие напряжения (1,5–5 В) и хорошая совместимость с КМОП- микросхемами;
- удобное конструктивное исполнение - плоская форма экрана и ограниченная толщина индикатора (до 0,6 мм);
- возможность эффективной индикации в условиях сильной внешней засветки;
- большая долговечность (около 10-12 лет непрерывной работы).

Основные недостатки - сравнительно низкое быстродействие, ограниченный угол обзора и необходимость внешнего освещения.

Жидкие кристаллы (ЖК) называют также анизотропными жидкостями, электрические и оптические свойства которых зависят от направления их наблюдения. Плотность ЖК близка к плотности воды и коэффициент отношения их плотностей незначительно отличается от единицы. Жидкие кристаллы - диамагнитный материал, ЖК выталкиваются из магнитного поля; ЖК относятся к диэлектрикам; удельное сопротивление составляет 10^6... 10^{10} Ом·см и зависит от наличия и концентрации проводящих примесей. Теплопроводность ЖК в направлении вдоль молекул отличается от теплопроводности в поперечном по отношению к молекулам направлении.

Вследствие анизотропии электрических и оптических свойств в ЖК наблюдаются различные электрооптические эффекты:

- связанные с движением вещества - динамическое рассеяние (ДР);
- с поворотом молекул в электрическом поле - твист-эффект (ТЭ);
- эффект гость - хозяин (Г - Х).

Конструктивные схемы ЖКИ показаны на рисунках 5.3 и 5.4.

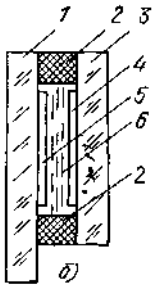

Рисунок 5.3 – Конструкция ЖКИ, работающего на просвет: 1,3 - стеклянные пластины;

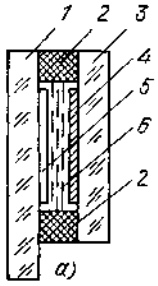

Рисунок 5.4 – Конструкция ЖКИ, работающего на отражение: 1,3 - стеклянные пластины;

2 - склеивающее соединение; 4 - задний прозрачный электрод; 5 - передний прозрачный электрод; 6 - жидкий кристалл

2 - склеивающее соединение; 4 - задний отражающий электрод; 5 - передний прозрачный электрод; 6 - жидкий кристалл

Основой простейшего индикаторного элемента с использованием ЖК являются две стеклянные пластины. Вне зависимости от используемого электрооптического эффекта ЖКИ разделяются на два класса:

- индикаторы, работающие на просвет;
- индикаторы, работающие на отражение.

У первого типа (рис 5.3) обе стеклянные пластины прозрачны, электродами служат прозрачные электропроводящие пленки (например, двуокись олова), между которыми помещено ЖК вещество. За индикатором помещается источник света. Цвет и ярость индикатора определяются цветом и яркостью источника света.

У второго типа (рис 5.4) «задний» электрод изготовлен в виде зеркала, на соответствующую пластину наносится прозрачная проводящая и отражающая свет пленка (например, пленка алюминия, никеля, золота). Такой индикатор использует внешнее отражающее освещение (специальная подсветка отсутствует).

Пример ЖК-индикатора со сложным структурным рисунком, применяемого в устройствах бытовой аппаратуры, с указанием реальных размеров, приведен на рисунке 5.5.

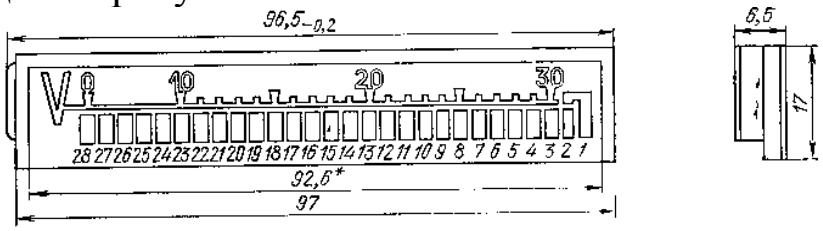

Рисунок 5.5 – ЖК индикатор со сложным структурным рисунком

На рисунке 5.6 показан набор букв, знаков и цифр, которые можно получить на 16-сегментном знакосинтезирующем ЖК индикаторе.

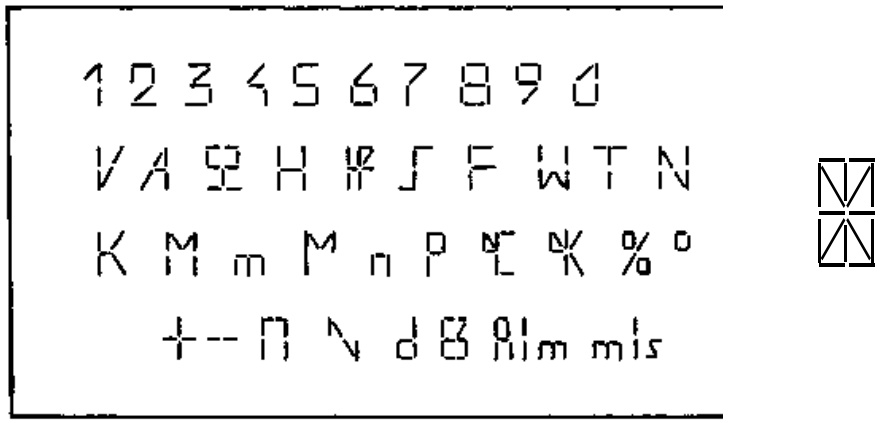

Рисунок 5.6 – Изображения букв, знаков и цифр, получаемых на 16 сегментном ЖК индикаторе

Конфигурация электродов индикатора определяется либо формой исходных стеклянных пластин, либо технологией металлизации. Как правило, пластины и электроды плоские, но в ряде приборов внутренняя поверхность задней пластины имеет сложную форму, образующую ряд оптических элементов, обеспечивающих отражение излучения в направлении источника света.

В **ЖКИ, работающем на основе Динамического Рассеяния** (ДР), при приложении электрического поля напряженностью около 5 кВ/см (примерно 30 В - к пленке ЖК толщиной 0,25 мм) молекулы переориентируются, возникает турбулентность и сильное оптическое рассеяние. Материал, прозрачный в отсутствии поля, становится непрозрачным. В таком ЖКИ, работающем на отражение, задний электрод представляет собой зеркало, на котором при подаче напряжения появляются участки молочно-белого цвета, форма которых соответствует конфигурации электродов. Заднюю стеклянную пластину индикатора чернят, тогда на черном фоне возникает белое изображение.

В **ЖКИ с использованием Твист-Эффекта (ТЭ)**, работающем на отражение, стеклянные пластины расположены между двумя скрещенными поляризаторами, за задним из которых помещен диффузный отражатель. Поверхности пластин, обращенные к ЖК, полируются так, чтобы молекулы ЖК в слоях, прилегающих к ним, ориентировались во взаимно перпендикулярных направлениях, в промежуточных слоях осуществляется постепенный поворот направления ориентации.

В отсутствии электрического поля свет в индикаторе следует за вращением молекул и на выходе индикатора плоскость его поляризации оказывается повернутой на 90°; свет проходит через индикатор. Поляри-

затор пропускает свет только определенной поляризации. При наличии электрического поля ориентация молекул изменяется, все молекулы ориентируются вдоль электрического поля. Плоскость поляризации света, проходящего в этом случае через индикатор, не вращается, и свет не проходит через индикатор. Так как отражатель диффузный, на слабоокрашенном сером фоне отображаются темные знаки.

В ЖКИ на основе ТЭ, работающем на просвет, поляризаторы устанавливают так, чтобы их плоскости поляризации были параллельны друг другу. Индикатор не пропускает свет в отсутствие электрического поля и пропускает при подаче напряжения.

К числу достоинств таких ЖКИ относится высокая эффективность.

- Индикаторы на эффекте ДР характеризуются уровнем потребляемой мощности 5…10 мкВт/см^2 для постоянного тока (0,5...1,0 мкА/см^2) и 50...200 мкВт/см^2 для переменного тока (2...10 мкА/см^2).
- Рабочее напряжение ЖКИ на эффекте ДР не превышает 20 В.
- Для индикаторов на основе ТЭ удельная потребляемая мощность составляет не более 20 мкВт/см^2 (менее 2 мкА/см^2).
- Рабочее напряжение ЖКИ на эффекте ТЭ не превышает 5 В.
- Долговечность при эксплуатации на переменном токе составляет более 40 тысяч часов.

По экономичности ЖКИ намного превосходят современные светоизлучающие диоды. К достоинствам ЖКИ на эффекте ДР и ТЭ можно отнести способность сохранять и увеличивать контраст изображения при повышении уровня внешней освещенности, прямую совместимость с КМОП-микросхемами, обеспечивающую возможность низковольтного управления ЖКИ. Они имеют удобное конструктивное оформление. Индикаторы плоские; толщина индикатора практически определяется толщиной двух стекол и может составлять 0,6 – 0,8 мм.

Вместе с тем ЖКИ характеризуются сравнительно низким быстродействием (десятки миллисекунд, особенно при пониженной температуре) и явно выраженной зависимостью параметров от температуры окружающей среды.

Индикаторы на эффекте ДР и ТЭ преимущественно применяются там, где экономичность играет решающую роль: в электронных наручных часах, микрокалькуляторах с автономным питанием, портативных многофункциональных измерительных приборах, индикаторах для перенос-

ных радиоприемников, магнитофонов, автомобильных индикаторных устройствах и т. п.

В **индикаторах на эффекте Гость - Хозяин** тонкий слой ЖК - «хозяина» взаимодействует с молекулами «гостя». Слой ЖК – «хозяина» за счет поглощения световой энергии при отсутствии электрического поля приобретает характерную для красителя «гостя» окраску, а под воздействием электрического поля он обесцвечивается. В этом случае на окрашенном фоне возникают участки, имеющие другой цвет.

Но существуют также вещества гостя и хозяина, в которых окрашивание происходит под воздействием электрического поля. Цветовые различия в индикаторах на эффекте Г-Х хорошо воспринимаются в условиях высокой освещенности даже при небольшом яркостном контрасте.

5.2.4 Управление ЖКИ

Способы управления индикаторными панелями (ИП) на основе ЖК материалов определяются особенностями их физических свойств. Долговечность ЖКИ, работающего на постоянном токе, примерно на порядок ниже, чем при использовании переменного напряжения. Снижение долговечности в варианте постоянного тока обусловлено миграцией примесей к отражающему электроду под воздействием постоянной составляющей управляющего сигнала, в результате этого падает контрастность и растет напряжение возбуждения.

Предпочтительным оказывается возбуждение ЖКИ переменным током. В этом случае на электроды передней и задней пластин подаются импульсы напряжения прямоугольной формы (рисунок 5.7 а) одинаковой полярности, но сдвинутые по фазе так, что управляющее напряжение представляет собой биполярный сигнал, не имеющий постоянной составляющей (рисунок 5.7 б).

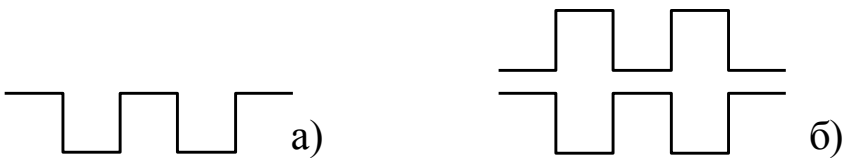

Рисунок 5.7 – Знакопеременное управляющее напряжение ЖКИ:
а - импульсы напряжения прямоугольной формы;
б - напряжения, сдвинутые по фазе

Принципиальная схема возбуждения ЖКИ переменным током (фазовым методом) приведена на рисунке 5.8.

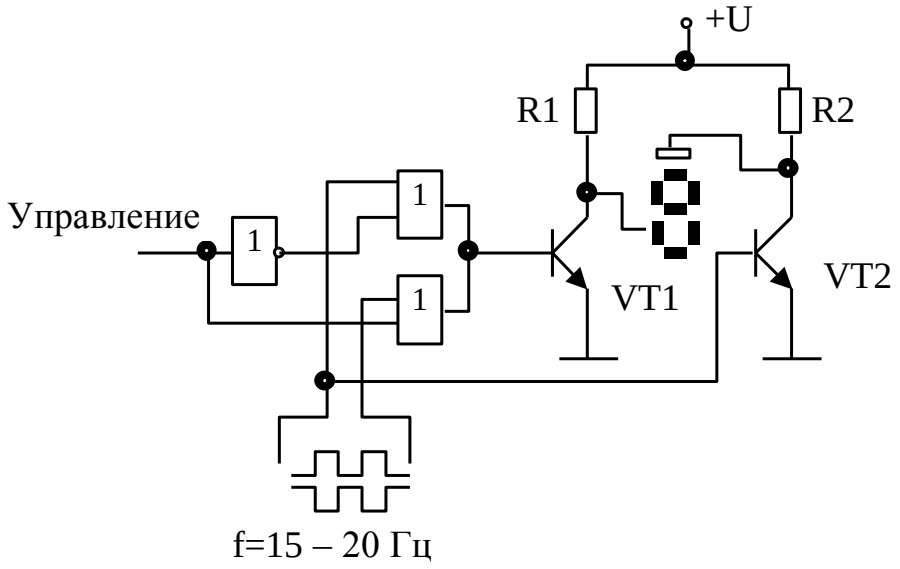

Рисунок 5.8 – Принципиальная схема возбуждения ЖКИ фазовым методом

В схеме предусматривается подача на входы вентилей ИЛИ импульсов напряжения с частотой 15... 20 Гц, сдвинутых по фазе относительно друг друга на 180°. В зависимости от уровня управляющего сигнала на сегмент с выхода формирователя подаются напряжения различных фаз. Сегмент не возбуждается при совпадении фаз на электродах ЖКИ, возбуждение происходит при различных фазах.

Управление многоразрядными ЖКИ может осуществляться в статическом или динамическом режиме.

Структурная схема управления индикатором в статическом режиме показана на рисунке 5.9. Каждое знакоместо индикатора $З_1 – З_n$ подключено к регистру оперативной памяти. Каждая кодовая комбинация регистра преобразуется в сегментный код индикатора дешифраторами управления ДУ, с выхода которых информация в коде индикатора через ключи блока формирователей БФ используется для коммутации питания сегментов индикатора.

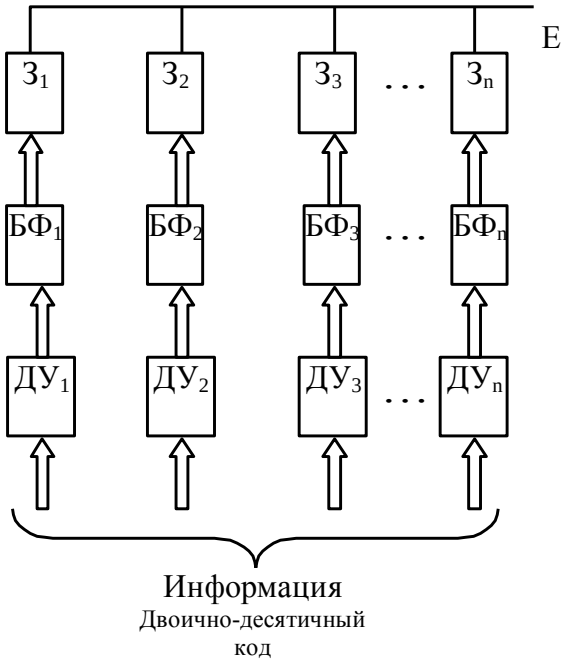

Рисунок 5.9 – Структурная схема управления индикатором в статическом режиме

Для этого устройства управления характерно полное использование контраста знакоместа, так как время возбуждения свечения равно длительности цикла индикации. Недостаток схемы - необходимость иметь для каждого знакоместа свой дешифратор и формирователь для каждого сегмента. Число внутрисхемных соединений велико, оно равно произведению числа выходов на один цифровой разряд на число цифровых разрядов.

При динамическом управлении пространственно разделенные разряды работают последовательно во времени. Возможны разные типы динамического управления. На рисунке 5.10 приведена структурная схема управления ЖКИ в динамическом режиме с последовательной выборкой знакоместа.

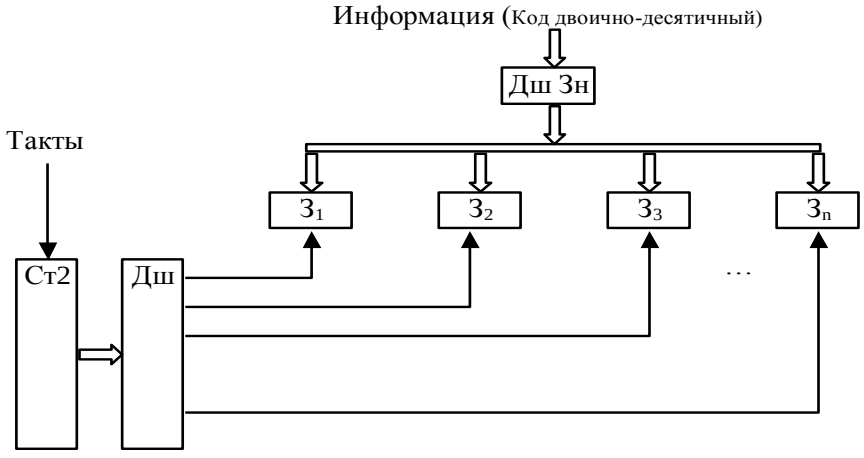

Рисунок 5.10 – Структурная схема управления индикатором в динамическом режиме

Счетчик Ст2 формирует через дешифратор Дш импульсы активизации знакомест $З_1 – З_n$. Информация с регистра памяти проходит через дешифратор знаков Дш Зн и поступает на индикаторы знакомест. Каждый индикатор работает небольшую часть общего времени. Достоинство схемы – в малом числе компонентов схемы управления и более низкое энергопотребление.

Частота тактов должна быть равной или выше некоторой критической частоты $f_{кр}$, при которой мерцание разрядов незаметно.

5.2.5 Долговечность ЖКИ

В процессе эксплуатации ЖКИ изменяется внешний вид информационных полей, что проявляется как ухудшение и исчезновение контраста между активными и пассивными зонами, увеличивается время реакции. Изменения внешнего вида и времени реакции являются следствием электрохимических явлений на границе «жидкокристаллическое вещество (ЖКВ) - поверхность подложки». Скорость деградационных процессов в основном определяется постоянной составляющей напряжения возбуждения, предельно допустимое значение которого указывается в справочных данных.

Наличие постоянной составляющей приводит к электролизу ЖКВ, в результате которого возникает газовыделение в объеме ЖКВ, образуются пузырьки газов, визуально воспринимаемые как черные точки. Электроды индикатора (проводящие пленки) теряют свою прозрачность, и сегменты становятся видимыми в отсутствие напряжения возбуждения.

В результате старения нарушается ориентация молекул ЖКВ и растет потребляемый индикатором ток.

В процессе эксплуатации ЖКВ потребляемый ток может расти за счет проникновения влаги через слой герметика. Влага разрушает ЖКВ. Особенно опасно сочетание влаги с воздействием высокой температуры.

При эксплуатации ЖКИ в условиях низкой температуры отдельные компоненты ЖКВ могут кристаллизоваться. Чередование замораживания и размораживания ЖКВ может привести к образованию воздушных пузырьков, которые выглядят как черные точки.

Индикаторы следует возбуждать переменным напряжением. Значение постоянной составляющей не должно превышать 250 мВ. Для контактирования выводов индикатора со схемой рекомендуется применять разъемы. Необходимо защищать кромку индикатора по всему периметру (место герметизации) от воздействия влаги.

5.2.6 Полупроводниковые знакосинтезирующие индикаторы

Единичные ППЗСИ

Единичный полупроводниковый знакосинтезирующий индикатор (ППЗСИ) – это полупроводниковый диод, в переходе которого в результате рекомбинации электронов и дырок при их инжекции в прямом направлении генерируется световое излучение. Принцип их работы заключается в том, что при прямом смещении потенциальный барьер *p-n* перехода понижается и происходит инжекция электронов в p-область и дырок в n-область. В процессе рекомбинации неосновных носителей в *p-n* переходе энергия выделяется в виде фотонов, т.е. процесс рекомбинации сопровождается световым излучением, частота которого пропорциональна энергии запрещенной зоны полупроводникового материала. Если ширина запрещенной зоны больше 1,8 эВ, то излучение видимое (длина волны меньше 700 нм), если меньше, то излучение невидимое и находится в инфракрасной части спектра.

Полупроводниковые знакосинтезирующие индикаторы – это низковольтные приборы, удобно совмещаемые с источниками питания и уровнями токов микросхем. Они миниатюрны и позволяют конструировать устройства, предназначенные для отображения информации различной сложности - от светящейся точки до текстов и графиков. ППЗСИ обладают малым временем переключения - менее 50 нс. Приборы характеризуются относительно высокими уровнями рабочих токов и умеренными уровнями яркости.

Основные материалы, используемые для изготовления светодиодов, - твердые растворы арсенида и фосфида галлия. Цвет видимого свечения: красный, желтый, зеленый.

В последние годы перспективным методом получения индикаторов с различным цветом свечения считается нанесение люминофорного покрытия непосредственно на кристалл с инфракрасным излучением. Такая конструкция позволяет преобразовать интенсивное инфракрасное излучение кристалла в видимый свет цветового люминофора, что существенно расширяет возможности практического применения этих индикаторов в технике и быту.

В простейшем случае единичный ППЗСИ представляет собой плоский p-n переход с омическими контактами (рисунок 5.11). Однако такая

конструкция недостаточно эффективна из-за внутреннего отражения генерируемого света на границе полупроводник - воздух.

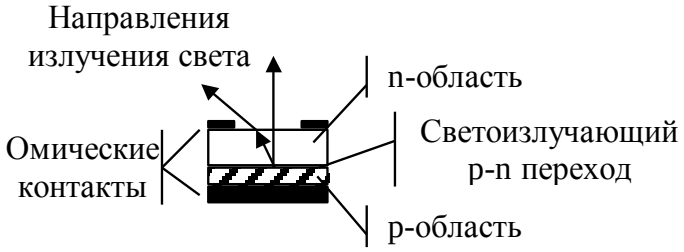

Рисунок 5.11 – Конструкция ППЗСИ с плоским p-n переходом

Для уменьшения отражения на поверхности полупроводника формируют полусферическое покрытие из прозрачного материала, коэффициент преломления которого имеет промежуточное значение между коэффициентами преломления воздуха и кристалла (рисунок 5.12).

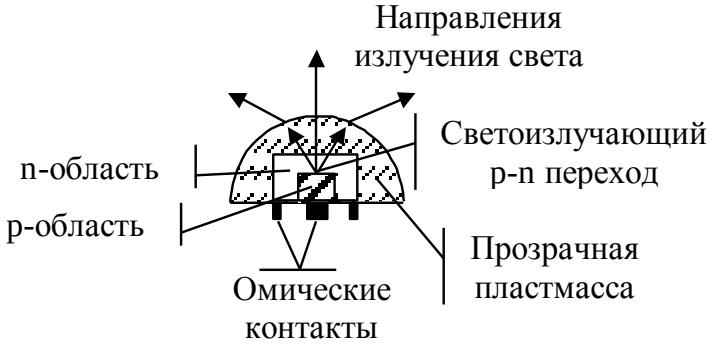

Рисунок 5.12 – Конструкция ППЗСИ с полусферическим покрытием

Эффективны конструкции единичных ППЗСИ, у которых n-область p-n перехода имеет форму полусферы (рисунок 5.13). В такой конструкции лучи генерируемого света подходят к разделу полупроводник-воздух практически перпендикулярно, что резко снижает потери на внутреннее отражение.

Рисунок 5.13 – Конструкция ППЗСИ с полусферической n областью

Размеры излучающих поверхностей единичных ППЗСИ малы, поэтому для увеличения размеров изображения в конструкциях индикаторов используются линзы, рефлекторы и другие устройства, увеличивающие видимый размер светящейся поверхности.

Кристалл единичного ППЗСИ с управляемым цветом свечения имеет два p-n перехода. Один из них излучает красный свет, другой излучает зеленый свет. При включении одного из p-n переходов диод излучает красный или зеленый свет, а при включении обоих p-n переходов благодаря оптической прозрачности фосфида галлия можно получить желтый или оранжевый цвет свечения в зависимости от соотношения токов через p-n переходы.

Единичные ППЗСИ обладают достаточно высоким быстродействием, однако для устройств отображения, в которых они обычно используются, временные параметры не являются критичными.

Светодиоды как элементы индикации обладают рядом достоинств:

- малые габаритные размеры;
- низкое напряжение питания;
- набор различных цветов свечения;
- устойчивость к механическим воздействиям;
- большой срок службы.

Многоэлементные ППЗСИ

Это прибор, состоящий из полупроводниковых излучающих элементов, предназначенный для представления информации в виде знаков и организованный в один или несколько разрядов. В настоящее время выпускается несколько сотен типов многоэлементных полупроводниковых индикаторов, в том числе знаковые, модули шкалы, модули экрана. Они

различаются числом, размерами и конфигурацией светоизлучающих элементов, цветом свечения, конструктивными решениями.

Знаковые индикаторы в основном предназначены для отображения информации в виде цифр и букв. На рабочем поле может одновременно отображаться одно знакоместо (одноразрядный индикатор) или несколько знакомест (многоразрядный). По числу элементов и их взаимному расположению в пределах поля одного разряда различают четыре типа знаковых индикаторов (рисунок 5.14):

I. - семисегментный

II. – девятисегментный;

III. – 35 сегментный матричный ППЗСИ;

IV. – пятисегментный, разных видов

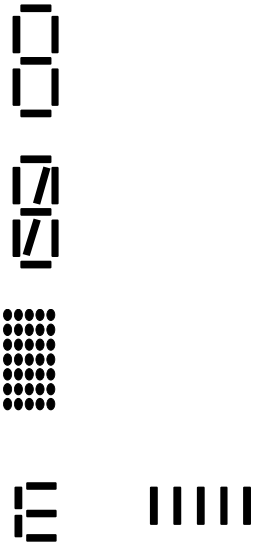

Рисунок 5.14 – Типы знаковых индикаторов

Индикаторы I типа могут быть цифровыми, буквенно-цифровыми и обеспечивают отображение всего ряда цифр с приемлемым для восприятия эстетическим качеством.

Индикаторы II типа позволяют отображать цифры и ограниченный набор букв русского и латинского алфавитов.

Наиболее универсальны индикаторы III типа. При работе с этими индикаторами можно изменять начертание отдельных символов.

Индикаторы IV типа дополняют группу индикаторов II типа и предназначены для отображения символов полярности и переполнения в цифровых устройствах с неполным числом разрядов. Другой вид используется для создания шкал.

С электрической точки зрения, знаковые индикаторы представляют наборы излучающих диодов. Для уменьшения числа контактных выводов индикатора у диодов аноды или катоды объединены вместе (рисунок 5.15).

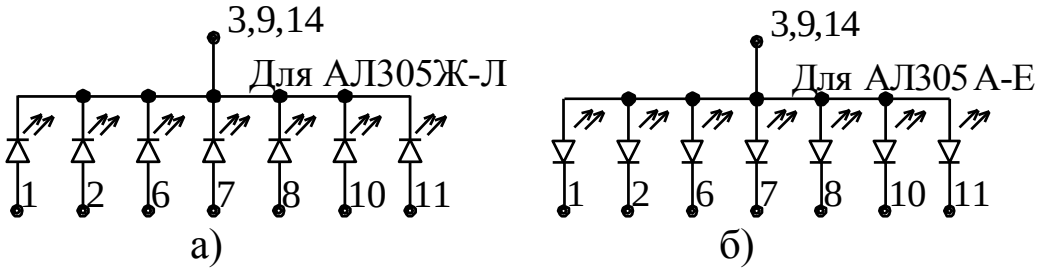

Рисунок 5.15 – Электрические схемы многоэлементных ППЗСИ:
а – общие катоды; б – общие аноды

Для активации 5, 7 или 9 сегментного индикатора при структуре с общим анодом (рисунок 5.15 б) достаточно на объединенный анод всех диодов подать положительное напряжение (около 2 В). На катоды нужных диодов подается потенциал земли. Выбранные диоды зажигаются за счет прямо протекающего через них тока. Катоды неиспользуемых диодов можно подключить к положительному напряжению или оставить неподключенными.

Для структуры с общим катодом (рисунок 5.15, а) на объединенный катод подается потенциал земли. На аноды нужных диодов подается положительное напряжение.

Начертание цифр, синтезируемых на 7-сегментном индикаторе, вполне приемлемо для четкого распознавания в бытовых устройствах (рисунок 5.16).

Рисунок 5.16 – Индикация цифр на семисегментном индикаторе

Девятисегментный индикатор позволяет улучшить форму цифр и даже букв за счет более правильного начертания знаков (рисунок 5.17).

Рисунок 5.17 – Индикация цифр на девятисегментном индикаторе

В 35 сегментном матричном ППЗС индикаторе имеется 35 светоизлучающих диодов, выполненных в виде точечных излучателей, собранных в форме матрицы 5*7 точек. Очень трудно напрямую на выход инди-

катора подключить выводы 35 анодов и катодов. Прямой вывод значительно увеличивает габариты индикатора и знакоместа. Поэтому такие индикаторы организованы по матричной схеме (рисунок 5.18). Светодиоды включены на пересечении строк и столбцов матрицы.

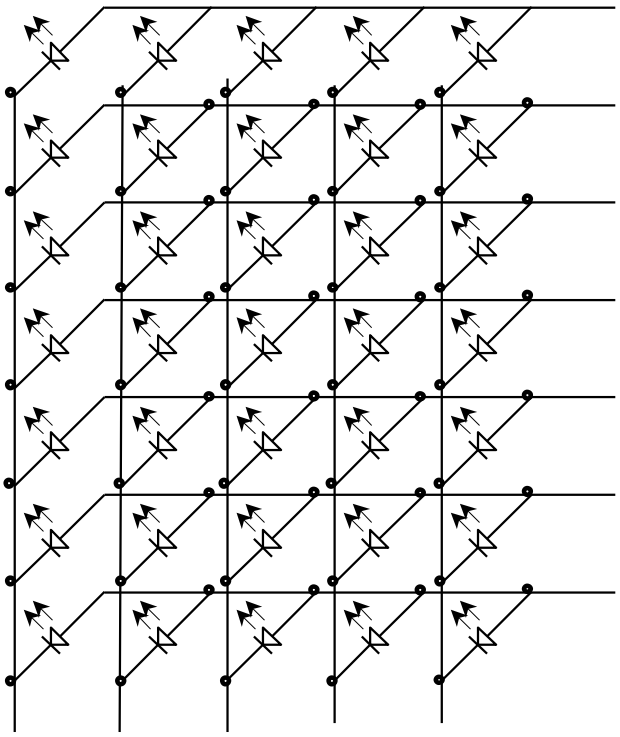

Рисунок 5.18 – Структура 35 сегментного индикатора

В такой схеме для активизации диода (зажигания точки) на вход анодов в нужной строке подается положительное напряжение, а столбец, к которому подключен катод нужного диода, должен быть заземлен. При протекании прямого тока высвечивается точка. При этом можно активизировать нужные диоды всего столбца. Для синтеза знака необходим последовательный перебор всех столбцов знаковой матрицы. Таким образом, диаграмма сигналов на выводах столбцов, имеет следующий вид (рисунок 5.19).

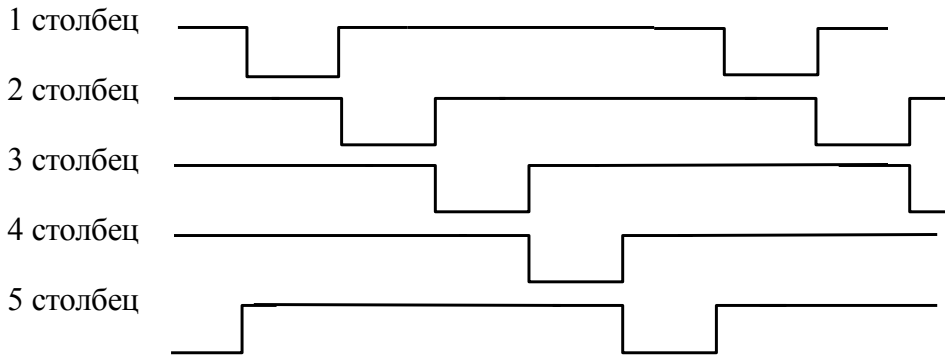

Рисунок 5.19 – Диаграмма напряжений на столбцах индикатора

На строки матричного индикатора подается 7-ми битный код, содержащий единицы в разрядах нужных строк.

Начертания цифр и букв на 35-сегментном матричном индикаторе очень правильные и отлично воспринимаются пользователем (рисунок 5.20).

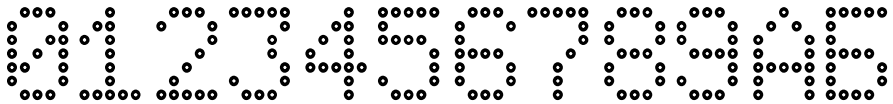

Рисунок 5.20 – Индикация цифр и букв на матричном индикаторе

С помощью 35-сегментного индикатора можно синтезировать буквы русского и латинского алфавитов, цифры, специальные знаки (точки, двоеточие, запятые, многоточия, проценты, градусы и т.п.) и многие другие знаки.

Кроме отображения цифр, букв и специальных знаков, полупроводниковые индикаторы с успехом используются в устройствах индикации включения готовности к работе, наличия напряжения в блоке, нормальной работоспособности узла, аварийной ситуации, достижения температурного порога, выполнения функционального задания и в других устройствах, хорошо согласуясь по электрическим параметрам с полупроводниковыми приборами и микросхемами.

Полупроводниковые индикаторы имеют очень высокую долговечность. В технической документации на индикаторы, как правило, указывается минимальная наработка в 15000 часов, а в облегченных режимах – 30000 ч. При этом продолжительность наработки оценивается по некоторым условным величинам, называемым критериями наработки. Например, принимается, что яркость свечения индикатора должна уменьшиться не более чем на 30% от первоначального значения.

Опыт показывает, что ППЗСИ работают 70…80 и более тысяч часов (около 8..10 лет непрерывной работы).

Эксплуатационные достоинства ППЗСИ способствовали достаточно быстрому и широкому распространению их в радиоэлектронной, вычислительной и другой аппаратуре, выполненной на интегральных микросхемах в качестве:

- визуальных индикаторов состояния полупроводниковых, интегральных и гибридных микросхем;
- источника видимого света для малогабаритных информационных быстродействующих табло в вычислительной технике и устройствах автоматики;
- источника видимого света в малогабаритных матрицах и алфавитно-цифровых модулях;
- малоинерционных излучателей света в быстродействующих системах записи информации на фотопленку;
- источника импульсного света в цепях контроля скоростных фотоэлектронных умножителей;
- элементов индикации в приборных щитах самолетов и космических кораблей;
- цифровых и знаковых индикаторов в устройствах сигнализации и автоматики;
- индикаторов отсчета времени в электронных часах и т.д.

5.2.7 Электронная бумага, электронные чернила

Электронная бумага (также электронные чернила) – технология отображения информации, разработанная для имитации обычной печати на бумаге и основанная на явлении электрофореза. В отличие от традиционных плоских жидкокристаллических дисплеев, электронная бумага формирует изображение в отражённом свете как обычная бумага и может хранить изображение текста и графики в течение достаточно длительного времени, не потребляя при этом электрической энергии и затрачивая её только на изменение изображения. В отличие от традиционной бумаги, технология позволяет произвольно изменять записанное изображение.

Изображение на электронной бумаге формируется аналогично письму на обычной бумаге карандашом — твёрдыми пигментными частицами в микроструктурном материале, дисперсно рассеивающем свет подобно волокнам бумаги, из-за чего угол обзора получается практически

такой же, как и у обычной бумаги и намного превосходит угол обзора плоских жидкокристаллических дисплеев. Электронная бумага также является устройством светомодулирующего типа и работает в чистом виде в отражённом свете без промежуточных преобразований светового потока – как обычный лист с печатным текстом или изображением. Вследствие этого достигается высокая яркость и контрастность получаемого изображения. Эффект памяти обеспечивается удержанием пигментных частиц на поверхности твёрдого тела (подложки) силами Ван-дер-Ваальса.

Принцип действия электронной бумаги

Электронная бумага была впервые разработана в Исследовательском Центре компании Xerox в Пало Альто в 1970-х годах. Первая электронная бумага состояла из полиэтиленовых сфер от 20 до 100 мкм в диаметре. Каждая сфера состояла из отрицательно заряженной чёрной и положительно заряженной белой половины. Все сферы помещались в прозрачный силиконовый лист, который заполнялся маслом, чтобы сферы свободно вращались. Полярность подаваемого напряжения на каждую пару электродов определяла, какой стороной повернется сфера, давая, таким образом, белый или чёрный цвет точки на дисплее.

В 1990-х годах Джозеф Якобсон изобрел другой тип электронной бумаги. Впоследствии он основал корпорацию E Ink Corporation, которая, совместно с Philips разработала и вывела эту технологию на рынок.

Принцип действия был следующий: в микрокапсулы, заполненные окрашенным маслом, помещались электрически заряженные белые частички. В ранних версиях низлежащая проводка управляла тем, будут ли белые частички вверху капсулы (чтобы она была белой для того, кто смотрит) или внизу (смотрящий увидит цвет масла). Это было фактически повторное использование уже хорошо знакомой электрофоретической технологии отображения, но использование капсул позволило сделать дисплей с использованием гибких пластиковых листов вместо стекла.

В настоящее время принцип действия электронной бумаги очень прост — между двумя листами-электродами (верхний прозрачный) находится специальная жидкость (рисунок 5.21) в которую помещены белые и черные частички выполняющие роль пикселов. Белые частички имеют положительный заряд, а черные - отрицательный заряд. При пропускании через электроды тока, частички всплывают или наоборот погружаются, в

зависимости от заряда. Таким образом, энергия расходуется только при перерисовке изображения, а не постоянно, как в случае с TFT-экранами. К тому же, для чтения информации достаточно света извне и подсветка не требуется.

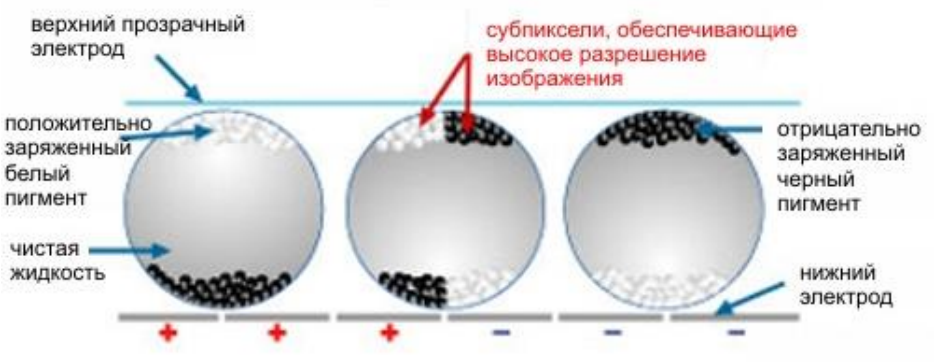

Рисунок 5.21 – Микрокапсулы - пиксели

Многоцветная (полихромная) электронная бумага

В последнее десятиление была разработана многоцветная электронная бумага, в которой применяются светофильтры.

Строение и принцип действия многоцветной электронной бумаги, использующей светофильтры, поясняется на рисунке 5.22.

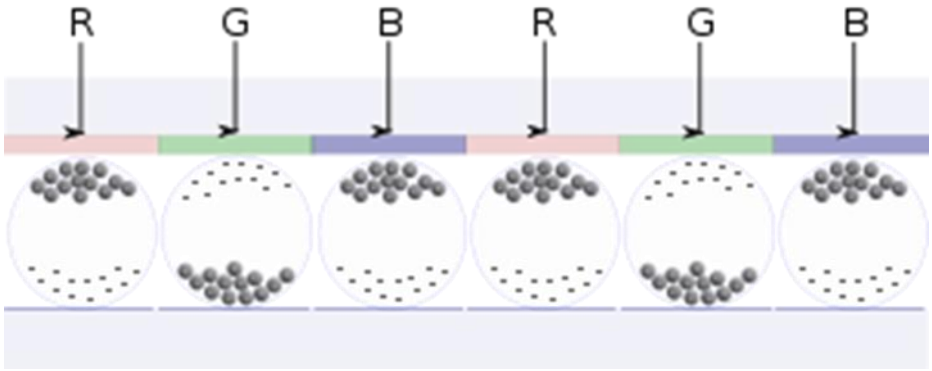

Рисунок 5.22 – Строение многоцветной электронной бумаги

Обычно цветная электронная бумага состоит из тонких окрашенных оптических фильтров, которые добавляются к монохромному дисплею, описанному выше. Множество точек разбито на триады, как правило, состоящие из трёх стандартных цветов CMYK (циановый, пурпурный и жёлтый). В этом случае цвета формируются методом вычитания, как и в

полиграфии. Данная технология уже широко применяется в электронных книгах (ридерах).

Преимущества и недостатки

Преимуществом можно назвать большее время автономной работы, которое отличается в лучшую сторону по сравнению с прочими электронными устройствами с дисплеями. Экран на основе электронной бумаги потребляет энергию только при изменении отображаемой информации (например, перелистывании страниц), тогда как типичный ЖК экран потребляет энергию постоянно. Также преимуществом следует отметить высокую яркость и контрастность получаемого изображения.

Недостатком является то, что в настоящее время дисплеи на основе электронной бумаги имеют очень большое (порядка 200 мс) время обновления по сравнению с ЖК-дисплеями. Это не позволяет использовать сложные интерактивные элементы интерфейса (анимированные меню, указатели мыши, скроллинг). Сильнее всего это сказывается на способности электронной бумаги показывать увеличенный фрагмент большого текста или изображения на маленьком экране.

Ещё одним недостатком этой технологии является подверженность экрана механическим повреждениям в некоторых модификациях таких экранов. В части дисплеев имеется подложка из очень тонкого хрупкого стекла. Однако уже используется технология, в которой стеклянная подложка заменена пластиковой и такие экраны можно даже немного изгибать. Такие экраны гораздо менее подвержены разрушениям от ударов и деформаций.

Применение электронной бумаги

Электронная бумага легка, надёжна, а дисплеи на её основе могут быть гибкими (хотя и не настолько, как обычная бумага). Предполагаемое применение включает электронные книги, которые могут хранить цифровые версии многих литературных произведений, электронные вывески, наружную и внутреннюю рекламу.

В настоящее время для электронной бумаги можно отметить применение в следующих случаях:

- электронные книги;
- электронные газеты;

- дисплеи для телефонов и графических планшетов;
- уличные плакаты и объявления;
- электронные ценники в магазинах;
- цифровые номера на автомобилях и других.

5.2.8 Другие типы индикаторов

Описанные выше типы индикаторов широко применяются в различных радиоэлектронных устройствах бытового назначения и в технических приложениях. Основным критерием их преимущественного распространения являются высокие эксплуатационные качества по потребляемой мощности, надежности, долговечности, стоимости и простоте схем управления.

Для полноты картины перечислим некоторые другие типы индикаторов, не вошедшие в перечисленные выше типы:

- накальные индикаторы – лампы;
- накальные вакуумные индикаторы;
- накальные индикаторы на флуоресцирующих стеклах;
- газоразрядные неоновые лампы;
- газовые электронно-световые индикаторы;
- тиратроны тлеющего разряда;
- газоразрядные знаковые индикаторы;
- газоразрядные индикаторные панели;
- электронно - лучевые индикаторы;
- пневматические индикаторы;
- термоиндикаторы и т.д.

5.3 Ввод информации в микро-ЭВМ

При работе с радиоэлектронной аппаратурой пользователь выражает свои желания с помощью устройства ввода. В качестве устройства ввода выступает клавиатура. Клавиатура – это набор коммутационных элементов – клавиш, при воздействии на которые формируются электрические сигналы.

Принципы, заложенные в реализацию таких элементов, могут быть различными. По виду воздействия можно выделить клавиши нажатия (тактильные) и прикосновения (сенсорные). Различают также контактные и бесконтактные клавишные устройства.

В контактных элементах механическое воздействие на клавишу приводит к переключению электромеханического контакта. Такие устройства отличаются низкой стоимостью, простотой конструкции, возможностью коммутации довольно большой мощности, огромным числом допустимых переключений.

Улучшенным вариантом электромеханических контактов являются герконы – герметизированные контакты, располагаемые в герметичном стеклянном баллоне. Срабатывание происходит при воздействии на контакты магнитного поля от маленького магнита или электромагнитной катушки.

В бесконтактных клавишах используются переключатели, управляемые магнитным полем, световым излучением, или воздействием от индукционных датчиков. К бесконтактным относятся и сенсорные переключатели.

При малом числе клавиш пару проводов от каждой клавиши можно завести прямо на устройство управления. При увеличении числа клавиш до десяти и более, значительно возрастает число проводов. Поэтому клавиши подключают по матричной схеме (рисунок 5.23).

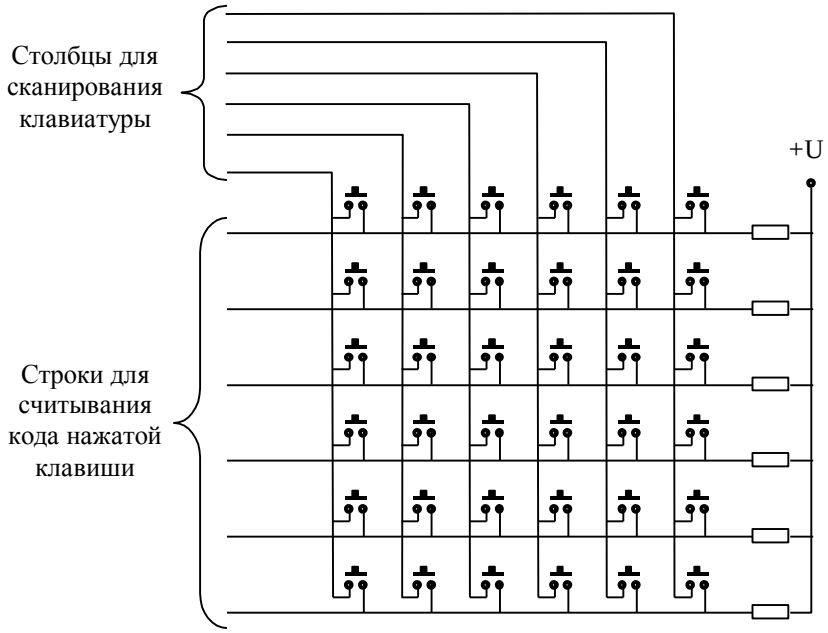

Рисунок 5.23 – Структура матричной клавиатуры

При матричной схеме организации клавиатуры процесс работы состоит из фазы вывода на столбцы кода сканирования и фазы считывания кода со строк. Код сканирования состоит из всех логических единиц и одного логического нуля. Нулевой потенциал в каждый момент времени

подается только на один столбец. Если столбец, подключенный к нулевому значению, не содержит нажатых клавиш, то на всех строках считывается логическая единица. Устройство обслуживания клавиатуры в этом случае считает, что данный столбец опрошен и нажатых клавиш нет.

Далее, логический нуль смещается на одну позицию, для активизации следующего столбца. Процесс считывания строк повторяется. Таким образом, в процессе перебора всех столбцов (сканирование столбцов), может быть обнаружена нажатая клавиша. Номер столбца определяется по позиции логического нуля, а строка с нажатой клавишей также будет содержать нулевое значение.

Адрес клавиши (столбец, строка) однозначно определяет ее позицию и, таким образом, позволяет определить ее значение (функцию). В устройство управления, работающее с клавиатурой, обычно закладывается функция устранения дребезга контактов (антидребезг). Дребезг контактов объясняется тем, что механические контакты при включении замыкаются не сразу. За короткое время контактное сопротивление изменяется скачкообразно несколько раз. Контакты как бы ударяются и отскакивают друг от друга, как мячики. Антидребезг срабатывает при повторном считывании кода клавиши через небольшое время, порядка $20 \cdot 10^{-3}$ секунды. За это время дребезжание заканчивается, и устройство фиксирует нажатие клавиши.

Еще одной проблемой является одновременное нажатие нескольких клавиш. В этом случае клавиатура обычно блокируется на небольшое время, и опрос повторяется с задержкой. Блокировка клавиатуры будет повторяться либо до момента отпускания всех клавиш, либо до фиксации всего одной нажатой клавиши. В любом случае неправильный код с клавиатуры не обрабатывается. Так гарантируется правильность работы всего устройства.

В случае использования микро-ЭВМ, имеющей встроенный Аналого-цифровой преобразователь, можно организовать клавиатуру по схеме с делителями напряжения (рисунок 5.24).

На аналоговый вход АЦП In_ADC подключается цепочка делителей напряжения R. Каждая нажатая клавиша заземляет часть цепочки делителей. На входе АЦП при этом изменяется входное напряжение. Таким образом, код, вычисленный на АЦП, однозначно определяет номер нажатой клавиши. Количество клавиш при 8 битовом АЦП не может превышать 255.

Достоинствами такой клавиатуры следует отметить малое число задействованных входов микро-ЭВМ, простоту алгоритма вычисления но-

мера нажатой клавиши. Устранение дребезга происходит при повторных запусках АЦП. Достаточно несколько раз подряд определить нажатие одной и той же клавиши и можно считать ее нажатой. Отсутствие нажатых клавиш определяется очень легко по значению кода АЦП.

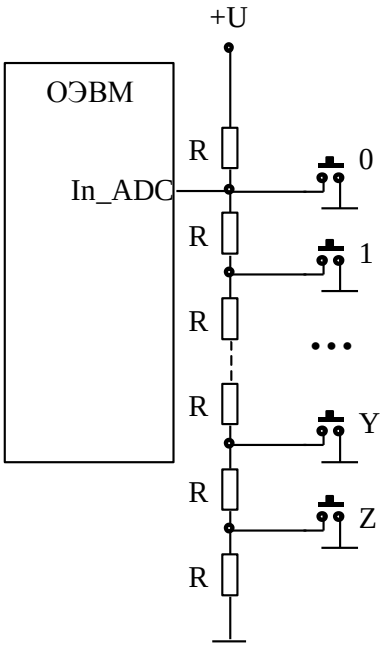

Рисунок 5.24 – Клавиатура с делителями напряжения

В такой клавиатуре просто решается вопрос обработки нескольких, одновременно нажатых, клавиш. В этом случае будет определен код наиболее близкой ко входу АЦП клавиши.

5.4 Контрольные вопросы

1. Что такое ВЛИ? Объясните принцип их работы.
2. Какие физические эффекты используются при работе ЖКИ?
3. Почему для питания ЖКИ необходимо использовать двухфазное питание?
4. Расшифруйте аббревиатуру ППЗСИ. Объясните принцип работы.
5. Какой ресурс работы у ВЛИ, ЖКИ и ППЗСИ?
6. Как работает электронная бумага и где ее применяют?
7. Какие достоинства и недостатки у электронной бумаги?
8. Что такое матричная клавиатура?
9. Опишите принцип работы и достоинства клавиатуры с использованием АЦП.

6 СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Гребешков А.Ю. Вычислительная техника, сети и телекоммуникации: Учебное пособие. — Москва : Горячая линия-Телеком, 2015. — 190 с.

2. Магда Ю.С. Микроконтроллеры серии 8051: практический подход. — Москва : ДМК Пресс, 2010. — 228 с.

3. Грошева Л.С. Архитектура микроконтроллеров MCS-51. — Нижний Новгород : ВГУВТ, 2014. — 68 с.

4. Лошаков С. Периферийные устройства вычислительной техники: учебное пособие / С. Лошаков. — Москва : , 2016. — 435 с.

5. Электронные вычислительные машины : практическое пособие для вузов в 8 кн. / ред. А. Я. Савельев. - М. : Высшая школа, 1987 - . Кн. 1 : Введение в ЭВМ : учебное пособие / А. Я. Савельев [и др.]. - М. : Высшая школа, 1987. - 159 с.

6. Максимов Н.В. Архитектура ЭВМ и вычислительных систем: учебник / Н. В. Максимов, Т. Л. Партыка, И. И. Попов. - 4-е изд., перераб. и доп. - М. : ФОРУМ, 2012. - 511 с.

7. Фрунзе А.В. Микроконтроллеры? Это же просто! Т. 1. — Москва : ДМК Пресс, 2010. — 311 с.

8. Матюшин А.О. Программирование микроконтроллеров: стратегия и тактика. — Москва : ДМК Пресс, 2017. — 356 с.

9. Хартов В.Я. Микропроцессорные системы : учебное пособие для вузов / В. Я. Хартов. - М. : Академия, 2010. - 352 с.

10. Бикташев Р.А. Введение в вычислительную технику: учебное пособие / Р.А. Бикташев, Л.И. Федосеева. — Пенза : ПензГТУ, 2012. — 116 с.

11. Чернышов А.В. Инструментальные средства программирования и их применение в современной вычислительной технике. — Москва : МГТУ им. Н.Э. Баумана, 2010. — 192 с.

12. Белов А.В. Создаем устройства на микроконтроллерах. — Санкт-Петербург : Наука и Техника, 2007. — 304 с.

13. Кузяков О.Н. Проектирование систем на микропроцессорах. — Тюмень : ТюмГНГУ, 2014. — 104 с.

14. Однокристальный микроконтроллер семейства MCS-51 фирмы INTEL 8XC51GB.– Томск: ТОО «SDD», 1995.– 126 с.

ПРИЛОЖЕНИЕ А
(СПРАВОЧНОЕ)
СПИСОК КОМАНД ОЭВМ КМ1816ВЕ51

Мнемокод	Название	Байты	Циклы	Операция
Группа команд передачи данных				
MOV @Ri,#d	Пересылка в РПД константы (i=0,1)	2	1	**((Ri))←#d**
MOV @Ri,A	Пересылка в РПД байта из аккумулятора	1	1	**((Ri))←(A)**
MOV @Ri,ad	Пересылка в РПД байта с прям.адреса ad	2	2	**((Ri))←(ad)**
MOV A,#d	Загрузка в аккумулятор константы	2	1	**(A)←#d**
MOV A,@Ri	Пересылка в аккумулятор из РПД (i=0,1)	1	1	**(A)←((Ri))**
MOV A,ad	Пересылка в аккумулятор байта с адреса ad	2	1	**(A)←(ad)**
MOV A,Rn	Пересылка в аккумулятор байта из регистра (n=0÷7)	1	1	**(A)←(Rn)**
MOV ad,#d	Пересылка константы по прямом.адресу	3	2	**(ad)←#d**
MOV ad,@Ri	Пересылка по прямому адресу байта из РПД (i=0,1)	2	2	**(ad)←((Ri))**
MOV ad,A	Пересылка аккумулятора по прямому адресу	2	1	**(ad)←(A)**
MOV ad,Rn	Пересылка регистра по прямому адресу	2	2	**(ad)←(Rn)**
MOV add,ads	Пересылка прямоадресуемого байта по прямому адресу	3	2	**(add)←(ads)**
MOV DPTR,#d16	Загрузка указателя данных словом	3	2	**(DPTR)←#d16**
MOV Rn,#d	Загрузка в регистр (n=0÷7) константы	2	1	**(Rn)←#d**

Мнемокод	Название	Байты	Циклы	Операция
MOV Rn,A	Пересылка в регистр (n=0÷7) байта из аккумулятора	1	1	**(Rn)←(A)**
MOV Rn,ad	Пересылка в регистр с прямого адреса	2	2	**(Rn)←(ad)**
MOVC A,@A+DPTR	Пересылка в аккумулятор байта из ПП	1	2	**(A)←((A)+(DPTR))**
MOVC A,@A+PC	Пересылка в аккумулятор байта из ПП	1	2	**(A)←((A)+(PC))**
MOVX @DPTR,A	Пересылка в расшир. ВПД из аккумулятора	1	2	**((DPTR))←(A)**
MOVX A,@DPTR	Пересылка в аккумулятор из расшир. ВПД	1	2	**(A)←((DPTR))**
MOVX @Ri,A	Пересылка в ВПД из аккумулятора	1	2	**((Ri))←(A)**
MOVX A,@Ri	Пересылка в аккумулятор байта из ВПД	1	2	**(A)←((Ri))**
POP ad	Извлечь из стека	2	2	**(ad)←((SP))** **(SP)←(SP)-1**
PUSH ad	Загрузить в стек	2	2	**(SP)←(SP)+1** **((SP))←(ad)**
XCH A,@Ri	Обмен аккумулятора с байтом из РПД	1	1	**(A)↔((Ri))**
XCH A,ad	Обмен аккумулятора и байта с адреса ad	2	1	**(A)↔(ad)**
XCH A,Rn	Обмен аккумулятора с регистром	1	1	**(A)↔(Rn)**
XCHD A,@Ri	Обменять младш. тетраду аккумулятора с мл. тетрадой байта РПД	1	1	**$(A_{0-3})\leftrightarrow((Ri)_{0-3})$**
SWAP A	Обменять тетрады в аккумуляторе	1	1	**$(A_{0-3})\leftrightarrow(A_{4-7})$**

Мнемокод	Название	Байты	Циклы	Операция
Группа арифметических операций				
ADD A,#d	Сложить аккумулятор с константой	2	1	**(A)←(A)+#d**
ADD A,@Ri	Сложить аккумулятор с байтом из РПД	1	1	**(A)←(A)+((Ri))**
ADD A,ad	Сложить аккумулятор с байтом по адресу ad	2	1	**(A)←(A)+(ad)**
ADD A,Rn	Сложить аккумулятор с регистром	1	1	**(A)←(A)+(Rn)**
ADDC A,#d	Сложить аккумулятор с константой и переносом	2	1	**(A)←(A)+#d+(C)**
ADDC A,@Ri	Сложить аккумулятор с байтом из РПД (i=0,1) и переносом	1	1	**(A)←(A)+((Ri))+(C)**
ADDC A,ad	Сложить аккумулятор с байтом по адресу ad и переносом	2	1	**(A)←(A)+(ad)+(C)**
ADDC A,Rn	Сложить аккумулятор с регистром (n=0÷7) и переносом	1	1	**(A)←(A)+(Rn)+(C)**
DA A	Десятичная коррекция аккумулятора	1	1	
SUBB A,#d	Вычесть из аккумулятора константу и заем	2	1	**(A)←(A)-#d-(C)**
SUBB A,@Ri	Вычесть из аккумулятора байт из РПД и заем	1	1	**(A)←(A)-((Ri))-(C)**
SUBB A,ad	Вычесть из аккумулятора байт с адреса ad и заем	2	1	**(A)←(A)-(ad)-(C)**
SUBB A,Rn	Вычесть из аккумулятора регистр и заем	1	1	**(A)←(A)-(Rn)-(C)**
INC @Ri	Инкремент байта в РПД	1	1	**((Ri))←((Ri))+1**
INC A	Инкремент аккумулятора	1	1	**(A)←(A)+1**
INC ad	Инкремент байта по адр. ad	2	1	**(ad)←(ad)+1**

Мнемокод	Название	Байты	Циклы	Операция
INC DPTR	Инкремент указателя данных	1	2	**(DPTR)←(DPTR)+1**
INC Rn	Инкремент регистра	1	1	**(Rn)←(Rn)+1**
DEC @Ri	Декремент байта в РПД	1	1	**((Ri))←((Ri))-1**
DEC A	Декремент аккумулятора	1	1	**(A)←(A)-1**
DEC ad	Декремент байта по адресу ad	2	1	**(ad)←(ad)-1**
DEC Rn	Декремент регистра	1	1	**(Rn)←(Rn)-1**
MUL AB	Умножение аккумулятора на регистр B	1	4	**(B)(A)←(A)·(B)**
DIV AB	Деление аккумулятора на регистр B	1	4	**(A).(B)←(A)/(B)**
Группа логических операций				
ANL A,#d	Логическое И аккумулятора и константы	2	1	**(A)←(A)∧#d**
ANL A,@Ri	Логическое И аккумулятора с байтом из РПД (i=0,1)	1	1	**(A)←(A)∧((Ri))**
ANL A,ad	Логическое И аккумулятора с байтом по адресу ad	2	1	**(A)←(A)∧(ad)**
ANL A,Rn	Логическое И аккумулятора с регистром (n=0÷7)	1	1	**(A)←(A)∧(Rn)**
ANL ad,#d	Логическое И байта по адресу ad с константой	3	2	**(ad)←(ad)∧#d**
ANL ad,A	Логическое И байта по адресу ad с аккумулятором	2	1	**(ad)←(ad)∧(A)**
ORL A,#d	Логическое ИЛИ аккумулятора с константой	2	1	**(A)←(A)∨#d**
ORL A,@Ri	Логическое ИЛИ аккумулятора с байтом из РПД (i=0,1)	1	1	**(A)←(A)∨((Ri))**
ORL A,ad	Логическое ИЛИ аккумулятора с байтом по адресу ad	2	1	**(A)←(A)∨(ad)**
ORL A,Rn	Логическое ИЛИ аккумулятора с регистром (n=0÷7)	1	1	**(A)←(A)∨(Rn)**

Мнемокод	Название	Байты	Циклы	Операция
ORL ad,#d	Логическое ИЛИ байта по адресу ad с константой	3	2	**(ad)←(ad)∨#d**
ORL ad,A	Логическое ИЛИ байта по адресу ad с аккумулятором	2	1	**(ad)←(ad)∨(A)**
XRL A,#d	Исключ. ИЛИ аккумулятора с константой	2	1	**(A)←(A)⊕#d**
XRL A,@Ri	Исключающее ИЛИ аккумулятора с байтом из РПД (i=0,1)	**1**	**1**	**(A)←(A)⊕((Ri))**
XRL A,ad	Исключающее ИЛИ аккумулятора с байтом по адресу ad	2	1	**(A)←(A)⊕(ad)**
XRL A,Rn	Исключающее ИЛИ аккумулятора с регистром (n=0÷7)	1	1	**(A)←(A)⊕(Rn)**
XRL ad,#d	Исключающее ИЛИ байта по адресу ad с константой	3	2	**(ad)←(ad)⊕#d**
XRL ad,A	Исключающее ИЛИ байта по адресу ad с аккумулятором	2	1	**(ad)←(ad)⊕(A)**
CLR A	Сброс аккумулятора	1	1	**(A)←0**
CPL A	Инверсия аккумулятора	1	1	**(A)←($\overline{A}$)**
RL A	Сдвиг аккумулятора влево по циклу	1	1	**(A_{n+1})←(A_n), n=0÷6** **(A_0)←(A_7)**
RR A	Сдвиг аккумулятора вправо по циклу	1	1	**(A_n)←(A_{n+1}), n=0÷6** **(A_7)←(A_0)**
RLC A	Сдвиг аккумулятора влево через перенос	1	1	**(A_{n+1})←(A_n), n=0÷6** **(A_0)←(C)** **(C)←(A7)**
RRC A	Сдвиг аккумулятора вправо через перенос	1	1	**(A_n)←(A_{n+1}), n=0÷6** **(A_7)←(C) (C)←(A_0)**
Группа битовых операций				
MOV bit,C	Пересылка разряда переноса в бит	2	2	**(b)←(C)**

Мнемокод	Название	Байты	Циклы	Операция
MOV C,bit	Пересылка бита в разряд переноса	2	1	**(C)←(b)**
ANL C,bit	Логическое И бита и переноса	2	2	**(C)←(C)∧(b)**
ANL C,/bit	Логическое И инверсии бита и переноса	2	2	**(C)←(C)∧($\overline{b}$)**
ORL C,bit	Логическое ИЛИ бита и переноса	2	2	**(C)←(C)∨(b)**
ORL C,/bit	Логическое ИЛИ инверсии бита и переноса	2	2	**(C)←(C)∨($\overline{b}$)**
CLR bit	Сброс бита	2	1	**(b)←0**
CLR C	Сброс переноса	1	1	**(C)←0**
SETB bit	Установка бита	2	1	**(b)←1**
SETB C	Установка переноса	1	1	**(C)←1**
CPL bit	Инверсия бита	2	1	**(b)←($\overline{b}$)**
CPL C	Инверсия переноса	1	1	**(C)←($\overline{C}$)**
Группа команд передачи управления				
LCALL ad16	Длинный вызов программы	3	2	**(PC)←(PC)+3, (SP)←(SP)+2, (SP-1)←(PC_{0-7}), (SP)←(PC_{8-15}), (PC)←ad16**
ACALL ad11	Абсолютный вызов подпрограммы в странице 2 Кбайта	2	2	**(PC)←(PC)+2, (SP)←(SP)+2, (SP-1)←(PC_{0-7}), (SP)←(PC_{8-15}), (PC_{0-10})←ad11**
LJMP ad16	Длинный переход	3	2	**(PC)←ad16**
AJMP ad11	Абсолютный переход в странице размером 2 Кбайт	2	2	**(PC_{0-10})←ad11**
SJMP rel	Короткий относительный переход	2	2	**(PC)←(PC)+2+rel**

Мнемокод	Название	Байты	Циклы	Операция
CJNE A,ad,rel	Сравнение аккумулятора с байтом по адресу ad и переход, если не равно	3	2	**(PC)←(PC)+3, если (A)≠(ad), то (PC)←(PC)+rel**
CJNE @Ri,#d,rel	Сравнение байта из РПД с константой и переход, если не равно	3	2	**(PC)←(PC)+3, если ((Ri))≠#d, то (PC)←(PC)+rel**
CJNE A,#d,rel	Сравнение аккумулятора с константой и переход, если не равно	3	2	**(PC)←(PC)+3, если (A)≠#d, то (PC)←(PC)+rel**
CJNE Rn,#d,rel	Сравнение регистра с константой и переход, если не равно	3	2	**(PC)←(PC)+3, если (Rn)≠#d, то (PC)←(PC)+rel**
DJNZ ad,rel	Декремент байта по адресу ad и переход, если не нуль	3	2	**(PC)←(PC)+2, (ad)←(ad)-1, если (ad)≠0, то (PC)←(PC)+rel**
DJNZ Rn,rel	Декремент регистра и переход, если не нуль	2	2	**(PC)←(PC)+2, (Rn)←(Rn)-1, если (Rn)≠0, то (PC)←(PC)+rel**
JB bit,rel	Переход, если бит равен единице	3	2	**(PC)←(PC)+3, если (b)=1, то (PC)←(PC)+rel**
JBC bit,rel	Переход, если бит установлен, с последующим сбросом бита	3	2	**(PC)←(PC)+3, если (b)=1, то (b)←0, (PC)←(PC)+rel**
JC rel	Переход, если перенос равен единице	2	2	**(PC)←(PC)+2, если (C)=1, то (PC)←(PC)+rel**
JMP @A+DPTR	Косвенный относительный переход	1	2	**(PC)←(A)+(DPTR)**
JNB bit,rel	Переход, если бит равен нулю	3	2	**(PC)←(PC)+3, если (b)=0, то (PC)←(PC)+rel**

Мнемокод	Название	Байты	Циклы	Операция
JNC rel	Переход, если перенос равен нулю	2	2	**(PC)←(PC)+2, если (C)=0, то (PC)←(PC)+rel**
JNZ rel	Переход, если аккумулятор не равен нулю	2	2	**(PC)←(PC)+2, если (A)≠0, то (PC)←(PC)+rel**
JZ rel	Переход, если аккумулятор равен нулю	2	2	**(PC)←(PC)+2, если (A)=0, то (PC)←(PC)+rel**
RET	Возврат из подпрограммы	1	2	**(PC_{8-15})←((SP)) (PC_{0-7})←((SP)-1) (SP)←(SP)-2**
RETI	Возврат из подпрограммы обработки прерывания	1	2	**(PC_{8-15})←((SP)) (PC_{0-7})←((SP)-1) (SP)←(SP)-2**
NOP	Нет операции	1	1	**(PC)←(PC)+1**