Министерство образования и науки
Российской Федерации
Рубцовский индустриальный институт (филиал)
ФГБОУ ВПО «Алтайский государственный технический
университет им. И.И. Ползунова»

Кафедра электроэнергетики

Н.И. Задоя

ЭЛЕМЕНТЫ ЦИФРОФОЙ АВТОМАТИКИ

Учебное пособие для бакалавров направления
«Электроэнергетика и электротехника»

Рубцовск 2014

УДК 681.31

Задоя Н.И. Элементы цифровой автоматики: Учебное пособие для бакалавров направления «Электроэнергетика и электротехника» / Рубцовский индустриальный институт. – Рубцовск, 2014. – 105 с.

В учебнике рассмотрены основные системы счисления и представление информации в микропроцессорных системах. Приведены классификация и типовая структура микропроцессоров, классификация запоминающих устройств, определение и функции интерфейса микропроцессорных систем. Изложены вопросы классификации команд, способы адресации, языки и средства программирования. В качестве примера подробно рассмотрены однокристальные микроЭВМ.

Пособие предназначено для студентов очной и заочной форм обучения.

Рассмотрено и одобрено
на заседании НМС РИИ.
Протокол №6 от 01.09.14.

Рецензент: профессор, к.ф.- м.н. В.В. Борисовский

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

В настоящее время наблюдается бурное развитие телекоммуникационного сектора экономики. Это стало возможным благодаря либерализации государственного контроля над телекоммуникациями, а также быстро расширяющему применению цифровых технологий в действующих и перспективных системах связи, радиовещания и телевидения.

Такое положение дел связано, прежде всего, с известными преимуществами применения цифровых сигналов: высокой потенциальной помехоустойчивостью, возможностями оптимизации использования частотного спектра, перспективами применения в различных телекоммуникационных и информационных системах универсальных аппаратных и программных решений и т.д.

Одним из ключевых факторов развития в этом направлении выступает технологический прогресс. Растущая производительность микропроцессоров, появление мощных сигнальных процессоров, создание высокоэффективных методов компрессии и транспортировки информации – это только часть списка технологических инноваций, ведущих к ускорению развития информационных цифровых технологий.

Развитие и совершенствование электронно-вычислительной техники в значительной степени определяются возможностями цифровых микросхем. Все узлы цифровых вычислительных машин содержат элементы цифровой техники, с помощью которых осуществляются запоминание и хранение информации, управление вычислительным процессом, ввод и вывод информации.

Сигналы, представленные в цифровой форме, практически не подвержены амплитудным и фазовым искажениям, что позволяет передавать информацию на большие расстояния с сохранением ее высокого качества. Это обусловлено определенными преимуществами цифровых устройств по сравнению с аналоговыми: более высокой надежностью; стабильностью параметров; высокой точностью обработки информации; значительным сокращением трудоемкости и упрощением операций регулировки и настройки; возможностью создания микросхем с очень высокой степенью интеграции.

Настоящее учебное пособие ориентировано на специалистов электротехнического профиля и предназначено для базовой подготовки, которая позволила бы в других дисциплинах рассматривать различные приложения средств вычислительной техники.

Цель учебного пособия – дать основное представление о структуре и функциях персонального компьютера, помочь студентам сориентироваться на рынке технических средств компьютерной техники.

В первой главе рассматриваются системы счисления и преобразование чисел из одной системы в другую.

Во второй главе рассмотрено представление информации в микропроцессорных системах.

Третья глава посвящена изложению классификации микропроцессоров, их типовой структуре, принципов построения и функционирования микропроцессорной системы.

В отдельную главу выделены сведения по оперативным, постоянным и внешним запоминающим устройствам микропроцессорных систем.

В пятой главе изучаются вопросы обмена информацией в микропроцессорной системе и основные внешние устройства ввода - вывода информации.

В шестой главе освещены вопросы программного обеспечения микропроцессоров, классификация команд, способы адресации, языки и средства программирования.

В седьмой главе подробно рассмотрены вопросы состава, структурной организации и работы однокристальных микроЭВМ семейства МК48.

Различным аспектам проблемы анализа и синтеза микропроцессорных систем посвящена обширная литература как отечественная, так и зарубежная, и число публикаций постоянно растет. Поэтому в списке литературы к учебному пособию даны лишь основные (по мнению автора) работы, в которых отдельные вопросы рассматриваются более подробно, чем в предлагаемом учебном пособии, с которыми будет целесообразно ознакомиться особо заинтересовавшемуся читателю.

Глава 1. ОСНОВНЫЕ СИСТЕМЫ СЧИСЛЕНИЯ

1.1. Системы счисления

Системой счисления называют совокупность приемов и правил наименования и обозначения чисел, с помощью которых можно установить однозначное соответствие между любым числом и его представлением в виде совокупности конечного числа символов. Система счисления, в которой величина цифры определяется ее местоположением (позицией), называется *позиционной.*

Общераспространенна *десятичная система счисления*. Для нас наибольший интерес представляет *двоичная система счисления*, в которой основанием является число 2. В этом случае для записи чисел используются два символа, 0 и 1.

Дробное число в двоичной системе счисления представляется в виде:

$A = a_n \cdot 2^{n-1} + a_{n-1} \cdot 2^{n-2} + \ldots + a_1 \cdot 2^0 , a_0 \cdot 2^{-1} + a_{-1} \cdot 2^{-2} + \ldots$

1.2. Преобразование чисел из одной системы в другую

Перевод целой части числа из десятичной системы в двоичную производится методом последовательного деления числа на 2 до тех пор, пока частное от деления не станет равным единице, например:

Таблица 1.1

42	2				
42	21	2			
0	20	10	2		
	1	10	5	2	
		0	4	2	2
			1	2	**1**
				0	

При этом число в двоичной системе счисления записывается в виде остатков от деления, начиная с последнего частного, **справа налево.** В рассмотренном примере: 42 (10) = 101010 (2).

Для перевода дробной части числа последовательно умножаем дробную часть на два.

Двоичное число записывается в виде целых частей чисел, полученных при умножении только дробной части, начиная сверху после запятой. В рассматриваемом примере (0,6875) (10) = 0,1011(2).

Таблица 1.2

0	6875
	X2
1	375
	X2
0	75
	X2
1	5
	X2
1	0

По рассмотренным правилам числа можно переводить и в другие системы счисления, например в *восьмеричную, шестнадцатеричную* и т.д., во всех случаях умножение или деление производится на основании новой системы счисления. Для представления чисел в любой системе счисления с основанием p используется набор из p символов: для p=2 - (0,1), для p=8 - (0,1,2,3,4,5,6,7), для p=10 - (0,...,9), для p=16 - (0,...,9,A,B,C,D,E,F).

Для отличия записи числа в одной системе счисления от другой в конце числа ставится знак – признак системы счисления.

B – двоичная система счисления.

Q – восьмеричная система счисления.

H – шестнадцатеричная система счисления.

Примеры:

101,101B = (4+0+1),(0,5+0+0,125) = 5,625.

AB9, 81H = (10∗256+11∗16+9), (8/16+1/256) = 2745, 50390625.

Нижеприведенная таблица используется для перевода одноразрядных шестнадцатеричных и восьмеричных чисел в двоичные числа.

Правила перевода из восьмеричной и шестнадцатеричной систем в двоичную систему: переводим по порядку все символы – цифры, затем нули слева и справа в записи двоичного числа отбрасываем. Пример:

725,54Q = (111 010 101, 101 100) = 111010101,1011B.

Обратный перевод из двоичной системы:

Для перевода в восьмеричную систему: разбиваем двоичное число на группы по 3 разряда, начиная от запятой вправо и влево, добавляем недостающие нули слева и справа.

Аналогично для перевода из двоичной в шестнадцатеричную разбиваем на группы по 4 разряда. Пример:

1110101101,10111B = (001 110 101 101,101 110) = 1655,56Q.

1110101101,10111B = (0011 1010 1101,1011 1000) = 3AD,B8H.

Таблица 1.3

Восьме-ричная	Двоичная	Шестнадца-теричная	Двоичная
0	000	0	0000
1	001	1	0001
2	010	2	0010
3	011	3	0011
4	100	4	0100
5	101	5	0101
6	110	6	0110
7	111	7	0111
		8	1000
		9	1001
		A	1010
		B	1011
		C	1100
		D	1101
		E	1110
		F	1111

Глава 2. ПРЕДСТАВЛЕНИЕ ИНФОРМАЦИИ В МИКРОПРОЦЕССОРНОЙ СИСТЕМЕ

Вся информация в МПС представлена в двоичном виде или в виде группы битов. Бит – один двоичный разряд, представимый в виде цифры 0 или 1. Группа из 8 битов называется байтом. Для обозначения больших массивов информации приняты приставки, подобно обозначению единиц измерения физических величин. 1 килобайт (1 КБ)= 1024 байт, 1 мегабайт (МБ)= 1024 КБ, 1 гигабайт = 1024 МБ.

Имеется два типа информации, представленной в МПС. Первый тип - программа, второй - данные, над которыми программа производит действия.

2.1. Представление чисел

Рассмотрим представление данных на примере 8-разрядного микропроцессора. Целое положительное число без знака представляется в диапазоне от 0 до 255.

Чтобы представить целое число со знаком, необходим какой-то способ представления положительных и отрицательных чисел. Принято следующее условие: старший разряд числа определяет его знак.

1XXXXXXX – число отрицательное.

0XXXXXXX – число положительное.

XXXXXXX – модуль числа. X – любая двоичная цифра (0 или 1).

Модуль числа представляется семью двоичными разрядами. Так как 2^7=128, то диапазон представления чисел принял вид – 128 ... -1, 0 ... 127, а общее количество чисел осталось равным 256. В зависимости от решаемой задачи 8 – разрядное двоичное число может представляться как число со знаком или как число без знака. Это представление существует только в сознании программиста. Микропроцессор обрабатывает числа, не зная, как они представляются программисту по общим правилам. Задача программиста – корректировать результат обработки.

Далее ответим на вопрос, как представляется модуль числа со знаком. Положительное число представляется своим прямым кодом, например 00000000 = 0; 00000001 = +1; 00000010 = +2; ... ; 01111111 = +127.

На первый взгляд кажется, что 10000001 = -1, но это не так. Очевидно, что должно выполняться правило +1 - 1 = 0.

Но $_{+}$10000001
00000001

10000010 не равно 0. Чтобы получить 0 к 00000001, необходимо прибавить 11111111. Проверим $_{+}$00000001
11111111
1 00000000

8 значащих разрядов представляют число, равное 0, имеется перенос из старшего разряда результата. Делаем вывод, что число – 1 представляется в виде 11111111. Как из положительного числа 00000001 получить противоположное ему отрицательное число – 1 (11111111)? Проинвертируем число 00000001, получим 11111110, прибавим к нему 1, получим 11111111.

Подобный метод справедлив для любого числа, например для числа – 5 имеем 00000101 => 1111010 => 1111011 – это представление числа – 5. Можно составить таблицу 2.1.

Таблица 2.1

Число	Представление числа
127	01111111
126	01111110
...	...
0	00000000
- 1	11111111
- 2	11111110
...	...
-128	10000000

Итак, положительные числа представлены в прямом коде. *Обратный код* получается из прямого кода поразрядным инвертированием. *Дополнительный код* получается из обратного путем прибавления единицы. Таким образом, отрицательные числа представляются дополнительным кодом. Для закрепления +01111111 (+127)

10000000 (-128)

11111111 (-1)

2.2. Двоично-десятичное представление чисел

В этом случае числа представляются в двоичной системе счисления, но для отображения каждой десятичной цифры используется 4 бита. Например, число 53 в двоично-десятичном коде может быть представлено как 0101 0011. Сравните с двоичным представлением этого же числа 0011 0101.

Глава 3. ТИПОВАЯ СТРУКТУРА МИКРОПРОЦЕССОРА

3.1. Классификация микропроцессоров

Рассмотрим особенности организации процесса обработки информации в цифровых устройствах (цифровых автоматах). Задача создания цифрового автомата, выполняющего определенные действия над двоичными сигналами, заключается в выборе элементов и способе их соединения, обеспечивающих заданное функциональное преобразование. Эти задачи решают с помощью алгебры логики.

По схемному решению и характеру связи между входными и выходными переменными различают два типа цифровых устройств – комбинационные и последовательные. В комбинационных цифровых устройствах совокупность сигналов на выходах в каждый конкретный момент времени полностью определяется входными сигналами, действующими в этот момент на его входах. Алгоритм функционирования комбинационных устройств может быть представлен в виде таблицы соответствия, содержащей значения выходных сигналов для всех возможных комбинаций значений входных сигналов. Цифровые устройства последовательного типа существенно отличаются от комбинационных прежде всего наличием памяти. Их выходные сигналы являются функцией не только входных сигналов, но и внутреннего состояния, в котором устройство находилось до поступления входных сигналов.

На основе цифрового устройства последовательного типа может быть спроектировано устройство, которое в зависимости от последовательности входных сигналов будет выполнять один из многих алгоритмов. Эти входные сигналы могут размещаться и последовательно извлекаться из внешнего блока регистров, называемого управляющей памятью. Некоторые входные сигналы могут использоваться для синхронизации поступления входных сигналов из управляющей памяти и для их адресации. Такое устройство может быть

названо устройством с программируемой логикой или программируемым устройством. К таким устройствам относится и микропроцессор.

Микропроцессоры обладают следующими характеристиками: разрядность адреса и данных, тип корпуса, количество и напряжение источников питания, мощность рассеяния, температурный диапазон, возможность расширения разрядности, время цикла выполнения команд, уровни сигналов, помехоустойчивость, нагрузочная способность, надежность и т.д.

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

Однокристальные микропроцессоры получают при реализации всех аппаратных средств процессора в виде одной БИС. По мере увеличения степени интеграции элементов в кристалле и числа выводов корпуса параметры однокристальных микропроцессоров улучшаются.

Многокристальные микропроцессоры получают при разбиении его логической структуры на функционально законченные части, которые реализуют в виде БИС. Один из возможных вариантов разбиения структуры процессора – это создание трехкристального микропроцессора, содержащего БИС операционного процессора, управляющего процессора и интерфейсного процессора. Операционный процессор (ОП) служит для обработки данных, управляющий процессор (УП) выполняет функции выборки, декодирования и вычисления адресов команд и операндов. Автономность работы и большое быстродействие БИС УП позволяет выбирать команды из памяти с большей скоростью, чем скорость их выполнения в ОП. При этом в УП образуется очередь еще не исполненных команд, заранее подготавливаются данные, которые потребуются ОП в следующих циклах работы. Такая опережающая выборка команд экономит время ОП на ожидание операндов, необходимых для выполнения команд программы. Интерфейсный процессор (ИП) позволяет подключить память и периферийные средства к микропроцессору.

Выбираемые из памяти команды распознаются и выполняются каждой частью микропроцессора автономно, и поэтому может быть обеспечен режим одновременной работы всех БИС МП, то есть конвейерный поточный режим исполнения последовательности команд программы (минимальное время между выполнениями команд). Такой режим работы значительно повышает производительность микропроцессора.

Многокристальные секционные микропроцессоры получаются в том случае, когда в виде БИС реализуются части (секции) логической структуры процессора. Микропроцессорная секция – это БИС, предназначенная для обработки нескольких разрядов данных или выполнения определенных управляющих операций. Секционность БИС МП определяет возможность “наращивания” разрядности обрабатываемых данных или усложнения устройств управления микропроцессором при “параллельном” включении большого числа БИС. Многокристальные секционные микропроцессоры имеют

разрядность от 2-4 до 8-16 бит и позволяют создавать высокопроизводительные процессоры ЭВМ.

По *назначению* различают универсальные и специализированные микропроцессоры. Универсальные микропроцессоры можно применять для решения разнообразных задач. Производительность универсальных микропроцессоров мало зависит от проблемной специфики решаемых задач. Специализация микропроцессора, то есть его проблемная ориентация на ускоренное выполнение определенных функций, позволяет резко увеличить эффективную производительность при решении только определенных задач. Среди специализированных микропроцессоров можно выделить: микроконтроллеры, ориентированные на выполнение сложных последовательностей логических операций; математические МП, предназначенные для повышения производительности при выполнении арифметических операций за счет, например, матричных методов их выполнения; МП для обработки данных в различных областях применения и т.д. С помощью специализированных МП можно эффективно решать сложные задачи параллельной обработки данных.

По виду обрабатываемых входных сигналов различают цифровые и аналоговые микропроцессоры. Сами микропроцессоры – это цифровые устройства, однако могут иметь встроенные аналого-цифровые и цифро-аналоговые преобразователи. Поэтому входные аналоговые сигналы передаются в МП через преобразователь в цифровой форме, обрабатываются и после обратного преобразования в аналоговую форму поступают на выход. С точки зрения архитектуры такие микропроцессоры представляют собой аналоговые функциональные преобразователи сигналов и называются *аналоговыми микропроцессорами*. Они могут выполнять функции любой аналоговой схемы.

По характеру временной организации работы различают синхронные и асинхронные микропроцессоры. Синхронные микропроцессоры – это микропроцессоры, в которых начало и конец выполнения операций задаются устройством управления (время выполнения операций в этом случае не зависит от вида выполняемых команд и величин операндов). Асинхронные микропроцессоры позволяют начало каждой следующей операции определить по сигналу фактического окончания выполнения предыдущей операции.

По количеству выполняемых программ различают одно- и многопрограммные микропроцессоры. В однопрограммных микропроцессорах одновременно выполняется только одна программа. Переход к выполнению другой программы происходит после завершения текущей программы. В много- или мультипрограммных микропроцессорах одновременно выполняются несколько программ.

3.2. Типовая структура микропроцессора

Типовая структура микропроцессора приведена на рисунке ниже. Микропроцессор состоит из трех основных блоков: арифметико – логического устройства (АЛУ), блока внутренних регистров и устройства управления. Для передачи данных между этими блоками используется внутренняя шина данных. АЛУ выполняет одну из главных функций микропроцессора – обработку данных. Перечень функций АЛУ зависит от типа микропроцессора. Операции, выполняемые АЛУ большинства микропроцессоров, следующие: сложение, вычитание, логическое сложение, логическое умножение, исключающее ИЛИ, инверсия, сдвиг вправо, сдвиг влево, увеличение на 1, уменьшение на 1.

Каждый регистр микропроцессора можно использовать для временного хранения одного слова данных. Некоторые регистры имеют специальное назначение, другие – многоцелевое. Последние называются регистрами общего назначения (РОН) и могут использоваться программистом по его усмотрению. Количество и назначение регистров зависит от его типа. Рассмотрим назначение основных регистров, имеющихся почти во всех микропроцессорах.

Аккумулятор – это главный регистр при различных манипуляциях с данными. Большинство арифметических и логических операций осуществляется путем использования АЛУ и аккумулятора. Любая из таких операций над двумя словами данных (операндами) предполагает размещение одного из них в аккумуляторе, а другого в памяти или каком-либо регистре. Результат выполнения операции АЛУ тоже обычно размещается в аккумуляторе, содержимое которого при этом теряется. Операцией другого типа, использующей аккумулятор, является передача данных из одной части микропроцессора в другую. Например, пересылка данных между портом ввода-вывода и памятью, между двумя областями памяти и т.д. Выполнение такой операции осуществляется в два этапа: сначала выполняется пересылка данных из источника в аккумулятор, затем – из аккумулятора в пункт назначения. Микропроцессор может выполнять некоторые действия над данными непосредственно в аккумуляторе. Например, аккумулятор можно очистить (сбросить) путем записи во все его разряды двоичных нулей, установить в единичное состояние путем записи во все его разряды двоичных единиц. Содержимое аккумулятора можно сдвигать влево или вправо (см. схему), инвертировать, то есть значения нулей заменить на значения единиц во всех разрядах и наоборот.

С*четчик команд* – это один из наиболее важных регистров микропроцессора. Как известно, программа – это последовательность команд (инструкций), хранимых в памяти и предназначенных для того, чтобы инструктировать процессор, как решать поставленную задачу. Для правильного выполнения программы команды должны поступать в строго определенном порядке. Счетчик команд обеспечивает формирование адреса очередной команды, записанной в памяти. Перед выполнением программы счетчик команд

необходимо загрузить адресом, указывающим на первую команду программы. Обычно это нулевой адрес. Адрес первой команды программы посылается через регистр адреса памяти по адресной шине к схемам управления памятью, в результате чего из памяти считывается содержимое первой команды. Далее эта команда передается в специальный регистр микропроцессора, называемый *регистром команд.* После извлечения команды из памяти микропроцессор автоматически дает приращение содержимому счетчика команд. Таким образом выполняется последовательность команд, расположенных в памяти одна за другой. Счетчик команд можно загрузить иным содержимым при выполнении особой группы команд. Может возникнуть необходимость выполнить часть программы, которая “выпадает” из последовательности команд основной (главной) программы. Например, такую часть программы, которая повторяется в процессе выполнения всей программы. Вместо того чтобы писать эту часть программы каждый раз, когда в ней возникает необходимость, её записывают один раз и возвращаются к ее повторному выполнению, отступая от указанной последовательности. Часть программы, выполняемая путем отступления от последовательности команд главной программы, называется *подпрограммой*. В данном случае в счетчик команд непосредственно записывается требуемый адрес.

Схема выполнения подпрограмм

Содержание команды	**Значение СК**
Любая	0
Любая	1
Команда перехода на подпрограмму по адресу N	N
Любая	N+1
Любая	…
Команда возврата из подпрограммы	2
Любая	3

Счетчик команд имеет обычно больше разрядов, чем длина слова данных микропроцессора. Так, в большинстве 8-разрядных микропроцессоров число разрядов счетчика команд равно 16. Разрядность счетчика команд должна обеспечить возможность выборки команды из любой области памяти программ (из любой ячейки памяти). Если число разрядов счетчика команд равно 16, то с помощью этих разрядов из памяти можно выбрать 2 в 16 степени ячеек памяти программ, то есть 65536 ячеек.

Регистр команд содержит команду в процессе ее дешифрирования и выполнения. Код команды поступает через шину данных из памяти.

Регистр адреса памяти. При каждом обращении к памяти процессор указывает адрес ячейки памяти, подлежащей использованию. В ячейке памяти может находиться либо код команды, либо данные, над которыми производится действие.

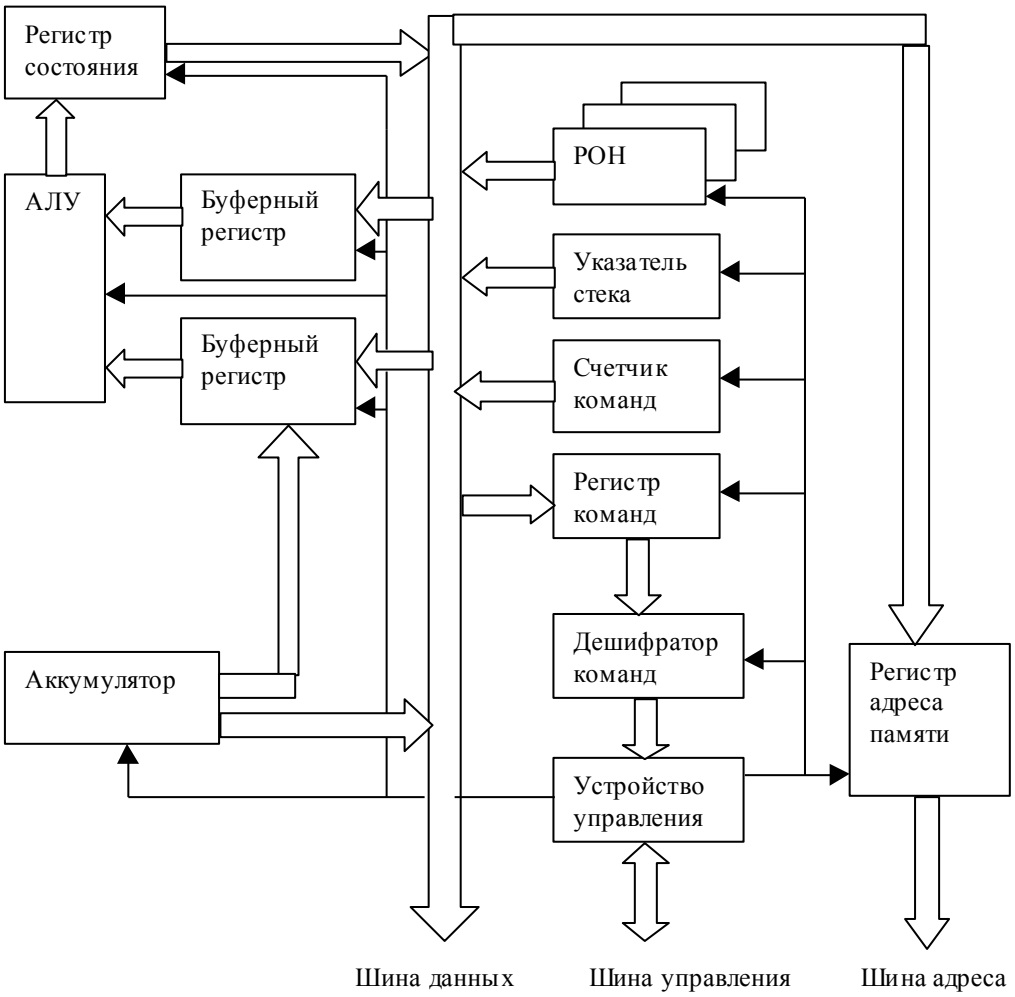


Рис. 3.1. Типовая структурная схема микропроцессора

Буферный регистр предназначен для временного хранения (буферирования) данных.

Регистр состояния предназначен для хранения результатов некоторых проверок, осуществляемых в процессе выполнения программы. Разряды регистра состояний принимают то или иное значение при выполнении операций, использующих АЛУ и некоторые регистры. Запоминание результатов упомянутых проверок позволяет использовать программы, содержащие переходы (нарушения естественной последовательности выполнения команд). При наличии в программе перехода по заданному признаку выполнение команды начинается с некоторой новой области памяти, то есть счетчик команд загружается новым числом. В случае условного перехода такое действие имеет место, если результаты определенных проверок совпадают с ожидаемыми значениями. Указанные результаты находятся в регистре состояния. Регистр состояния предоставляет программисту возможность организовать работу микропроцессора так, чтобы при определенных условиях менялся порядок выполнения команд. Рассмотрим наиболее часто используемые разряды регистра состояния.

1. *Перенос-заем.* Данный разряд указывает, что последняя выполняемая операция сопровождалась переносом или заемом (отрицательным переносом). Значение этого разряда устанавливается равным 1, если в результате сложения двух чисел имеет место перенос из старшего разряда АЛУ. Пример 10001110+11000011=01010001 и имеем перенос из старшего разряда. Заем фиксируется в регистре состояния при вычитании большего числа из меньшего.

2. *Нулевой результат.* Этот разряд принимает единичное значение, если после окончания операции во всех разрядах регистра результата обнаружены двоичные нули.

3. З*наковый.* Принимает единичное значение, когда старший бит регистра результата становится равным 1. При выполнении арифметических операций с числами в дополнительном коде единичное значение старшего бита показывает, что в регистре находится отрицательное число.

Регистры общего назначения (РОН). Большинство МП имеют в своем составе набор регистров, используемых в качестве сверхоперативных запоминающих устройств. Так как АЛУ может совершать операции с содержимым РОН без выхода на внешнюю магистраль адресов и данных, то они происходят много быстрее, чем операции с внешней памятью. Количество РОН и возможность программного доступа к ним у разных микропроцессоров различны.

Указатель стека. Стек – набор регистров микропроцессора или ячеек оперативной памяти, откуда данные или адреса выбираются “сверху” по принципу: первым выбирается элемент, поступивший в стек последним. При записи в стек очередного слова все ранее записанные слова смещаются на один регистр вниз. При выборке слова из стека оставшиеся слова перемещаются на один регистр вверх.

Указанные процедуры иллюстрирует рисунок 3.2.

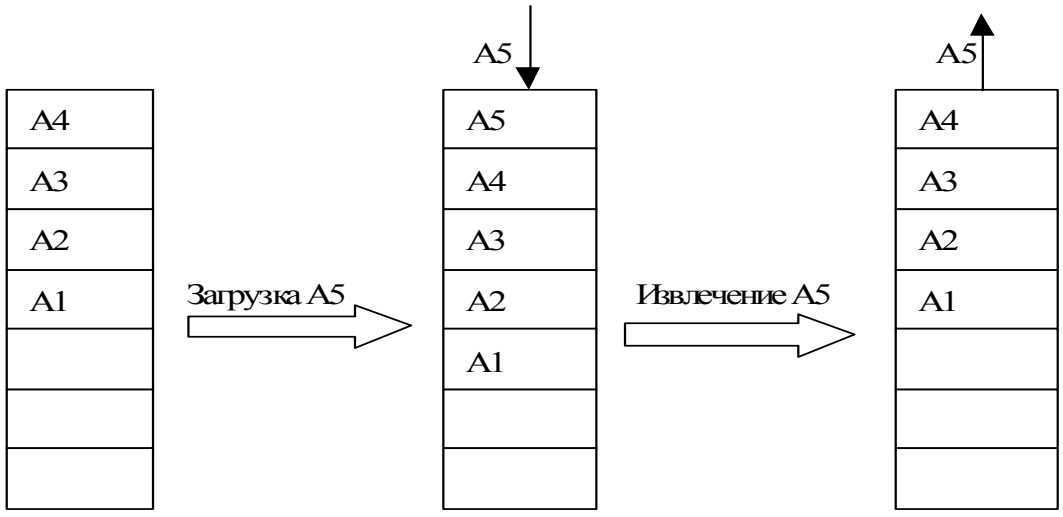


Рис. 3.2

Здесь стек состоит из семи регистров. Если в стек загружается какое-либо слово, например A5, то оно записывается в верхнем регистре, а каждое из слов A1…A4 перемещается в соседние нижние регистры. Если же A5 извлекается из стека, то каждое из словA1…A4 перемещается в соседние верхние регистры. Нельзя извлечь A4 раньше A5, то есть автоматически реализуется принцип “последний зашел – первый вышел, первый зашел – последний вышел”. Стек обычно используется в микропроцессорах для хранения адресов возврата при обращении к подпрограммам, а также для запоминания состояния внутренних регистров при обработке прерываний. Остановимся подробнее на использовании стека при организации подпрограмм. Если в программе имеется подпрограмма, то на выполнение подпрограммы можно перейти из любого места программы по команде CALL <метка>, где CALL – мнемоническое обозначение команды, а <метка> – любое символьное обозначение строки программы, где расположена первая команда подпрограммы. Последняя команда подпрограммы имеет обозначение RET, что означает “выход из подпрограммы”. Команда выхода из подпрограммы не имеет явного указания на адрес команды, которую нужно выполнить после этого, то есть не указан адрес возврата из подпрограммы. Так вот, при выполнении команды CALL адрес возврата из подпрограммы, а это адрес команды CALL, увеличенный на 1, сохраняется в стеке. При выполнении команды RET адрес извлекается из стека и поступает в счетчик команд, то есть далее выполняется команда, следующая в программе за командой CALL.

Таким образом, процесс функционирования стека напоминает работу с пачкой документов, когда каждый новый документ кладется сверху пачки. При такой организации стека необходим специальный регистр – *указатель стека* (УС) для хранения адреса последнего по времени поступления элемента стека.

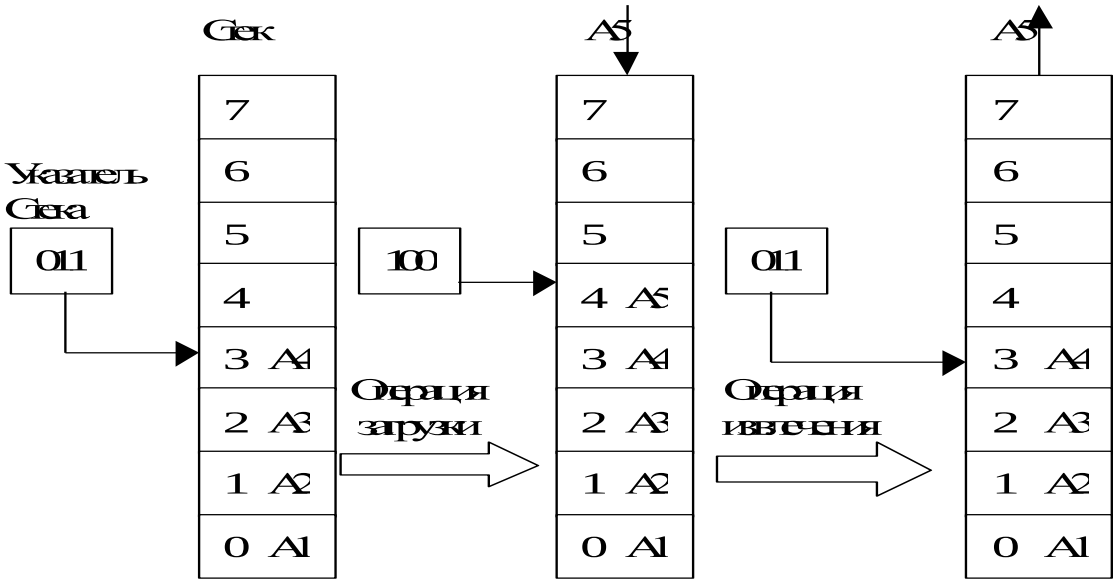


Рис. 3.3

Здесь УС – трехразрядный регистр с двоичным представлением информации. Из рисунка 3.3. видно, что УС всегда хранит адрес регистра стека, заполненного последним, или адрес “вершины стека”.

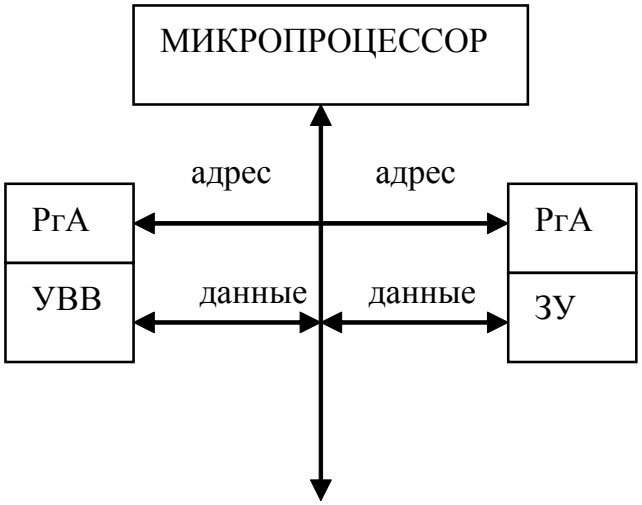
Схемы управления. Роль схем управления заключается в поддержании требуемой последовательности функционирования всех остальных его звеньев. По сигналам схем управления очередная команда извлекается из регистра команд. При этом определяется, что необходимо делать с данными, а затем обеспечивается последовательность действий для выполнения поставленной задачи.

Одна из главных функций схем управления – декодирование команды, находящейся в регистре команд, посредством дешифратора команд, который в результате выдает сигналы, необходимые для ее выполнения.

Помимо указанных выше действий схемы управления выполняют некоторые специальные функции: управление последовательностью включения питания и процессами прерываний. Прерывание – это своего рода запрос, поступающий на схемы управления от других устройств (памяти, устройств ввода-вывода). Прерывание связано с использованием внутренней шины данных микропроцессора. Схемы управления принимают решение, когда и в какой последовательности другие устройства могут пользоваться внутренней шиной данных.

Система шин. На характеристики микропроцессора большое влияние оказывает способ организации его связи с внешней средой – устройствами ввода-вывода (УВВ) и запоминающими устройствами (ЗУ). По способу организации связей с внешней средой различают микропроцессоры с мультиплексированной шиной адреса и данных и с раздельными шинами адреса и данных.

В микропроцессорах с временным мультиплексированием шины адреса данных при обращении к внешнему устройству на общей шине какой-то промежуток времени выставляется адрес, а затем шина предоставляется данным. Такие шины требуют включения дополнительного регистра адреса (РгА), в который записывается адрес по сигналу “адрес-данные”, пока адрес находится на общей шине.



Шина адреса-данных

Рис. 3.4

Ниже представлена схема системы с раздельными шинами адреса-данных.

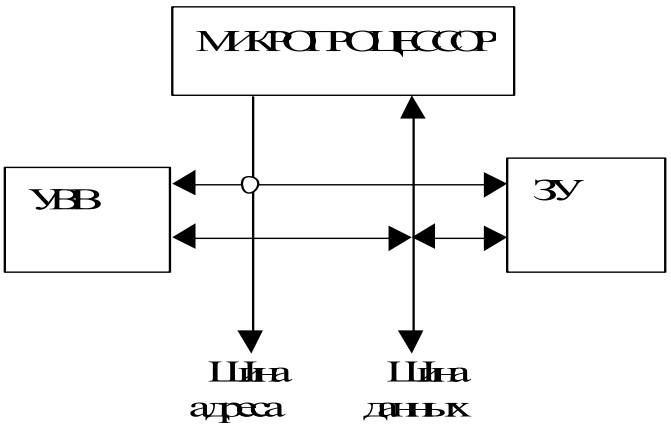


Рис. 3.5

Глава 4. ЗАПОМИНАЮЩИЕ УСТРОЙСТВА МИКРОПРОЦЕССОРНЫХ СИСТЕМ

4.1. Классификация запоминающих устройств

По функциональному назначению все ЗУ, используемые в микропроцессорных системах, разделяются на следующие группы:

- сверхоперативные ЗУ (СОЗУ), представляющие набор регистров, содержимое которых непосредственно используется при обработке информации в микропроцессорах; время доступа к СОЗУ – минимально;

- оперативные ЗУ (ОЗУ), хранящие оперативную информацию (операнды, части программы), требующуюся в процессе работы;
- постоянные ЗУ (ПЗУ), предназначенные для длительного хранения неизменяемой в процессе работы информации (программ, микропрограмм, констант);
- внешние ЗУ, предназначенные для хранения больших объемов информации с небольшой удельной стоимостью бита хранимой информации.

ЗУ характеризуются рядом *качественных показателей.*

1. Емкость ЗУ определяется максимально возможным количеством битов хранимой информации.
2. Ширина выборки определяется количеством информации, записываемой в ЗУ или извлекаемой из него за одно обращение.
3. Время обращения определяется с момента подачи в устройство сигнала записи или чтения до того момента, когда закончатся все действия, связанные с выполняемой операцией.
4. Стоимость бита хранимой информации – отношение стоимости устройства к его информационной емкости.
5. Способность сохранения информации при отключении источников питания. В энергонезависимой памяти при нарушениях в работе системы питания хранимые данные не разрушаются, а в энергозависимой – разрушаются.

4.2. Оперативные запоминающие устройства

По принципу хранения информации ОЗУ делятся на динамические и статические.

Динамические ЗУ строятся на основе запоминающего элемента, сохраняющего свое состояние только определенный промежуток времени и поэтому требующего периодического восстановления. Запоминающим элементом динамического ЗУ служит конденсатор, в котором информация хранится в форме наличия или отсутствия заряда. Из-за утечек заряд на конденсаторе постепенно уменьшается, для восстановления заряда требуется периодическое подключение конденсатора к источнику питания. Этот процесс называется регенерацией памяти. Для обеспечения регенерации требуется внешняя дополнительная логическая схема, реализующая автоматическое восстановление с интервалом в несколько десятых долей миллисекунды. Достоинства: высокий уровень интеграции и быстродействия, низкая стоимость. Недостаток – необходимость регенерации.

Статические ЗУ являются наиболее распространенным видом памяти микропроцессорных систем. Ячейка памяти статического ЗУ представляет собой обычный триггер. Он может быть установлен либо в состояние “1”, либо в состояние “0”. Подобные ячейки памяти объединяются в матричную структуру, то есть размещаются по строкам и столбцам. При построении

статических ЗУ наибольшее распространение получили БИС ЗУ с конфигурацией (n · 1) бит, где n – количество ячеек, равное 256, 512, 1024, 2048, …$2^к$, например (n · 8).

Типичные БИС динамического ОЗУ: КР565РУ6 – емкость 16384 · 1 бит; К565РУ5 – емкость 65536 · 1 бит.

Типичные БИС статического ОЗУ: КР537РУ14 – емкость 4096 · 1 бит; КР132РУ6А – емкость 16384 · 1 бит; К537РУ9 – емкость 2048 · 8 бит.

4.3. Постоянные запоминающие устройства

ПЗУ в микропроцессорных вычислительных системах служат для хранения программ и другой неизменяемой информации. Важное преимущество ПЗУ по сравнению с ОЗУ – сохранение информации при выключении питания. В настоящее время наиболее распространены следующие виды ПЗУ: программируемые на заводе-изготовителе или масочные ПЗУ (МПЗУ); программируемые пользователем; перепрограммируемые ПЗУ.

МПЗУ программируются их изготовителем, который по подготовленной пользователем информации делает фотошаблоны, с помощью которых заносит эту информацию в процессе производства на кристалл ПЗУ. Этот способ самый дешевый и предназначен для крупносерийного производства ПЗУ. Эти ПЗУ можно запрограммировать лишь один раз.

Программируемые пользователем ПЗУ являются более универсальными и, следовательно, более дорогими приборами. Такие приборы доступны для программирования пользователем с помощью специальных устройств, называемых программаторами. Такие ПЗУ применяются на этапе разработки микропроцессорных устройств.

Перепрограммируемые ПЗУ – это ПЗУ с изменяемым содержимым. Допускают многократное программирование. При необходимости в перепрограммировании микросхемы предварительно записанную информацию стирают ультрафиолетовым светом через прозрачное кварцевое окно на поверхности корпуса микросхемы. После очередного программирования информация может сохранятся в ПЗУ десять лет и более.

Микросхемы ПЗУ с электрическим стиранием информации популярны у разработчиков микропроцессорной техники благодаря возможности быстрого стирания и записи, большого допустимого числа циклов перезаписи информации (10000 раз и более). Однако они сложнее и дороже по сравнению с ПЗУ с УФ-стиранием.

4.4. Внешние запоминающие устройства

Внешняя (долговременная) память – это место длительного хранения данных (программ, результатов расчётов, текстов и т.д.), не используемых в данный момент в оперативной памяти компьютера. Внешняя память, в отличие от оперативной, является энергонезависимой. Носители внешней памяти, кроме того,

обеспечивают транспортировку данных в тех случаях, когда компьютеры не объединены в сети (локальные или глобальные). Для работы с внешней памятью необходимо наличие *накопителя* (устройства, обеспечивающего запись и считывание информации) и устройства хранения – *носителя.*

Основные виды накопителей:

- накопители на гибких магнитных дисках (НГМД);
- накопители на жестких магнитных дисках (НЖМД);
- накопители на магнитной ленте (НМЛ);
- накопители CD-ROM, CD-RW, DVD.

Им соответствуют основные виды носителей:

- гибкие магнитные диски *(Floppy Disk)*:

- диаметром 3,5" и ёмкостью 1,44 Мб;

- диаметром 5,25" и ёмкостью 1,2 Мб (в настоящее время устарели и практически не используются, выпуск накопителей, предназначенных для дисков диаметром 5,25", тоже прекращён);

- диски для сменных носителей;

- жёсткие магнитные диски *(Hard Disk)*;
- кассеты для стримеров и других НМЛ;
- диски CD-ROM, CD-R, CD-RW, DVD.

Запоминающие устройства принято делить на виды и категории в связи с их принципами функционирования, эксплуатационно-техническими, физическими, программными и др. характеристиками.

Основные характеристики накопителей и носителей:

- информационная ёмкость;
- скорость обмена информацией;
- надёжность хранения информации;
- стоимость.

Принцип работы *магнитных запоминающих устройств* основан на способах хранения информации с использованием магнитных свойств материалов. Как правило, магнитные запоминающие устройства состоят из собственно *устройств чтения/записи информации* и *магнитного носителя*, на который непосредственно осуществляется запись и с которого считывается информация. Магнитные запоминающие устройства принято делить на виды в связи с исполнением, физико-техническими характеристиками носителя информации и т.д. Наиболее часто различают: дисковые и ленточные устройства. Общая технология магнитных запоминающих устройств состоит в намагничивании переменным магнитным полем участков носителя и считывания информации, закодированной как области переменной намагниченности. Дисковые носители, как правило, намагничиваются вдоль концентрических полей – дорожек, расположенных по всей плоскости дискоидального вращающегося носителя. Запись производится в цифровом коде. Намагничивание достигается за счет создания переменного магнитного поля при помощи головок чтения/записи. Головки представляют собой два или более магнитных управляемых контура с сердечниками, на обмотки

которых подается переменное напряжение. Изменение величины напряжения вызывает изменение направления линий магнитной индукции магнитного поля и, при намагничивании носителя, означает смену значения бита информации с 1 на 0 или с 0 на 1.

Дисковые устройства делят на гибкие *(Floppy Disk)* и жесткие *(Hard Disk)* накопители и носители. Основным свойством дисковых магнитных устройств является запись информации на носитель на концентрические замкнутые дорожки с использованием физического и логического цифрового кодирования информации. Плоский дисковый носитель вращается в процессе чтения/записи, чем и обеспечивается обслуживание всей концентрической дорожки, чтение и запись осуществляется при помощи магнитных головок чтения/записи, которые позиционируют по радиусу носителя с одной дорожки на другую. Для операционной системы данные на дисках организованы в дорожки и секторы. *Дорожки* (40 или 80) представляют собой узкие концентрические кольца на диске. Каждая дорожка разделена на части, называемые *секторами*.

При чтении или записи устройство всегда считывает или записывает целое число секторов независимо от объёма запрашиваемой информации. Размер сектора на дискете равен 512 байт. *Цилиндр* – это общее количество дорожек, с которых можно считать информацию, не перемещая головок. Поскольку гибкий диск имеет только две стороны, а дисковод для гибких дисков – только две головки, в гибком диске на один цилиндр приходится две дорожки. В жестком диске может быть много дисковых пластин, каждая из которых имеет две (или больше) головки, поэтому одному цилиндру соответствует множество дорожек. *Кластер* (или ячейка размещения данных) – наименьшая область диска, которую операционная система использует при записи файла. Обычно кластер – один или несколько секторов.

Перед использованием дискета должна быть форматирована, т.е. должна быть создана её логическая и физическая структура. Дискеты требуют аккуратного обращения. *Они могут быть повреждены, если:*

- дотрагиваться до записывающей поверхности;
- писать на этикетке дискеты карандашом или шариковой ручкой;
- сгибать дискету;
- перегревать дискету (оставлять на солнце или около батареи отопления);
- подвергать дискету воздействию магнитных полей.

Накопители на жестких дисках объединяют в одном корпусе *носитель* и *устройство чтения/записи*, а также нередко и *интерфейсную часть*, называемую *контроллером жесткого диска*. Типичной конструкцией жесткого диска является исполнение в виде одного устройства – камеры, внутри которой находится один или более дисковых носителей, помещённых на одну ось, и блок головок чтения/записи с их общим приводящим механизмом.

Обычно рядом с камерой носителей и головок располагаются схемы управления головками, дисками и часто интерфейсная часть и (или) контроллер. На интерфейсной карте устройства располагается собственно интерфейс дискового устройства, а контроллер с его интерфейсом располагается на самом

устройстве. С интерфейсным адаптером схемы накопителя соединяются при помощи комплекта шлейфов.

Принцип функционирования жёстких дисков аналогичен принципу для ГМД.

Основные физические и логические параметры жестких дисков:

• *Диаметр дисков.* Наиболее распространены накопители с диаметром дисков 2.2, 2.3, 3.14 и 5.25 дюймов;

• *Число поверхностей* – определяет количество физических дисков, нанизанных на ось;

• *Число цилиндров* – определяет, сколько дорожек будет располагаться на одной поверхности;

• *Число секторов* – общее число секторов на всех дорожках всех поверхностей накопителя;

• *Число секторов на дорожке* – общее число секторов на одной дорожке. Для современных накопителей показатель условный, т.к. они имеют неравное число секторов на внешних и внутренних дорожках, скрытое от системы и пользователя интерфейсом устройства;

• *Время перехода от одной дорожки к другой* обычно составляет от 3.5 до 5 миллисекунд, а у самых быстрых моделей может быть от 0.6 до 1 миллисекунды. Этот показатель является одним из определяющих быстродействие накопителя, т.к. именно переход с дорожки на дорожку является самым длительным процессом в серии процессов произвольного чтения/записи на дисковом устройстве;

• *Время установки или время поиска* – время, затрачиваемое устройством на перемещение головок чтения/записи к нужному цилиндру из произвольного положения;

• *Скорость передачи данных,* называемая также *пропускной способностью,* определяет скорость, с которой данные считываются или записываются на диск после того, как головки займут необходимое положение. Измеряется в мегабайтах в секунду (MBps) или мегабитах в секунду (Mbps) и является характеристикой контроллера и интерфейса.

В настоящее время используются в жёсткие диски ёмкостью от 40 Гб до 1000 Гб. Наиболее популярными являются диски ёмкостью 120, 360, 512 Гб.

Кроме НГМД и НЖМД довольно часто используют сменные носители. Довольно популярным накопителем является Zip. Он выпускается в виде встроенных или автономных блоков, подключаемых к параллельному порту.

Эти накопители могут хранить 100 и 250 Мб данных на картриджах, напоминающих дискету формата 3,5", обеспечивают время доступа, равное 29 мс, и скорость передачи данных до 1 Мб/с. Если устройство подключается к системе через параллельный порт, то скорость передачи данных ограничена скоростъю параллельного порта.

К типу накопителей на сменных жёстких дисках относится накопитель Jaz. Ёмкость используемого картриджа – 1 или 2 Гб. Недостаток – высокая стоимость картриджа. Основное применение – резервное копирование данных.

В накопителях на магнитных лентах (чаще всего в качестве таких устройств выступают *стримеры)* запись производится на мини-кассеты. Ёмкость таких кассет

– от 40 Мб до 13 Гб, скорость передачи данных – от 2 до 9 Мб в минуту, длина ленты – от 63,5 до 230 м, количество дорожек – от 20 до 144.

CD-ROM – это оптический носитель информации, предназначенный только для чтения, на котором может храниться до 650 Мб данных. Доступ к данным на CD-ROM осуществляется быстрее, чем к данным на дискетах, но медленнее, чем на жёстких дисках. Компакт-диск диаметром 120 мм (около 4,75") изготовлен из полимера и покрыт металлической плёнкой. Информация считывается именно с этой металлической плёнки, которая покрывается полимером, защищающим данные от повреждения. CD-ROM является односторонним носителем информации. Считывание информации с диска происходит за счёт регистрации изменений интенсивности отражённого от алюминиевого слоя излучения маломощного лазера. Приёмник или фотодатчик определяет, отразился ли луч от гладкой поверхности, был рассеян или поглощён. Рассеивание или поглощение луча происходит в местах, где в процессе записи были нанесены углубления. Фотодатчик воспринимает рассеянный луч, и эта информация в виде электрических сигналов поступает на микропроцессор, который преобразует эти сигналы в двоичные данные или звук. Скорость считывания информации с CD-ROM сравнивают со скоростью считывания информации с музыкального диска (150 Кб/с), которую принимают за единицу. На сегодняшний день наиболее распространенными являются 52x – скоростные накопители CD-ROM (скорость считывания 7500 Кб/с). Накопители CD-R (CD-Recordable) позволяют записывать собственные компакт-диски. Более популярными являются накопители CD-RW, которые позволяют записывать и перезаписывать диски CD-RW, записывать диски CD-R, читать диски CD-ROM, т.е. являются в определённом смысле универсальными.

Аббревиатура DVD расшифровывается как *Digital Versatile* Disk, т.е. *универсальный цифровой диск*. Имея те же габариты, что обычный компакт-диск, и весьма похожий принцип работы, он вмещает чрезвычайно много информации – от 4,7 до 17 Гбайт. Возможно, именно из-за большой емкости он и называется универсальным. Правда, на сегодня реально применяется DVD-диск лишь в двух областях: для хранения видеофильмов (DVD-Video или просто DVD) и сверхбольших баз данных (DVD-ROM, DVD-R).

В отличие от CD-ROM, диски DVD записываются с обеих сторон. Более того, с каждой стороны могут быть нанесены один или два слоя информации. Таким образом, односторонние однослойные диски имеют объем 4,7 Гбайт (их часто называют DVD-5, т.е. диски емкостью около 5 Гбайт), двусторонние однослойные – 9,4 Гбайт (DVD-10), односторонние двухслойные – 8,5 Гбайт (DVD-9), а двусторонние двухслойные – 17 Гбайт (DVD-18). В зависимости от объема требующих хранения данных и выбирается тип DVD-диска. Если речь идет о фильмах, то на двусторонних дисках часто хранят две версии одной картины – одна широкоэкранная, вторая в классическом телевизионном формате.

Глава 5. ОБМЕН ИНФОРМАЦИЕЙ В МИКРОПРОЦЕССОРНОЙ СИСТЕМЕ

5.1. Определение и функции интерфейса микропроцессорных систем

Для включения микропроцессора в любую микропроцессорную систему необходимо установить единые принципы и средства его сопряжения с остальными устройствами системы. Для этих целей служит *интерфейс*, представляющий собой совокупность правил, устанавливающих единые принципы взаимодействия устройств микропроцессорной системы. В состав интерфейса входят: аппаратурные средства соединения (разъем и связи), номенклатура и характер связей, программные средства, описывающие характер сигналов интерфейса и их временную диаграмму, а также описание электрофизических параметров сигналов.

Сложность выполнения разветвленных связей между различными узлами при проектировании БИС и устройств на их основе привела к тому, что практически реализованы и получили широкое распространение магистральные структуры связей, к которым подключены входы и выходы электронных узлов (блоков).

Единая информационная магистраль микропроцессорной системы связывает между собой все устройства и функционально состоит из информационных шин адресов, данных и сигналов управления. Это так называемая трехшинная организация связей.

Шина адресов. В простой микропроцессорной системе только микропроцессор может вырабатывать адреса передаваемой в системе информации. Поэтому шина адресов однонаправленная. Микропроцессор генерирует сигналы кода адреса, а остальные устройства, подключенные к шине адресов, могут только воспринимать их, выполняя непрерывно операцию опознавания кода адреса. Количество линий шины адресов совпадает с разрядностью передаваемого кода адреса. Если используется 16-разрядный код, то в системе разрешается выработка $2^{16} = 65536$ адресов. Они могут все относиться к адресам ячеек памяти или частично к адресам ячеек памяти и адресам регистров устройств ввода-вывода.

Шина данных. Одни устройства, входящие в микропроцессорную систему, могут только передавать или только принимать данные, другие и передавать, и принимать данные. Шина данных является двунаправленной, так как необходимо обеспечить все возможные связи системы. Ее разрядность определяется разрядностью микропроцессора и равна 2, 4, 8, 16, 32 бита.

Шина сигналов управления. Микропроцессор, а также некоторые из устройств ввода-вывода генерируют управляющие сигналы, предназначенные для синхронизации и определения типа операций, выполняемых устройствами. Эти сигналы передаются по совокупности линий, в целом образующих шину сигналов управления. Все сигналы управления согласованы с системными сигналами синхронизации. Они задают начало и последовательность

срабатывания различных устройств системы. Для задания главной последовательности синхронизирующих импульсов обычно используется внешний генератор на основе кварцевого резонатора. Каждый микропроцессор имеет свою уникальную систему сигналов управления. Но все микропроцессоры имеют общие сигналы, например сигнал СБРОС. Внешний сигнал СБРОС приводит к сбросу всех внутренних регистров микропроцессора и загрузке счетчика команд начальным значением адреса, то есть адресом памяти программ, где записана первая команда программы (обычно это адрес 0000H). Важнейшая управляющая функция микропроцессора – определение направления потоков данных в системе. Когда микропроцессор посылает данные какому-то устройству, происходит операция записи данных, а когда получает данные от какого-то устройства, операция чтения данных. Чтобы задать направление передачи по шине данных, микропроцессор генерирует сигналы *чтение – запись.*

Некоторые устройства ввода-вывода могут генерировать сигнал *запрос прерывания*, если им требуются ресурсы микропроцессора для выполнения какой-то задачи, не допускающей отлагательств. Микропроцессор имеет по крайней мере один вход запроса прерываний. Если запрос принимается микропроцессором, то последний информирует систему и устройство, запросившее прерывание микропроцессора, вырабатывая ответный сигнал *подтверждение запроса прерывания.* Разная скорость работы устройств ввода-вывода и микропроцессора порождает необходимость приостановки процессора на время подготовки данных во внешнем устройстве. Для этого некоторые микропроцессоры имеют вход *ожидание.* Подавая сигнал на этот вход, медленно работающее внешнее устройство может приостановить работу микропроцессора на неопределенное время. Микропроцессор в ответ на сигнал *ожидание* вырабатывает сигнал *подтверждение ожидания* и переходит в режим ожидания до снятия сигнала *ожидание* внешним устройством. Всего по шине управления передается до 10 и более разнообразных сигналов управления.

5.2. Связь микропроцессора с устройствами ввода-вывода информации

Во всех микропроцессорных системах применяется программно-управляемая передача данных. Известны три типа программно-управляемой передачи данных: *синхронная, асинхронная и с прерыванием программы.*

Синхронная передача данных характерна для периферийных устройств, для которых известны временные соотношения. При этом типе передачи устройство ввода-вывода должно быть готово к приему или передаче данных за время, равное времени выполнения определенной команды процессора. Синхронная передача реализуется при минимальных затратах технических и программных средств.

Асинхронная передача данных, иногда называемая обменом посредством “рукопожатия”, широко используется. При такой передаче данных микропроцессор перед выполнением операции ввода-вывода проверяет состояние периферийного устройства. Обычно при обмене необходимо:

- Проверить состояние внешнего устройства;
- Активизировать устройство, если последнее готово к обмену;
- Передать данные (ввести или вывести);
- Дезактивировать устройство.

Асинхронная передача является идеальной в смысле согласования временных различий между периферией и микропроцессором. Недостаток ее в том, что микропроцессор вынужден ожидать, пока периферийное устройство не будет готово к обмену. Методом, позволяющим устранить этот недостаток, является передача данных с прерыванием программы.

Передача данных с прерыванием программы – это такой тип обмена данными, при котором для выполнения операций ввода-вывода производится прерывание программы микропроцессора. Такой тип обмена особенно удобен при работе с периферийными устройствами с низким быстродействием, а также в случаях, когда момент передачи данных в микропроцессоре непредсказуем, например при работе с каналами связи. Основная черта такой передачи в том, что обмен инициируется самими внешними устройствами. Для реализации данного обмена в микропроцессоре предусматриваются специальные схемы, которые в конце выполнения каждой машинной операции проверяют наличие сигнала прерывания. Если сигнал прерывания обнаружен, то выполняется следующая последовательность действий:

- После выполнения текущей команды микропроцессор выдает сигнал подтверждения прерывания;
- Микропроцессор запоминает содержимое счетчика команд (обычно в стеке) для того, чтобы после выполнения подпрограммы обслуживания прерывания вернуться к выполнению прерванной программы;
- Запоминается содержимое внутренних регистров и выполняется передача данных под управлением специальной программы (подпрограмма обслуживания прерывания);
- Осуществляется возврат к продолжению выполнения основной программы.

Начало подпрограммы обслуживания прерывания обычно имеет фиксированный адрес в памяти, который называется вектором. В системе могут быть реализованы больше, чем одно прерывание. В этом случае каждое прерывание имеет свой вектор и свою подпрограмму обслуживания. Каждое прерывание в системе может быть программно запрещено или разрешено, это действие называется маскированием и демаскированием соответственно. Если система прерываний одноуровневая, то ни одно из прерываний не может быть обслужено, пока не закончено обслуживание какого-то другого. В многоуровневой системе прерываний каждое прерывание имеет свой приоритет (значимость). Если в какой-то момент времени выполняется прерывание с

более низким приоритетом, чем пришедшее, то обслуживание этого прерывания откладывается, а начинается выполнение вновь поступившего прерывания.

Рассмотренные методы предназначены для обмена данными между микропроцессором и внешними устройствами. Для обмена данными между внешними устройствами и памятью нет необходимости пересылать данные через микропроцессор, так как это займет ресурсы микропроцессора и время. Можно ввести в систему *контроллер прямого доступа в память*, который выполняет указанные функции обмена. Проблема использования контроллера прямого доступа в память заключается в том, то внешнее устройство должно обмениваться данными с памятью, используя уже имеющиеся информационные шины (шина адреса, данных, управления). Задача разделения единого информационного канала между микропроцессором и каналом прямого доступа в память решается путем использования свойств трехуровневого состояния информационных магистралей. На рисунке 5.1 показано подключение микропроцессора, памяти и внешнего устройства к шинам при обычном режиме.

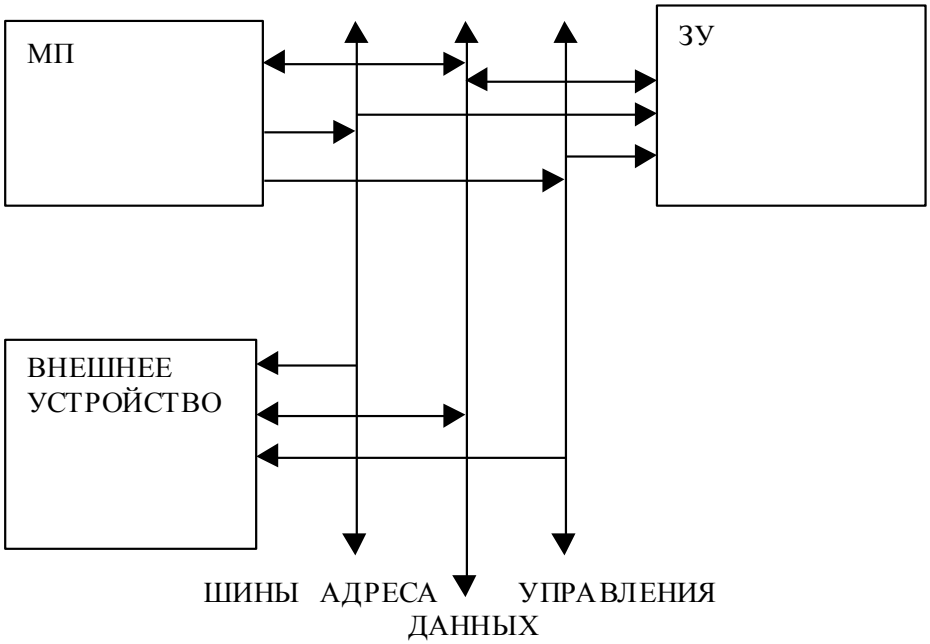


Рис. 5.1. Схема подключения микропроцессора, памяти и внешнего устройства к шинам при обычном режиме

В этом режиме состояние информационных магистралей может иметь два уровня: нулевое и единичное. В режиме прямого доступа в память внешнее устройство берет на себя функции по управлению обменом информацией с памятью, а выводы микропроцессора, подключенные к шинам, должны быть переведены в третье состояние.

Третье состояние – это состояние высокого сопротивления (высокоимпедансное состояние), что означает отключение от шины. Для

реализации этого внешнее устройство посылает сигнал *захват шин* на специальный вывод микропроцессора, в ответ на этот сигнал микропроцессор посылает сигнал *подтверждение захвата* и переводит свои выводы в третье состояние. Режим захвата шин для микропроцессора продолжается до тех пор, пока внешнее устройство работает с памятью и пока им не будет снят сигнал захвата шин. В этом режиме состояние микропроцессора "замораживается" без каких-либо изменений. Режим иллюстрируется рисунком 5.2.

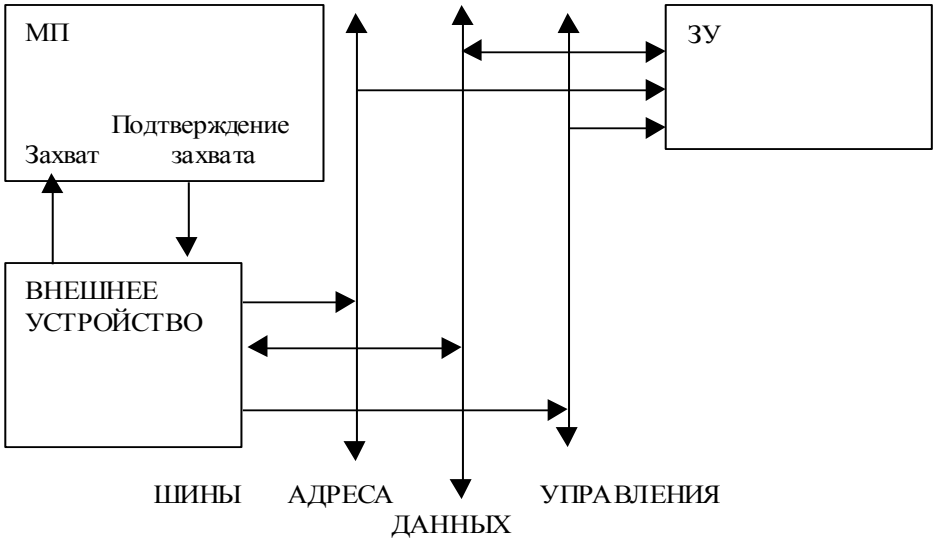


Рис. 5.2

В таком варианте контроллер прямого доступа в память не используется, так как функции управления шинами выполняет частично сам микропроцессор, частично внешнее устройство. Контроллер используется тогда, когда простои микропроцессора в режиме захвата шин недопустимы. Тогда контроллер берет на себя функции анализа свободности шин и выполнения операций обмена тогда, когда шины свободны. В таком режиме микропроцессор даже не замечает, что какие-то устройства занимают шины.

5.3. Основные внешние устройства ввода-вывода информации

5.3.1. Клавиатура

Клавиатура – важнейшее для пользователя устройство, с помощью которого осуществляется ввод данных, команд и управляющих воздействий в ПК. На клавишах нанесены буквы латинского и русского алфавитов, десятичные цифры, математические, графические и специальные служебные знаки, знаки препинания, наименования некоторых команд, функций и др. В зависимости от типа ПК назначение клавиш, их обозначение и размещение могут варьироваться.

Чаще всего клавиатура содержит 101 клавишу, но встречаются еще и старые клавиатуры с 84 клавишами и новые, удобные для использования в системе Windows клавиатуры со 104 клавишами. Имеются клавиатуры со встроенными манипуляторами типа "трекбол" (Treck Ball) и др.

Все клавиши можно разбить на следующие группы:

• буквенно-цифровые клавиши, предназначенные для ввода текстов и чисел;

• клавиши управления курсором, эта группа клавиш может быть использована также для ввода числовых данных, просмотра и редактирования текста на экране;

• специальные управляющие клавиши для переключения регистров, прерывания работы программы, вывода содержимого экрана на печать, перезагрузки ПК и др.;

• функциональные клавиши, широко используемые в сервисных программах в качестве управляющих клавиш.

Буквенно-цифровые клавиши занимают центральную часть клавиатуры. Расположение букв и цифр на клавишах соответствует расположению их на клавиатуре пишущей машинки.

Латинские буквы на клавиатуре расположены по стандарту **QWERTY,** названному так по последовательности первых шести букв в верхнем ряду буквенной клавиатуры**.**

Для русского алфавита размещение буквенно-цифровых клавиш соответствует расположению клавиш на пишущих машинках с русским шрифтом – стандарт **ЙЦУКЕН** (первые шесть букв в верхнем ряду буквенной клавиатуры).

Для обеспечения ввода с клавиатуры русских букв необходим соответствующий драйвер, который должен быть предварительно загружен в оперативную память и оставаться в ней резидентно.

Переключение клавиатуры в режим ввода русских букв (символов кириллицы) и обратный переход на ввод латинских букв осуществляются нажатием двух специальных клавиш <Ctrl> и <Shift>.

Для буквенно-цифровых клавиш существует понятие регистра, т.е. режима их использования. Имеются две пары регистров: *верхний/нижний и латинский/русский*.

На верхнем регистре вводятся прописные (заглавные) буквы, а на нижнем – строчные (маленькие), а также специальные символы и цифры, помещенные соответственно на верхней и нижней части клавиши.

На русском регистре вводятся символы кириллицы, а на латинском – латиницы. Регистры могут использоваться в различных сочетаниях, например верхний – латинский, нижний – русский и т.п.

Выбор режима нижний/верхний производится при помощи клавиши <Caps Lock> (Capitals Lock – фиксация прописных букв) и <Shift> (Shift – сдвиг, замена). Клавиша <Caps Lock> закрепляет режим ввода прописных или строчных букв. В режиме прописных букв светится индикатор Caps Lock в верхней правой части клавишной панели. Клавиша <Shift> изменяет режим клавиатуры на противоположный, пока она нажата. Клавиша <Space> вводит пробел в строку символов.

Клавиши управления курсором расположены в правой части панели клавиатуры. Для удобства работы они продублированы и состоят из трех групп:

- малая цифровая клавиатура;
- клавиши просмотра текста на экране и его редактирования;
- клавиши управления курсором.

Клавиши *малой цифровой клавиатуры* могут быть использованы в двух режимах:

- в режиме управления курсором;
- в режиме ввода цифр, знаков математических операций и точки.

Выбор режима производится при помощи клавиш <Num Lock> (Number Lock – фиксация цифр) и <Shift>. Клавиша <Num Lock> закрепляет режим ввода цифр, а <Shift> изменяет режим клавиатуры на противоположный, пока она нажата.

В режиме *ввода цифр*, математических знаков и точки светится индикатор Num Lock в верхней правой части клавишной панели, клавиши имеют следующее назначение (см. таблицу 5.1).

Таблица 5.1

Клавиша	Назначение	Клавиша	Назначение
+	**Сложение**	/	**Деление**
-	**Вычитание**	.	**Ввод точки**
*	**Умножение**	0-9	**Ввод соответствующих цифр**

Курсором называется символ (обычно это узкий мерцающий прямоугольник или жирная черта), указывающий позицию на экране дисплея, в которой будет отображаться очередной выведенный на экран символ. Назначение клавиш в *режиме управления курсором* – см. таблицу 5.2.

Таблица 5.2

Клавиша	Назначение
<	Перемещение курсора влево на одну позицию при кратковременном нажатии; при длительном нажатии курсора перемещается влево непрерывно
^	Перемещение курсора вверх на одну позицию при кратковременном нажатии; при длительном нажатии курсор перемещается вверх непрерывно
>	Перемещение курсора вправо на одну позицию при кратковременном нажатии; при длительном нажатии курсор перемещается вправо непрерывно
∨	Перемещение курсора вниз на одну позицию при кратковременном нажатии; при длительном нажатии курсор перемещается вниз непрерывно
Home	Перемещение курсора в первую позицию строки (Home – домой)
End	Перемещение курсора в последнюю позицию строки (End – конец)
PgUp	Перемещение по тексту в направлении его начала на одну страницу (обычно на 25 строк), т.е. возврат на одну страницу (Page Up – страница вверх)
PgDn	Перемещение по тексту в направлении его конца на одну страницу, т.е. продвижение вперед на одну страницу (Page Down – страница вниз)
Ins	Переключение клавиатуры из режима замены в режим вставки и обратно; в режиме вставки каждый вновь введенный символ помещается перед символом, на который указывает курсор; часть же строки, расположенная правее курсора, сдвигается на одну позицию вправо (Insert – вставить)
Del	Удаление на экране указанного символа; при этом часть строки, расположенная правее курсора, сдвигается на одну позицию влево, исключая разрыв строки (Delete – удалить)

Специальные управляющие клавиши (их называют также служебными), расположенные вокруг группы алфавитно-цифровых клавиш, имеют следующее назначение (таблица 5.3).

Таблица 5.3

Клавиша	Назначение
Esc	Отмена каких-либо действий и/или выхода из программы, подменю и т.п. (Escape – выход, переход)
Ctrl	Клавиша используется совместно с другими клавишами, изменяя их действия (Control – управление)
Alt	Клавиша используется совместно с другими клавишами, изменяя их действия (Alter – изменять)
Enter	Клавиша ввода информации и возврата каретки, служит для завершения ввода очередной строки информации (Enter – ввод)
Backspace	Возврат на одну позицию по экрану влево с удалением предыдущего символа (Backspace – пробел назад)
Tab	Перемещение курсора вправо на задаваемое по запросу количество позиций или перемещение, заранее предопределенное выполняемой программой (Tabulation – табуляция)
Shift	Клавиша смены регистра (Shift – сдвиг)
Print Scrn	Распечатка на принтере информации, выведенной на экран (Print Screen – печать экрана)
Caps Lock	Фиксация прописных/ строчных букв (Caps Lock – фиксация прописных букв)
Nut Lock	Фиксация режимов работы малой цифровой клавиатуры (Number Lock – фиксация цифр)
Scroll Lock	Переключение режима вывода на экран дисплея; при включении прокрутки светится соответствующий индикатор в верхней правой части панели (Scroll Lock – фиксация прокрутки)
Pause/Break	Прерывание (приостановка) выполнения программ и процедур, например, вывода информации на экран; для продолжения выполнения приостановленной программы нужно нажать любую клавишу (Pause/Break – пауза/прерывание)

Некоторые важные *специальные комбинации клавиш,* при которых клавиши нажимаются одновременно (таблица 5.4).

Таблица 5.4

Клавиша	Назначение
Ctrl+Alt+Del	Перезагрузка DOS
Ctrl+Break	Прекращение работы выполняемой программы
Ctrl+C	Прекращение работы выполняемой программы
Ctrl+Num Lock	Приостановка выполнения программы
Ctrl+S	Приостановка выполнения программы

Функциональные клавиши <F1>-<F12> размещены в верхней части клавиатуры, Эти клавиши предназначены для различных специальных действий; они программируются и для каждого программного продукта имеют свое назначение (в принципе программироваться могут и некоторые специальные клавиши).

В большинстве программ принято, что клавиша <F1> связана с вызовом подсказки. При входе в программу по <F1> выдается общая подсказка с кратким описанием вариантов функционирования программы и назначением функциональных клавиш в ней. При работе с программой по <F1> выдается контекстно-зависимая подсказка, т.е. подсказка по тому режиму, по той функции, которая программой реализуется в данный момент.

Блок клавиатуры в профессиональных ПК конструктивно выполнен автономно от основной платы компьютера и кроме клавиатуры содержит контроллер клавиатуры, состоящий из буферной памяти и схемы управления. Он подключается к основной плате с помощью 4-проводного интерфейса (линии интерфейса используются для передачи соответственно тактовых импульсов, данных, напряжения питания +5 вольт и нуля).

Контроллер клавиатуры осуществляет:

- сканирование (опрос) состояния клавиш;
- буферизацию (временное запоминание) до 20 отдельных кодов клавиш на время между двумя соседними опросами клавиатуры со стороны МП;
- преобразование кодов нажатия клавиш (scan – кодов) в коды ASC11 с помощью хранящихся в ПЗУ программируемых системных таблиц драйвера клавиатуры;
- тестирование (проверку работоспособности) клавиатуры при включении ПК.

При нажатии и отпускании клавиши в буферную память контроллера клавиатуры поступает код нажатия или отпускания (соответственно 0 или 1) в седьмой бит байта и номер клавиши или ее scan – код в остальные 7 бит байта. При поступлении любой информации в буферную память посылается запрос на аппаратное прерывание, инициируемое клавиатурой. При выполнении прерывания scan-код преобразуется в код ASC11, и оба кода (scan

– код и ASC11 – код) пересылаются в соответствующее поле ОЗУ машины. При этом по наличию кода отпускания проверяется, все ли клавиши отпущены в момент нажатия следующей клавиши (это необходимо для организации совместной работы с клавишами <Shift>, <Ctrl> и <Alt>).

Контроллер клавиатуры организует и автоматическое повторение клавишной операции: если клавиша нажата более 0,5 с, то генерируются повторные коды нажатия клавиши через регулярные интервалы так, как если бы вы клавишу нажимали повторно.

5.3.2. Видеотерминальные устройства

Видеотерминал состоит из *видеомонитора (дисплея)* и *видеоконтроллера (адаптера)*. Видеоконтроллеры входят в состав системного блока ПК (находятся на *видеокарте*, устанавливаемой в разъем материнской платы), а видеомониторы – это внешние устройства ПК.

5.3.2.1. Видеомониторы

Видеомонитор, дисплей или просто монитор – устройство отображения текстовой и графической информации на экране (в стационарных ПК – на экране электронно-лучевой трубки (ЭЛТ). В портативных ПК – на жидкокристаллическом плоском экране).

Размер экрана монитора задается обычно величиной его диагонали в дюймах: от 10 до 21 дюйма (наиболее типичное значение – 14 дюймов).

Разрешающая способность мониторов. Видеомониторы обычно могут работать в двух режимах: *текстовом* и *графическом*. В *текстовом* режиме изображение на экране монитора состоит из символов расширенного набора ASC11, формируемых знакогенератором (возможны примитивные рисунки, гистограммы, рамки, составленные с использованием символов псевдографики). В *графическом* режиме на экран выводятся более сложные изображения и надписи с различными шрифтами и размерами букв, формируемых из отдельных мозаичных элементов – *пикселей*.

Разрешающая способность мониторов нужна прежде всего в графическом режиме и связана с размером пикселя. Измеряется разрешающая способность максимальным количеством пикселей, размещающихся по горизонтали и по вертикали на экране монитора. Зависит разрешающая способность как от характеристик монитора, так, даже в большей степени, и от характеристик видеоадаптера. Стандартные значения разрешающей способности современных мониторов: 640x480, 800x600,1024x768, 1600x1200, но реально могут быть и иные значения.

Важной характеристикой монитора, определяющей четкость изображения на экране, является *размер зерна* (точки) *люминофора* экрана монитора. Чем меньше зерно, тем, естественно, выше четкость и тем меньше устает глаз. Величина зерна мониторов имеет значения от 0,41 до 0,18 мм.

Следует иметь в виду, что у мониторов с большим зерном не может быть достигнута высокая разрешающая способность (например, экран с диагональю 14 дюймов имеет ширину 265 мм, для получения разрешающей способности 1024 точки по горизонтали размер зерна не должен превышать 265/1024 = 0,22 мм, в противном случае пиксели сливаются и изображение не будет четким).

Совместно с компьютерами IBM PC могут использоваться различные типы мониторов, *как монохромные,* так и *цветные.*

Монохромные мониторы. Они значительно дешевле цветных, но имеют большую разрешающую способность. Среди монохромных чаще других используются:

- монохромные мониторы прямого управления – обеспечивают высокую разрешающую способность при отображении текстовых и псевдографических символов, но не предназначены для формирования графических изображений, построенных из отдельных пикселей. Работают совместно только с монохромными видеоконтролерами;
- композитные монохромные мониторы – обеспечивают качественное отображение и символьной, и графической информации при совместной работе с цветным графическим адаптером (но выдают, естественно, монохромное: зеленое или чаще всего янтарное изображение).

Цветные мониторы. В качестве *цветных мониторов* используются:

- композитные цветные мониторы и телевизоры, обеспечивающие и цвет, и графику, но имеющие довольно низкую разрешающую способность;
- цветные RGB-мониторы являются самыми качественными, обладающими высокой разрешающей способностью и графики, и цвета (RGB-Red-Green-Blue (красный – зеленый – синий), используют для каждого из этих цветовых сигналов свой провод, а в композитных – все три цветовых сигнала идут по одному проводу). RGB – мониторы работают совместно с цветным графическим контроллером.

Для настольных компьютеров используются различные типы видеомониторов: CD (Color Display – цветной дисплей), ECD (Enhanced CD – улучшенный цветной дисплей) и PGS (Professional Grafics System – профессиональная графическая система) и др.

Наибольшую разрешающую способность с хорошей передачей полутонов из применяемых в настоящее время мониторов имеют монохромные композитные мониторы с черно-белым изображением типа "paper white" (используемые часто в настольных издательских системах); их разрешающая способность при совместной работе с видеоконтроллером типа SVGA: 1280x1024 пикселей.

Среди прочих характеристик мониторов следует отметить: наличие плоского или выпуклого экрана (первый вариант предпочтительнее: большая прямоугольность изображения, меньшие блики); уровень высокочастотного радиоизлучения (увеличивается с увеличением полосы частот видеосигнала, но значительно уменьшается при хорошем экранировании –

мониторы с низким уровнем излучения типа LR (Low Radiation); наличие защиты экрана от электростатических полей – мониторы типа AS (Anti Static); наличие системы энергосбережения – мониторы типа G (Green) и др.

5.3.2.2. Видеоконтроллеры

Видеоконтроллеры (видеоадаптеры) являются внутрисистемными устройствами, непосредственно управляющими мониторами и выводом информации на их экран. Видеоконтроллер содержит: схему управления монитором, растровую память *(видеопамять,* хранящую воспроизводимую на экране информацию и использующую поле видеобуфера в оперативной памяти), сменные микросхемы ПЗУ (матрицы знаков), порты ввода-вывода.

Основные характеристики видеоконтроллера:

- режимы работы (текстовый и графический);
- воспроизведение цветов (монохромный и цветной);
- разрешающая способность (число адресуемых на экране монитора пикселей по горизонтали и вертикали);
- емкость и число страниц в буферной памяти (число страниц – это число запоминаемых текстовых экранов, любой из которых путем прямой адресации может быть выведен на отображение в мониторе);
- размер матрицы символа (количество пикселей в строке и столбце матрицы, формирующей символ на экране монитора);
- разрядность шины данных, определяющая скорость обмена данными с системной шиной, и др.

Важная характеристика – *емкость видеопамяти,* она определяет количество хранимых в памяти пикселей и их атрибутов. Разрядность атрибута пикселя определяет, в частности, максимально возможное число полутонов или цветовых оттенков, учитываемых при отображении пикселя. Необходимую емкость видеопамяти можно приблизительно сосчитать, умножив количество байтов атрибута на количество пикселей экрана. Например, при разрешающей способности монитора 800x600 пикселей и стандарте True Color емкость видеопамяти должна быть не менее 1440000 байт.

Наиболее широкое применение нашли следующие видеоконтроллеры:

- Hercules – монохромный графический адаптер;
- MDA (Monochrome Displey Adapter) – монохромный дисплейный адаптер;
- MGA (Monochrome Graphics Adapter) – монохромный графический адаптер;
- CGA (Color Graphics Adapter) – цветной графический адаптер;
- EGA(Enhanced Graphics Adapter) – улучшенный графический адаптер;
- VGA (Video Graphics Adapter) – видеографический адаптер, иногда его называют видеографической матрицей (Video Graphics Array);
- SVGA (Super VGA) – улучшенный видеографический адаптер;
- PGA (Professional GA) – профессиональный графический адаптер.

Например, видеоконтроллеры SVGA типа VESA (видеокарты VESA) с объемом видеопамяти 1-2 Мбайта обеспечивают наибольшую разрешающую способность 1280x1024 при отличной передаче полутонов и цветовых оттенков.

5.3.3. Принтеры

Принтеры (печатающие устройства) – это устройства вывода данных из ЭВМ, преобразующие информационные ASC11– коды в соответствующие им графические символы (буквы, цифры, знаки и т.п.) и фиксирующие эти символы на бумаге.

Принтеры являются наиболее развитой группой внешних устройств ПК, насчитывающей до 1000 различных модификаций. Принтеры разнятся между собой по различным признакам:

- цветность (черно-белые и цветные):
- способ формирования символов (знакопечатающие и знакосинтезирующие):
- принцип действия (матричные, термические, струйные, лазерные);
- способы печати (ударные, безударные) и формирования строк (последовательные, параллельные);
- ширина каретки (с широкой (375-450 мм) и узкой (250 мм) кареткой);
- длина печатной строки (80 и 132-136 символов);
- набор символов (вплоть до полного набора символов ASC11);
- скорость печати:
- разрешающая способность, наиболее употребительной единицей измерения является dpi (dots per inch) – количество точек на дюйм.

Внутри ряда групп можно выделить по несколько разновидностей принтеров. Например, широко применяемые в ПК матричные знакосинтезирующие принтеры по принципу действия могут быть ударными, термографическими, электрографическими, электростатическими, магнитографическими и др. Среди ударных принтеров часто используются литерные, шаровидные, лепестковые (типа "ромашка"), игольчатые (матричные) и др.

Печать у принтеров может быть посимвольная, построчная, постраничная. Скорость печати варьируется от 10-300 зн./с (ударные принтеры) до 500-1000 зн./с и даже до нескольких десятков (до 20) страниц в минуту (безударные лазерные принтеры); разрешающая способность – от 3-5 точек на миллиметр до 30-40 точек на миллиметр (лазерные принтеры).

Многие принтеры позволяют реализовать:

- эффективный вывод графической информации (с помощью символов псевдографики);
- сервисные режимы печати: плотная печать, печать с двойной шириной, с подчеркиванием, с верхними и нижними индексами, выделенная печать (каждый символ печатается дважды), печать за два прохода (второй раз символ

печатается с незначительным сдвигом) и многоцветная (до 100 различных цветов и оттенков) печать.

5.3.3.1. Матричные принтеры

В *матричных* принтерах изображение формируется из точек ударным способом, поэтому их более правильно называть ударно-матричные принтеры, тем более что и прочие типы знакосинтезирующих принтеров тоже чаще всего используют матричное формирование символов, но безударным способом. Тем не менее *матричные принтеры* – это их общепринятое название, поэтому и будем его придерживаться.

Матричные принтеры могут работать в двух режимах – текстовом и графическом. В текстовом режиме на принтер посылаются коды символов, которые следует распечатать, причем контуры символов выбираются из знакогенератора принтера. В графическом режиме на принтер пересылаются коды, определяющие последовательность и местоположение точек изображения.

В игольчатых (ударных) матричных принтерах печать точек осуществляется тонкими иглами, ударяющими бумагу через красящую ленту. Каждая игла управляется собственным электромагнитом. Печатающий узел перемещается в горизонтальном направлении, и знаки в строке печатаются последовательно. Многие принтеры выполняют печать как при прямом, так и при обратном ходе. Количество иголок в печатающей головке определяет качество печати. Недорогие принтеры имеют 9 игл. Матрица символов в таких принтерах имеет размерность 7x9 или 9x9 точек. Более совершенные матричные принтеры имеют 18 игл и даже 24.

Качество печати матричных принтеров определяется также возможностью вывода точек в процессе печати с частичным перекрытием за несколько проходов печатающей головки.

Для текстовой печати в общем случае имеются следующие режимы, характеризующиеся различным качеством печати:

- режим черновой печати (Draft);
- режим печати, близкий к типографскому (NLQ – Near – Letter – Quality);
- режим с типографским качеством печати (LQ – Letter – Quality);
- сверхкачественный режим (SLQ – Super Letter – Quality).

Режимы LQ и SLQ поддерживаются только струйными и лазерными принтерами.

В принтерах с различным числом иголок эти режимы реализуются по-разному. В 9-игольчатых принтерах печать в режиме Draft выполняется за один проход печатающей головки по строке. Это самый быстрый режим печати, но зато имеет самое низкое качество. Режим NLQ реализуется за два прохода. После первого прохода головки бумага протягивается на расстояние, соответствующее половинному размеру точки, а затем совершается второй

проход с частичным перекрытием точек. При этом скорость печати уменьшается вдвое.

Матричные принтеры, как правило, поддерживают несколько шрифтов и их разновидностей, среди которых получили широкое распространение roman (мелкий шрифт пишущей машинки), italic (курсив), bold – face (полужирный), expanded (растянутый), elite (полусжатый), condenced (сжатый), pica (прямой шрифт – пицеро), courier (курьер), san serif (рубленый шрифт – сенсериф), serif (сериф), prestige elite (престиж – элита) и пропорциональный шрифт (ширина поля, отводимого под символ, зависит от ширины символа).

Переключение режимов работы матричных принтеров и смена шрифтов могут осуществляться как программно, так и аппаратно путем нажатия имеющихся на устройствах клавиш и/или соответствующей установки переключателей.

Быстродействие матричных принтеров при печати текста в режиме находится в пределах 100-300 зн./с, что соответствует примерно двум страницам в минуту (с учетом смены листов).

5.3.3.2. Термопринтеры

Кроме матричных игольчатых принтеров есть еще группа матричных *термопринтеров*, оснащенных вместо игольчатой печатающей головки головкой с термоматрицей и использующих при печати специальную термобумагу или термокопирку (что безусловно, является их существенным недостатком).

5.3.3.3. Струйные принтеры

В печатающей головке этих принтеров вместо иголок имеются тонкие трубочки – сопла, через которые на бумагу выбрасываются мельчайшие капельки красителя (чернил). Это безударные печатающие устройства. Матрица печатающей головки обычно содержит от 12 до 64 сопел. В последние годы в их совершенствовании достигнут существенный прогресс: созданы струйные принтеры, обеспечивающие разрешающую способность до 20 точек/мм и скорость печати до 500 зн./с при отличном качестве печати, приближающемся к качеству лазерной печати. Имеются цветные струйные принтеры.

5.3.3.4. Лазерные принтеры

В них применяется электрографический способ формирования изображений, используемый в одноименных копировальных аппаратах. Лазер служит для создания сверхтонкого светового луча, вычерчивающего на поверхности предварительно заряженного светочувствительного барабана контуры невидимого точечного электронного изображения – электрический

заряд стекает с засвеченных лучом лазера точек на поверхности барабана. После проявления электронного изображения порошком красителя (тонера), налипающего на разряженные участки, выполняется печать – перенос тонера с барабана на бумагу и закрепление изображения на бумаге разогревом тонера до его расплавления.

Лазерные принтеры обеспечивают наиболее качественную печать с разрешением до 50 точек/мм (1200 dpi) и скорость печати до 1000 зн./с. Широко используются цветные лазерные принтеры. Например, лазерный принтер фирмы Tektronix (США) Phaser 550 имеет разрешение и по горизонтали, и по вертикали 1200 dpi; скорость цветной печати – 5 страниц формата А4 в минуту, скорость монохромной печати – 14 стр./мин.

К микропроцессорам принтеры могут подключаться и через параллельный, и через последовательный порт. Параллельные порты используются для подключения параллельно работающих (воспринимающих информацию сразу по байту) принтеров. Например, адаптеры типа Centronics позволяют подключать одновременно до трех принтеров. Последовательные порты (2 шт.) служат для подключения последовательно работающих (воспринимающих информацию последовательно по 1 биту) принтеров, например адаптеры типа RS – 232C (стык С2). Большинство принтеров используют параллельные порты.

Многие быстродействующие принтеры имеют собственную буферную память, емкостью до нескольких сотен килобайт. Самые популярные принтеры (их доля составляет не менее 30%) выпускает японская фирма Seiko Epson. Язык управления этими принтерами (ESP/P) стал фактическим стандартом. Широко используются также принтеры фирм Star Micronics, Hewlett, Xerox, Mannesmann, Citizen, Panasonic и др.

5.3.4. Сканеры

Сканер – это устройство ввода в ЭВМ информации непосредственно с бумажного документа. Можно вводить тексты, схемы, рисунки, графики, фотографии и другую графическую информацию.

Сканеры являются важнейшим звеном электронных систем обработки документов и необходимым элементом любого "электронного стола". Записывая результаты своей деятельности в файлы и вводя информацию с бумажных документов в ПК с помощью сканера с системой автоматического распознавания образов, можно сделать реальный шаг к созданию систем безбумажного делопроизводства.

Сканеры весьма разнообразны, и их можно классифицировать по целому ряду признаков. Сканеры бывают черно-белые и цветные.

Черно-белые сканеры могут считывать штриховые изображения и полутоновые. Штриховые изображения не передают полутонов или, иначе, уровней серого. Полутоновые позволяют распознать и передать 16, 64 или 256 уровней серого.

Цветные сканеры работают и с черно-белыми, и с цветными оригиналами. В первом случае они могут использоваться для считывания и штриховых, и полутоновых изображений. В цветных сканерах используется цветовая модель RGB: сканируемое изображение освещается через вращающийся RGB – светофильтр или от последовательно зажигаемых трех цветных ламп. Сигнал, соответствующий каждому основному цвету, обрабатывается отдельно. Число передаваемых цветов колеблется от 256 до 65536 (стандарт High Color) и даже до 16,7 миллионов (стандарт True Color).

Разрешающая способность сканеров составляет от 75 до 1600 dpi.

Конструктивно сканеры бывают *ручные* и *настольные*.

5.3.4.1. Ручные сканеры

Ручные сканеры конструктивно самые простые: они вручную перемещаются по изображению. С их помощью за один проход вводится лишь небольшое количество строчек изображения (их захват обычно не превышает 105 мм). У ручных сканеров имеется индикатор, предупреждающий оператора о превышении допустимой скорости сканирования. Эти сканеры имеют малые габариты и низкую стоимость. Скорость сканирования 5-50 мм/с (зависит от разрешающей способности). Например, сканеры Mustek: GS - 400L – черно-белый полутоновый; CG - 8400I – цветной.

5.3.4.2. Настольные сканеры

Настольные сканеры, в свою очередь, делятся на *планшетные*, *роликовые* и *проекционные*.

Планшетные сканеры самые распространенные. В них сканирующая головка перемещается относительно оригинала автоматически. Они позволяют сканировать и листовые, и сброшюрованные документы. Скорость сканирования 2-10 страниц формата А4 в секунду. Классическим примером цветных сканеров являются: Mustek Paragon 1200, Epson ES1200, HP Scan 1, – 11CX.

Роликовые сканеры наиболее автоматизированы. В них оригинал автоматически перемещается относительно сканирующей головки, часто имеется автоматическая подача документов, но сканируемые документы только листовые. Примером является сканер Mustek SF – 630 со скоростью сканирования 10 страниц формата А4 в секунду.

Проекционные сканеры внешне напоминают фотоувеличитель, но внизу лежит сканируемый документ, а наверху находится сканирующая головка. Сканер оптическим образом сканирует информационный документ и вводит полученную информацию в виде файла в память компьютера. Файл, создаваемый сканером в памяти машины, называется *битовой картой*.

Существуют два формата представления графической информации в файлах компьютера: *растровый* формат и *векторный*. В *растровом* формате графическое изображение запоминается в файле в виде мозаичного набора множества точек (нулей и единиц), соответствующих пикселям отображения этого изображения на экране дисплея. Редактировать этот файл средствами стандартных текстовых и графических процессоров не представляется возможным, ибо эти процессоры не работают с мозаичным представлением информации. В *текстовом* формате информация идентифицируется характеристиками шрифтов, кодами символов и абзацев. Стандартные текстовые процессоры предназначены для работы именно с таким представлением информации.

Следует также иметь в виду, что битовая карта требует большого объема памяти для своего хранения. Так, битовая карта с 1 листа документа формата А4 (204х297 мм) с разрешением 10 точек/мм и без передачи полутонов (штриховое изображение) занимает около 1 Мбайта памяти, она же при воспроизведении 16 оттенков серого – 4 Мбайта, при воспроизведении цветного качественного изображения (стандарт High Color – 65536 цветов) – 16 Мбайт. Иными словами, при использовании стандарта True Color и разрешающей способности 50 точек/мм для хранения даже одной битовой карты может не хватить емкости НЖМД. Сокращение объема памяти, необходимой для хранения битовых карт, осуществляется различными способами сжатия информации.

Наиболее предпочтительным является использование сканера совместно с программами *систем распознавания образов*, например типа OCR (Optical Character Recognition). Система OCR распознает считанные сканером с документа битовые (мозаичные) контуры символов (букв и цифр) и кодирует их ASC11 – кодами, переводя в удобный для текстовых редакторов векторный формат.

Некоторые системы OCR предварительно нужно обучить распознаванию – ввести в память сканера шаблоны и прототипы распознаваемых символов и соответствующие им коды. Сложности возникают при распознавании букв, совпадающих по начертанию в разных алфавитах (например, в латинском (английском) и в русском – кириллица), и разных гарнитур (способов начертания) шрифтов. Но большинство систем не требуют обучения, в их памяти уже заранее помещены распознаваемые символы. Так одна из лучших OCR – программный пакет TIGER 2.0 – содержит прототипы 30 различных гарнитур, а для распознавания английских и русских букв использует встроенные электронные словари.

В последние годы появились интеллектуальные программы распознавания образов типа Omnifont, которые опознают символы не по точкам, а по характерной для каждого из них индивидуальной топологии. При наличии системы распознавания образов текст записывается в память микропроцессора уже не в виде *битовой карты*, а в виде кодов, и его можно редактировать обычными текстовыми редакторами.

Сканер подключается к параллельному порту ПК. Для работы со сканером ПК должен иметь специальный драйвер, желательно драйвер, соответствующий стандарту TWAIN. В последнем случае возможна работа с большим числом TWAIN-совместимых сканеров и обработка файлов поддерживающими стандарт TWAIN программами, например распространенными графическими редакторами Corel Draw, Max Mate, Picture Publisher, Adobe Photo Shop, Photo Finish. Большинство драйверов ориентированы на работу с локальным компьютерным интерфейсом SCSI.

Глава 6. ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ МИКРОПРОЦЕССОРОВ

6.1. Классификация команд

Все операции в микропроцессоре выполняются в двоичной системе счисления. Команды в программе тоже представляются в двоичной форме. Каждому типу микропроцессора присуща определенная система команд.

Однако составление программы в двоичных кодах – трудоемкая и сложная для человека задача. Программист должен знать двоичный или шестнадцатеричный код каждой команды, а таких команд бывает около сотни. Для упрощения процесса написания, отладки и чтения программы используют мнемонический или символический код. Каждую команду представляют простым трех- или четырехбуквенным мнемоническим символом (мнемоникой). Мнемоника подбирается так, чтобы она напоминала название команды. Написанную с помощью мнемоник программу транслируют в ее двоичный эквивалент. Это можно сделать вручную с помощью таблиц соответствия или используя специальную программу, которая называется *Ассемблером (программный транслятор)*. Каждый микропроцессор способен выполнять точно определенный для него набор команд. Программист ограничен только этим набором команд.

Классификация команд.

По числу ячеек памяти, необходимых для размещения одной команды, различают команды в одно, два или три слова. Команды длиной в два и три слова требуют для выборки соответственно два и три цикла обращения к памяти.

По функциональным признакам выделяют три большие группы команд: передачи данных, управления и обработки данных. Рассмотрим подробнее основные команды, при этом название команд приведем по-русски.

Команды передачи данных обеспечивают простую пересылку информации без выполнения каких-либо операций обработки. Существуют три вида операндов, участвующих в командах передачи: внутренние регистры микропроцессора, ячейки памяти, регистры устройства ввода-вывода, которые будем называть портами. Можно выделить следующие типы команд:

- ЗАГРУЗИТЬ (ПРОЧИТАТЬ), по которой содержимое одной из ячеек памяти засылается в регистр;
- ЗАПОМНИТЬ (ЗАПИСАТЬ), по которой содержимое регистра засылается в ячейку памяти;
- ПЕРЕСЛАТЬ, по которой содержимое одного регистра пересылается в другой;
- ЗАГРУЗИТЬ НЕПОСРЕДСТВЕННО, по которой в регистр записывается константа, указанная в коде команды;
- ВВОД, по которой содержимое порта засылается во внутренний регистр;
- ВЫВОД, по которой содержимое внутреннего регистра пересылается в порт.

Команды управления, часто называемыми командами перехода, позволяют выполнять различные действия в соответствии со значением внешних сигналов или выработанных внутри системы условий. Команды управления делятся на команды безусловного и условного перехода. К командам безусловного перехода относятся:

- БЕЗУСЛОВНЫЙ ПЕРЕХОД (БП), по которой в программный счетчик записывается содержимое адресного поля в коде команды, то есть обеспечивается переход по адресу, указанному в команде;
- ПРОПУСТИТЬ, по которой пропускается следующая команда в программе;
- БЕЗУСЛОВНЫЙ ПЕРЕХОД С ВОЗВРАТОМ (переход к подпрограмме), по которой в программный счетчик записывается новое содержимое (адрес первой команды подпрограммы), но в отличие от команды БП в памяти сохраняется текущее содержимое программного счетчика. После выполнения подпрограммы по ее последней команде ВОЗВРАТ восстанавливается содержимое программного счетчика.

Команды условного перехода проверяют состояние какого-либо разряда регистра, триггера или другого параметра. От результата проверки зависит, будет выполняться переход или нет. К командам условного перехода относятся:

- УСЛОВНЫЙ ПЕРЕХОД (УП) по адресу; в коде команды указывается проверяемое условие: нулевое или ненулевое значение результата, положительный или отрицательный знак результата, наличие или отсутствие сигналов переноса, переполнения и т.д. При выполнении условия обеспечивается переход в программе по адресу, указанному в команде, при невыполнении условия управление передается следующей команде программы;
- УСЛОВНЫЙ ПЕРЕХОД С ВОЗВРАТОМ;
- УСЛОВНО ПРОПУСТИТЬ;
- ЦИКЛ.

Команды обработки данных делятся на арифметические и логические. К арифметическим относятся команды:

- СЛОЖИТЬ содержимое двух регистров или регистра и ячейки памяти;

- ВЫЧЕСТЬ из содержимого ячейки памяти или регистра содержимое регистра;
- УВЕЛИЧИТЬ НА ЕДИНИЦУ (ИНКРЕМЕНТ) содержимое ячейки памяти или регистра;
- УМЕНЬШИТЬ НА ЕДИНИЦУ (ДЕКРЕМЕНТ) содержимое ячейки памяти или регистра;
- СЛОЖИТЬ С УЧЕТОМ ПЕРЕНОСА, по которой выполняется сложение с учетом состояния флага переноса, это позволяет легко организовать обработку чисел большой длины;
- ВЫЧЕСТЬ С УЧЕТОМ ЗАЕМА;
- СДВИГ содержимого ячейки памяти или регистра влево или вправо на один разряд.

К логическим относятся следующие команды:

- И (ЛОГИЧЕСКОЕ УМНОЖЕНИЕ) между двумя операндами;
- ИЛИ (ЛОГИЧЕСКОЕ СЛОЖЕНИЕ) между двумя операндами;
- НЕРАВНОЗНАЧНОСТЬ между двумя операндами;
- ИНВЕРСИЯ содержимого операнда.

6.2. Способы адресации

Операция выполняется над двумя числами – операндами. Порядок вычислений в микропроцессоре построен таким образом, что один операнд должен находиться в аккумуляторе. Откуда поступает второй операнд – зависит от *способа адресации*, используемой конкретной командой. Все команды микропроцессора можно разбить на группы по этому признаку.

Способы адресации, используемые в микропроцессоре, следующие:

- непосредственный;
- прямой;
- регистровый;
- косвенно-регистровый;
- неявный.

Существуют также сложные команды, выполняющиеся в несколько действий. В таких командах могут использоваться два способа адресации. Их относят к группе с *комбинированным способом адресации*.

Непосредственный способ адресации – операнд содержится непосредственно в команде. В двухбайтовых командах – во втором байте команды, в трехбайтовых – во втором и третьем байтах.

Прямой способ адресации – адрес ячейки памяти, где расположен операнд, указывается в самой команде. Если операнд располагается в ячейке памяти, тогда, помимо кода операции, команда содержит в себе его двухбайтовый адрес. Очевидно, такая команда состоит из трех байт. Младшая часть адреса указывается во втором байте команды, а старшая – в третьем байте.

Регистровый способ адресации – операнд отыскивается во внутреннем регистре микропроцессора. Команды с *регистровым способом адресации*

однобайтовые, потому что они не требуют данных и адресов вне микропроцессора.

Косвенно-регистровый способ адресации – адрес ячейки памяти, где расположен операнд, определяется содержимым парного регистра. Старший байт адреса находится в первом регистре пары, а младший – во втором регистре.

Неявный способ адресации – операнд определен существом команды и находится внутри микропроцессора.

Все *способы адресации*, не относящиеся к вышеперечисленным, называют *неявными*. Такие команды всегда однобайтовые, т.к. не требуют дополнительных данных для их выполнения.

6.3. Языки программирования

Языки программирования для микропроцессоров можно разделить на три основные уровня: *машинные, алгоритмические высокого уровня и ассемблера.*

Машинные языки находятся на самом нижнем уровне иерархии языков программирования. Будучи языками цифр, они неудобны для описания вычислительных процессов, требуют от программиста больших усилий для написания и отладки программ, хотя программа, написанная на машинном языке, сразу готова для исполнения микропроцессором.

Алгоритмические языки высокого уровня занимают верхнее положение в иерархии языков программирования. Форма языков высокого уровня приближена к привычной математической нотации, обеспечивает естественную форму описания вычислительных процессов. Недостатком является применение трансляторов, неэффективность получаемых после трансляции кодов.

Языки Ассемблера, занимая промежуточное положение между машинными языками и языками высокого уровня, объединяют в себе некоторые достоинства самого нижнего и самого верхнего уровней языков программирования. Транслятор программ с языка Ассемблера гораздо проще и компактнее транслятора для программ, написанных на языке высокого уровня, а результирующая машинная программа на выходе Ассемблера получается столь же эффективной, как и программа, которую написали на машинном языке.

6.4. Средства программирования

Имеется три класса средств программирования: редактирующие программы, транслирующие программы, моделирующие и отладочные программы. Каждый из этих классов облегчает выполнение некоторого этапа разработки программ. Все эти средства программирования делятся на кросс-средства и резидентные. *Кросс-средства* – это программы, которые выполняются не на микроЭВМ, для которого разрабатывается программное

обеспечение, а на какой-нибудь другой машине. *Резидентные средства* – это программы, которые выполняются на микропроцессоре, для которого разрабатывается программное обеспечение.

Редактирующие программы – это программы, облегчающие создание исходных программ. Эти программы позволяют производить набор исходных текстов, их корректировку. Эти программы оперируют с исходной программой как с текстом, совершенно не учитывая те синтаксические правила, которым должна удовлетворять программа.

Транслирующие программы обеспечивают получение объектной программы из программы, написанной на языке Ассемблера в среде какой-то редактирующей программы. Имеются два вида транслирующих программ: *компиляторы и интерпретаторы. Компиляторы* транслируют весь текст программы в машинный код в ходе одного непрерывного процесса. При этом создается полная программа в машинных кодах, которую затем можно выполнить без участия компилятора. *Интерпретатор* – это программа, осуществляющая пошаговую трансляцию входной программы с последующим исполнением машинной программы, полученной на каждом шаге трансляции.

Моделирующие и отладочные программы – это программы, позволяющие отлаживать объектную программу без участия самого микропроцессора, для которого эта программа предназначена. Моделирующие программы позволяют: оперировать и выводить на дисплей содержимое памяти моделируемой микроЭВМ и регистров микропроцессора, устанавливать контрольные точки останова, выполнять программу по шагам с приостановкой после каждой команды, выводить информацию о времени выполнения программы и т.д. Однако независимо от того, насколько хороша моделирующая программа, она никогда не может заменить полностью отладку программы на реальном микропроцессоре. Это объясняется тем, что специфические временные соотношения и условия внешнего окружения аппаратуры микропроцессора невозможно смоделировать полностью.

Глава 7. ОДНОКРИСТАЛЬНЫЕ МИКРОЭВМ СЕМЕЙСТВА МК48

7.1. Введение

В последнее время в микропроцессорной технике выделился самостоятельный класс БИС (большие интегральные схемы) – однокристальные микроЭВМ.

Однокристальные микроЭВМ конструктивно выполнены в виде одной БИС и включают в себя все устройства (см. рис. 7.1), необходимые для реализации цифровой системы управления минимальной конфигурации:

- 8 – разрядный центральный процессор, управляющий работой исполнительных устройств микроЭВМ и имеющий аппаратную поддержку

операций умножения и деления. Всего процессор выполняет 111 команд разрядностью в 1, 2 или 3 байта;

- ЗУ данных – внутренняя память данных объемом 128 байт, используемая для организации регистровых банков, стека и хранения пользовательских данных;
- ЗУ программ – внутренняя память программ объемом 4 Кбайта, расположена на кристалле;
- встроенный тактовый генератор, частота которого задается с помощью внешнего кварцевого резонатора, LC – цепочки или внешнего генератора;
- два 16 – разрядных многорежимных таймера – счетчика (TC0 и TC1), используемые для подсчета внешних событий, организации временных задержек и тактирования последовательного порта;
- двунаправленный дуплексный асинхронный последовательный приемопередатчик – последовательный порт;
- двухуровневая приоритетная система прерываний от четырех внутренних и двух внешних источников;
- 32 двунаправленые интерфейсные линии, индивидуально настраиваемые на ввод или вывод информации и организованные в виде трех 8 – разрядных параллельных портов ввода/вывода P0 – P2.

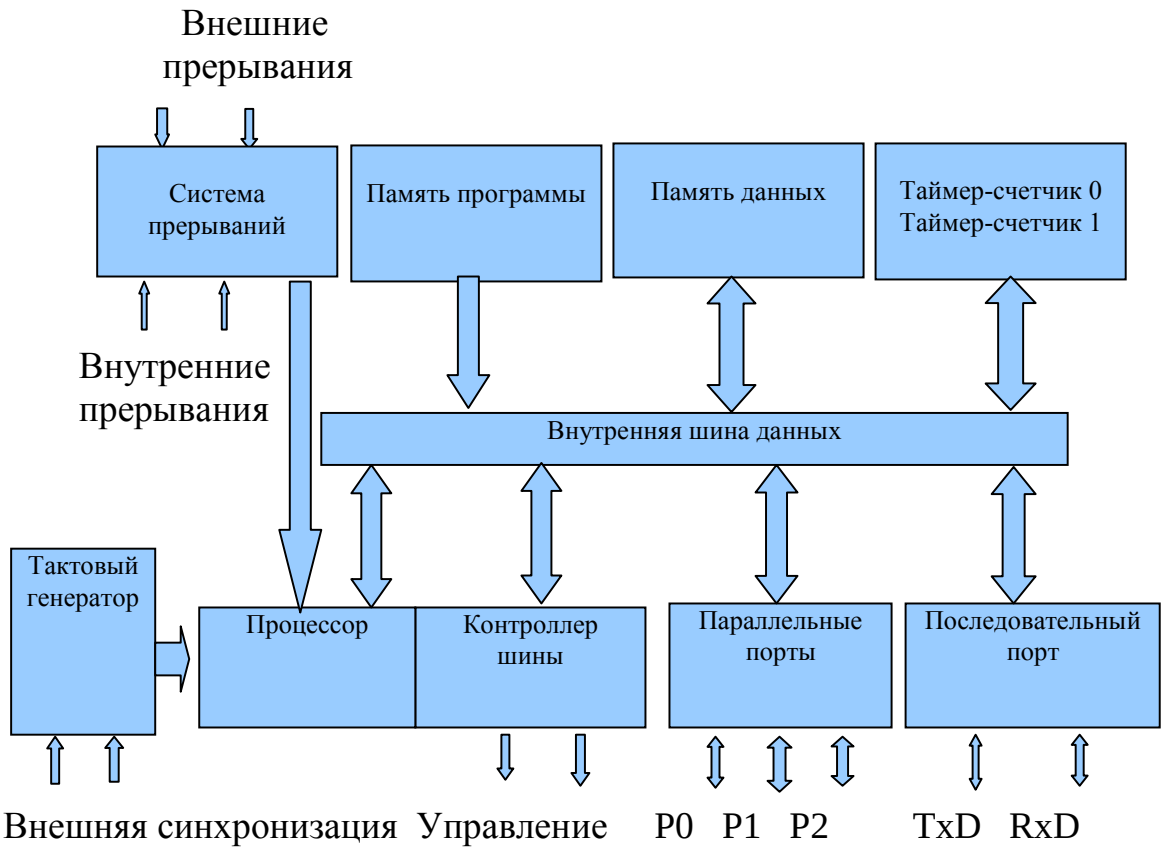


Рис. 7.1. Состав и структура микроЭВМ

Специфическая организация ввода/вывода информации, структурная организация, набор команд лучше всего приспособлены для решения задач

управления и регулирования в приборах, устройствах и системах автоматики, а не для решения задач обработки данных. В связи с этим микросхемы имеют незначительные емкости памяти, физическое и логическое разделение памяти программ и памяти данных, упрощенную и ориентированную на задачи управления систему команд. Правильнее было бы назвать эти приборы не микроЭВМ, а микроконтроллеры. В дальнейшем обозначение **МК48** будет обозначать класс микроконтроллеров, которые рассмотрим подробнее.

7.2. Общие сведения

Семейство МК48 включает ряд микросхем, полностью совместимых по системе команд, разводке выводов, совокупности функциональных основных устройств. Приведем таблицу наиболее распространенных микросхем.

Таблица 7.1

Микро схема	Аналог (Intel)	Объем внутренней памяти программ	Тип памяти программ	Объем памяти данных, байт	Макси-мальная тактовая частота, МГц	Ток потреб-ления, мА
КР1816 ВЕ35	8035	Нет	Внешняя	64	6,0	135
КР1816 ВЕ48	8748	1 Кбайт	УФППЗУ	64	6,0	135
КР1816 ВЕ39	8039	Нет	Внешняя	128	11,0	110
КР1816 ВЕ49	8049	2 Кбайта	ПЗУ	128	11,0	110
КР1830 ВЕ35	80С35	Нет	Внешняя	64	6,0	8
КР1830 ВЕ48	80С48	1 Кбайт	ПЗУ	64	6,0	8

МК48 имеют три восьмиразрядных порта ввода-вывода, которые могут обеспечивать обмен информацией с периферией по 24 независимым линиям, каждая из которых может быть настроена на ввод или вывод данных. Порты ввода-вывода имеют названия: P0, P1, P2.

Для возбуждения генератора тактовой частоты необходимо внешнее подключение кварцевого резонатора к двум выводам микросхемы. Частота кварцевого резонатора является задающей для формирования внутренней частоты синхронизации путем деления внешней частоты на три. При $f_{внеш}$=6 МГц, $f_{внутр}$=2 МГц. Сигнал внутренней частоты синхронизации, деленный на 5, поступает на счетчик машинных циклов. Таким образом, частота машинных

циклов составляет 0,4 МГц, длительность машинного цикла – 2,5 мкс. Все команды МК48 выполняются за 1-2 машинных цикла.

7.3. Условно-графическое обозначение (УГО). Назначение выводов

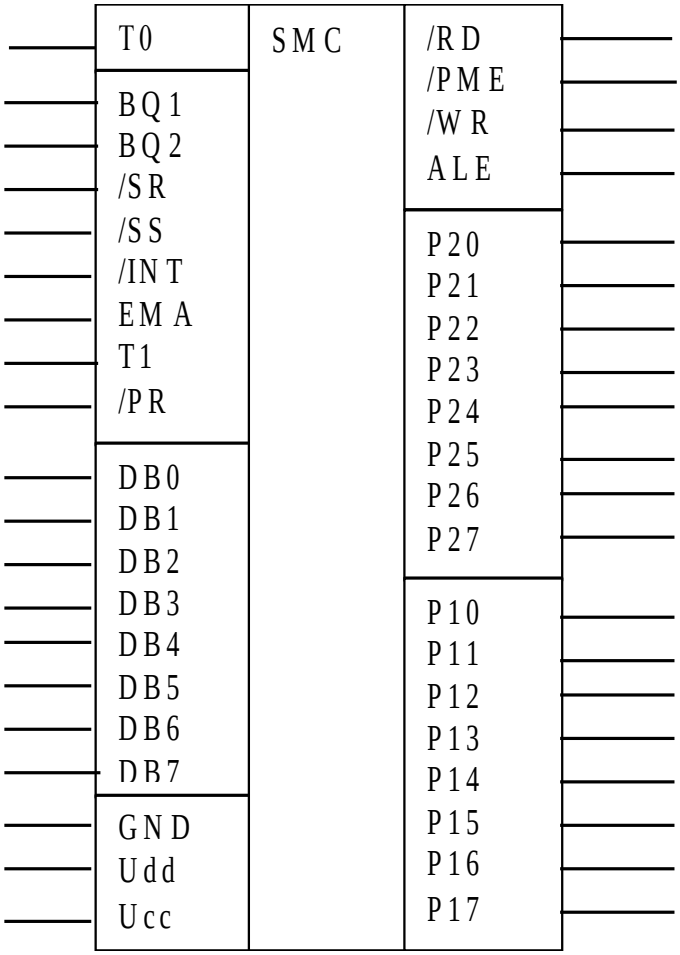


Рис. 7.2. Схема условно-графического обозначения (УГО)

На рис. 7.2 при описании выводов после обозначения вывода, в скобках, указан номер этого вывода.

T0 (1) – тестируемый вход. Вход может программно опрашиваться и, в зависимости от состояния этого входа, в программе могут быть организованы ветвления. Альтернативная функция этого вывода – использование его как выхода внутренних тактовых сигналов, частотой 2 МГц, если применить команду **ENTO CLK.** Сигнал этой частоты может быть использован для тактирования внешних устройств, входящих в микропроцессорную систему.

BQ1, BQ2 (2,3) – выводы, к которым подключается источник частоты тактирования (кварцевый генератор, LC – цепь или внешний сигнал).

/SR (4) – Вход сигнала сброса. Вывод предназначен для установки МК48 в начальное состояние.

/SS (5) – вывод, используемый для организации пошагового выполнения программ. Подача сигнала с уровнем “лог. 0” на этот вывод приводит к

приостановке работы МК48. С помощью специальных внешних схем можно использовать этот вывод для останова работы МК48 после выполнения каждой команды.

/INT (6) – вход внешнего прерывания. Подробнее о внешнем прерывании будет рассказано позже. Вход используется для организации работы МК48 в режиме реакции на какое-либо внешнее прерывание.

EMA (7) – вход переключения на режим работы с внешним ПЗУ. Как было видно из таблицы 7.1, не все типы ОМЭВМ имеют внутреннюю память программ. Некоторые из них могут использоваться только с внешней памятью программ. В МК48 имеется возможность работы как с внутренней, так и с внешней памятью программ. Подача на вывод ЕМА сигнала с уровнем "лог. 1" инициирует режим работы микросхемы с внешним ПЗУ, а "лог. 0" с внутренним ПЗУ.

T1 (39) – тестируемый вход. Как тестируемый вход этот вывод аналогичен выводу Т0. Альтернативная функция этого вывода – это использование вывода как входа счетчика внешних событий. Под внешним событием понимается изменение состояния сигнала на этом выводе, а именно переход состояния сигнала из единицы в нуль. В структуре МК48 имеется таймер – счетчик, который может быть запрограммирован на режим работы счетчика. В режиме счетчика он подсчитывает внешние события.

/PR (25) – вход программирования для микросхемы КР1816ВЕ48.Эта микросхема имеет внутреннюю память программ с ультрафиолетовым стиранием. Процесс записи программы в эту память называется программированием. Вывод используется для программирования микросхемы.

DB0 … DB7 (12…19) – шина данных (двунаправленный порт Р0). Один из трех портов ввода/вывода. Но в отличие от других портов ввода/вывода имеет дополнительную функцию магистрали адреса/данных, когда используются внешняя память программ или внешняя память данных.

P10…P17 (27…34) – двунаправленный порт ввода/вывода Р1.

P20…P27 (21…24,36…39) – двунаправленный порт ввода/вывода Р2. Линии Р20…Р23 используются как линии адреса при работе с внешней памятью программ.

/RD (8) – чтение внешней памяти данных.

/WR (10) – запись во внешнюю память данных. Эти два сигнала вырабатываются МК48 в тех случаях, когда эта микросхема обращается к внешней памяти данных.

/PME (9) – разрешение чтения внешней памяти программ.

ALE (11) – разрешение фиксации адреса. Эти два сигнала вырабатываются МК48 в тех случаях, когда микросхема обращается к внешней памяти программ.

Ucc – напряжение питания +5В.

Udd – напряжение резервного питания ОЗУ для КР1816ВЕ49/ВЕ39.

GND – общий.

7.4. Структурная организация МК48

Основой структуры МК48 является процессор. Основа процессора – 8 - разрядное **АЛУ**, которое выполняет арифметические, логические, сдвиговые операции над данными в двоичном и двоично-десятичном коде. Основной элемент АЛУ – восьмиразрядный аккумулятор. Имеется схема десятичной коррекции (**СДК**), предназначенная для обработки двоично-десятичных данных.

7.4.1. Работа схемы двоично-десятичной коррекции

Представления чисел в двоичном и двоично-десятичном виде отличаются. Речь пойдет о числе, занимающем 8 двоичных разрядов (байте). В двоично-десятичном (**BCD**) представлении байт разбивается на два полубайта (тетрады) и в каждую тетраду записывается цифра в двоичном виде. Например, число 35, записанное в BCD-формате, будет иметь вид: **00110101.** Как видно из примера, в старшей тетраде записана цифра 3, а в младшей цифра 5. Для записи числа, имеющего большую разрядность, потребуется и больше тетрад. Диапазон BCD-чисел, занимающих один байт, равен 0…99. Еще примеры: десятичное число 83=10000011 в BCD-формате,

64=01100100.

Обратите внимание, что эти числа в двоичной системе будут иметь другое представление: 83=01010011 64=01000000.

При выполнении арифметических операций над BCD-числами, так как процессор представляет числа только в двоичном виде, может получиться неверный результат. Например, сложим два числа, представленные в BCD-формате. Это числа 69 и 27.

69(BCD)=01101001, 27(BCD)=00100111.

Процессор произведет сложение по правилу сложения двоичных чисел и получит в результате: $_{+}$ 01101001

00100111

10010000

Если продолжать рассматривать результат в BCD-формате, то получим неверный результат. Получили 90(BCD), а должны были получить 96(BCD).

Другой пример.

19(BCD)=00011001,66(BCD)=01100110.

$_{+}$ 00011001

01100110

01111111

Получилось число, младшая цифра которого выходит за пределы максимального представления чисел в BCD-формате.

Для того чтобы и в первом и во втором примере скорректировать результат, то есть получить верное BCD-представление числа, используется

схема десятичной коррекции, которая исправляет результат любой арифметической операции по команде **DA A.**

По команде **DA A** выполняется:

1. Если младшая тетрада результата больше 9 или при выполнении операции был перенос из 3 разряда в 4, то к младшей тетраде добавляется 6(0110).

В первом примере был перенос из 3 разряда в 4. Поэтому команда **DA A** добавит к результату 0110, и правильный результат будет 10010110. Для второго примера после добавления 0110, так как младшая тетрада больше 9, получится результат

$$+\begin{array}{r} 01111111 \\ \underline{\quad 0110} \\ 10000110 \end{array}$$

, равный в BCD-представлении 85.

2. Если старшая тетрада результата больше 9 или был перенос из 7 разряда, то к старшей тетраде добавляется 0110=6.

Вывод. Если в программе числа представлены в BCD-формате, то после арифметических команд в программе необходимо вставлять команду **DA A.**

7.4.2. Счетчик команд СК и регистр состояния программы PSW

В состав процессора входят также счетчик команд (**СК**) и регистр состояния программы (**PSW**).

Счетчик команд предназначен для формирования текущего адреса местонахождения команды в памяти программ.

Разрядность счетчика команд равна 12. Счетчик команд всегда хранит адрес обращения к памяти программ и позволяет адресоваться к 4 килобайтам памяти программ (2^{12}=4096).

МК48 имеет особенность адресации памяти программ, связанную с тем, что все доступное адресное пространство разбито на два банка: банк 0 и банк 1. Адресное поле банка 0=000H-7FFH, банка 1=800H-FFFH. Выбор соответствующего банка памяти программ производится по командам:

SEL MB0 - выбор банка 0

SEL MB1 – выбор банка 1.

При этом происходит занесение 0 или 1 в старший разряд счетчика команд.

Регистр состояния программы (PSW) предназначен для хранения данных о состоянии МК48. Формат регистра PSW имеет вид:

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

C – Бит переноса, указывающий на переполнение аккумулятора после предыдущей операции.

AC – бит полупереноса (перенос из 3 разряда в 4), используется в команде **DA A.**

F0 – флаг пользователя, используется по команде условного перехода.

BS – бит выбора банков регистров общего назначения. Смотри раздел “Память данных”.

S2, S1, S0 – указатель стека. 3-хразрядное двоичное число, указывающее адрес вершины стека. Организации стека посвящен специальный раздел.

Существует команда **MOV A, PSW**, по которой содержимое PSW пересылается в аккумулятор. Так как с числом в аккумуляторе можно проделывать любые модификации, то любой бит регистра PSW можно модифицировать. Модифицированное в аккумуляторе значение регистра PSW возвращается обратно по команде **MOV PSW,A.**

7.4.3. Память программ

Память программ предназначена для хранения и считывания команд. Общий объем адресуемой памяти программ 4 кбайта. Но в зависимости от типа МК-48 память программ разделена на внутреннюю или резидентную (РПП) объемом 1 кбайт или 2 кбайта и внешнюю (ВПП), составляющую в сумме с резидентной памятью – 4 кбайта. Если внутренней памяти программ достаточно, то внешняя память может не использоваться вообще.

Таблица 7.2

Распределение памяти программ

Диапазон адресов отдельных областей памяти		Содержание областей памяти
Десятичный вид	Шестнадцате-ричный вид	
4095	FFFH	Внешняя память программ для 1816ВЕ49. Банк МВ!
2048	800H	
2047	7FFH	Внешняя память программ для 1816ВЕ48,1830ВЕ48. Внутренняя память программ для 1816ВЕ49. (Отсюда и ниже – Банк МВ).
1024	400H	
1023	3FFH	Отсюда и ниже – внутренняя память программ для 1816ВЕ48,1830ВЕ48,1816ВЕ49.
7	7H	Вектор прерывания от таймера – счетчика
3	3H	Вектор внешнего прерывания
0	0H	Адрес начальной установки

Микросхемы 1816ВЕ35, 1816ВЕ39, 1830ВЕ35 вообще не имеют внутренней памяти программ и могут использоваться только с внешней

памятью программ. Следует напомнить, что переключение памяти программ с одного банка на другой осуществляется командами **SEL MB0, SEL MB1.** При этом старший разряд счетчика команд устанавливается или сбрасывается по первой команде **JMP** или **CALL,** следующей за командой **SEL MB0** или **SEL MB1**.

Память программ кроме деления на банки делится на страницы по 256 байт в каждой. Например, в командах условного перехода задается 8 – битный адрес передачи управления в пределах текущей страницы (смотри систему команд). Как видно из таблицы распределения памяти программ, имеется три ячейки специального назначения:

Адрес 0 – адрес, по которому выполняется первая выборка команды после сброса, т.е. по этому адресу расположена первая команда программы.

Адрес 03 – начальный адрес подпрограммы, вызываемой по сигналу внешнего прерывания INT, если прерывание разрешено.

Адрес 07 – начальный адрес подпрограммы, вызываемой по переполнению внутреннего таймера – счетчика (прерывание от таймера – счетчика), если это прерывание разрешено.

Таким образом, МК48 имеет два вида прерываний и для каждого вида прерываний имеется свой вектор, т.е. адрес в памяти программ, где располагается начальная команда программы обработки прерывания. В связи с этим структура программы на языке Ассемблера выглядит так (если оба прерывания используются, см. таблицу 7.3):

Таблица 7.3

Адрес	Команда	Комментарии
0000H	JMP START	Переход на начало основной программы
0003H	Подпрограмма обработки прерывания по входу INT	
0007H	Подпрограмма обработки прерывания по T\C	
START	Основная программа	

7.4.4. Память данных

Память данных предназначена для записи, хранения и считывания данных, получаемых в процессе обработки информации. Память данных может быть внутренней (резидентной) (**РПД**) или внешней (**ВПД**).

Таблица 7.4

Распределение внутренней памяти данных

Адрес ячейки памяти данных		Содержание	Обозначение (имя) регистров	Наличие прямой адресации	Наличие косвенной адресации
Десятичный	Шестнадцатеричный				
63(127)	3FH(7FH)	ОЗУ данных 32·8 (96*8)	____	Нет	Есть
…	…				
32	20H				
31	1FH	Банк RON1 (8·8)	R7	Есть	Есть
…	…		…		
24	18H		R0		
23	17H	8-уровневый стек 8·16 или ОЗУ данных	____	Нет	Есть
…	…				
8	08h				
7	07H	Банк RON0 (8·8)	R7	Есть	Есть
6	06H		R6		
5	05H		R5		
4	04H		R4		
3	03H		R3		
2	02H		R2		
1	01H		R1		
0	0H		R0		

Как видно из таблицы, имеется два банка рабочих регистров: банк RON0 и банк RON1. Размер каждого банка 8 байт. В распоряжении программиста, таким образом, 16 байтов с прямой адресацией, но не все они доступны одновременно. Одновременно доступен либо банк RON0, либо банк RON1. Команды, в которых имя регистра расположено в команде, называются командами с прямой адресацией, а регистры R0 – R7 – это регистры с прямой адресацией.

Например: 1**) MOV A, R0** – переслать содержимое регистра R0 в аккумулятор. Здесь прямо указан регистр, участвующий в команде.

2) MOV R1, A

3) MOV R7, A

Схема вышеприведенных команд:

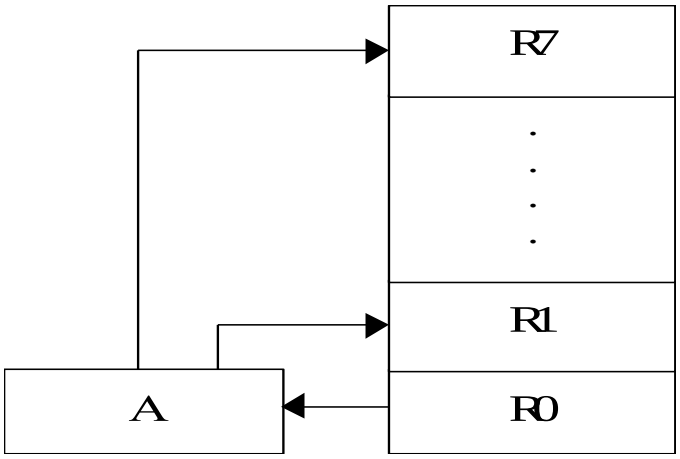


Рис. 7.2

По этим командам производится обмен данными между регистрами банка RON0 и аккумулятором. Каким образом аккумулятор может обменяться с регистрами банка RON1? По тем же командам, но предварительно нужно сменить банк регистр RON0 на RON1 по команде **SEL RB1.**

1) **SEL RB1**
2) **MOV A, R0**
3) **MOV R1, A**

Схема пересылок:

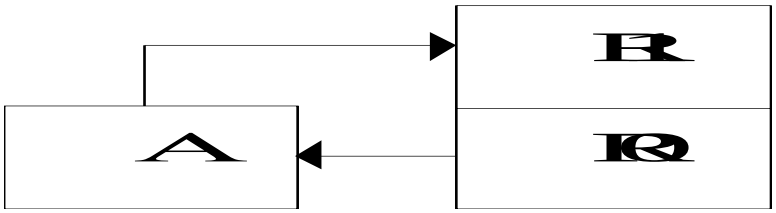


Рис. 7.3

Следующая программа пересылает значение R0 RON0 в R7 RON1 , а R0 RON1 в R7 RON0.

SEL RB0
MOV A, R0
SEL RB1
MOV R7, A
MOV A, R0
SEL RB0
MOV R7, A

Обычно основная работа программы происходит с банком RON0 и после сброса МК48 автоматически проходит команда SEL RB0, т.е. назначается банк

RON0. Банк RON1 удобно использовать при наличии подпрограмм в программе. Если в основной программе и в подпрограмме используются одни и те же регистры R0…R7, то при переходе к подпрограмме удобно перейти к банку RON1, а в основной программе использовать банк RON0.

Например:

MOV R5, #26H

CALL PP1 ; команда перехода на подпрограмму с именем (меткой) PP1.

… ; здесь и далее R5 из банка RON0=26H.

…

…

PP1: SEL RB1

MOV R5, #39H

… ; здесь и до команды выхода из подпрограммы R5 из банка RON1=39H

…

SEL RB0

RET ; выход из подпрограммы.

Регистры общего назначения R0…R7 обоих банков памяти данных – это прямо адресуемые регистры. Все остальные ячейки памяти данных не имеют имен, а имеют адреса. Как работать с такими ячейками? Считается, что именем ячейки является ее адрес. Адрес, как видим, 8-разрядное число 20H…3FH. Если бы была реализована прямая адресация, то команда MOV A,21H означала бы: содержимое ячейки с именем 21H (адресом 21H) переслать в аккумулятор, но эта возможность в данной системе команд не реализована. Чтобы выполнить эту операцию, используется команда

MOV A, @R_i , где i=0 или 1, т.е. может быть **MOV A, @R0** или **MOV A, @R1.**

Знак @ означает, что это не **MOV A, R0** или **MOV A, R1.**

А где же в этой команде имя (адрес) ячейки 21H? Адрес ячейки необходимо предварительно поместить в R0 или R1, т.е.

MOV R0, #21H

MOV A, @R0

Это означает, что содержимое ячейки с адресом 21H пересылается в аккумулятор.

MOV R1, #30H

MOV @R1, A означает: содержимое аккумулятора пересылается в ячейку памяти с адресом 30H.

В этих командах R0 и R1 – регистры косвенной адресации. Сочетания @R2…@R7 недопустимы. Косвенную адресацию можно применить к любой ячейке внутренней памяти данных.

Восьмиуровневый 16-разрядный стек с адресами от 08H до 17H адресуется указателем стека из PSW.

Таблица 7.5

Организация стека

Указатель стека SP из регистра PSW	Содержимое ячеек стека (приведено для ячеек 08H и 09H, для других – аналогично)	Адрес ячейки стека
111		16H
110		14H
…		…
001		0AH
000PSW (7…4)	СК(11…8)	09H
СК(7…4)	СК(3…0)	08H

В таблице PSW (7…4) – означает 4…7 разряды регистра PSW, СК (3…0) – 0…3 разряды счетчика команд СК и так далее.

Стек используется в следующих случаях:

А) при выполнении команд **CALL addr** – переход на подпрограмму, где addr – адрес начала подпрограммы. **RET** – возврат из подпрограммы.

При выполнении команды **CALL addr** содержимое счетчика команд (все 12 разрядов) запоминается в стеке по адресу, на который указывает указатель стека SP. Если SP=000, то идет заполнение 8 и 9 ячеек ОЗУ. Туда же заносятся 4…7 разряды регистра состояния программы (PSW). Далее SP увеличивается на единицу для новой записи в стек, т.е. по следующей команде **CALL addr** будут заполняться ячейки 10,11 ОЗУ, т.к. на них указывает указатель стека SP и так далее. По команде RET содержимое SP уменьшается на единицу и из ячеек ОЗУ, соответствующих полученному SP извлекается содержимое СК и четырех разрядов регистра PSW.

Вывод: стек предназначен для временного хранения адресов возврата из подпрограмм, причем стек МК48 позволяет одновременно хранить 8 адресов возврата максимум. Это означает, что в программе не должно быть более 8 вложений подпрограмм друг в друга. Вложение одной подпрограммы в другую – это случай, когда осуществляется переход на подпрограмму В (условное имя) из области подпрограммы А (условное имя). Говорят, что стек МК48 имеет 8 уровней вложения. Рассмотрим пример использования стека с вложениями одной подпрограммы в другую.

1) …
2) CALL PPA
3) …
4) …
и т.д. продолжение программы.
100) PPA: …

101) CALL PPB
102) …
103) …
…
110) RET
111) PPB: …
112) …
113) …
…
119) RET

В этой программе можно выделить три области. Область, охватывающая команды с адресами 1…99, – это область основной программы. Область команд с адресами 100…110 – подпрограмма с условным именем PPA. Область команд с адресами 111…119 – подпрограмма с условным именем PPB. Размеры областей могут иметь различное значение. Область подпрограммы начинается с команды, отмеченной меткой – названием подпрограммы, в нашем случае – это команды 100 и 111. Каждая подпрограмма заканчивается специальной командой **RET** – возврат (выход) из подпрограммы. Как правило, подпрограммы располагаются сразу за основной программой по порядку, одна за другой. Рассмотрим использование стека при выполнении команд вышеприведенной программы. Указатель стека SP = 0. Выполнение команды по адресу 2 (CALL PPA) происходит в несколько этапов. В ячейки стека 08H и 09H заносится содержимое счетчика команд, увеличенное на 1, для нашего случая – это число 3, а также 4 разряда регистра PSW. Указатель стека увеличивается на 1, то есть принимает значение 1. Счетчик команд (СК) принимает значение адреса той команды, которая помечена названием PPA, то есть 100. Таким образом, программа переходит на выполнение команды с адресом 100, то есть на выполнение подпрограммы PPA. При выполнении команды с адресом 101 (CALL PPB) в ячейки 0AH и 0BH заносится содержимое СК, увеличенное на 1, то есть 102 и 4 разряда регистра PSW. Указатель стека увеличивается на 1, то есть становится равным 2.

Счетчик команд принимает значение адреса команды, помеченной названием PPB, то есть 111. Не выходя из подпрограммы PPA, программа вошла в подпрограмму PPB. Это и есть вложение одной подпрограммы в другую. На данный момент стек хранит и адрес возврата из подпрограммы PPA (3), и адрес возврата из подпрограммы PPB (102). Выполнение подпрограммы PPB заканчивается командой с адресом 119 (RET). По этой команде указатель стека уменьшается на 1, становится равным 1, поэтому именно из ячеек 0AH, 0BH, а ни из каких других в счетчик команд переписывается число, там хранящееся, а именно 102. Это означает, что будет выполняться команда с адресом 102, то есть продолжится выполнение подпрограммы PPA. Последняя команда подпрограммы PPA имеет адрес 110 (RET). При выполнении этой команды указатель стека уменьшится на 1 и станет равным 0, поэтому именно из ячеек 08H, 09H в счетчик команд перепишется значение, равное 3. Это

значит, программа продолжит свое выполнение с адреса 3, то есть продолжится выполнение основной программы. Следует обратить внимание на то, что в правильно написанной программе число вхождений в подпрограмму должно быть равно числу выходов из подпрограммы так, чтобы указатель стека в конце программы был равен 0.

Б) Второй случай использования стека:

При прерываниях, аналогично команде **CALL addr**, в стек заносится содержимое СК и 4…7 разрядов PSW. По команде **RETR** – возврат из подпрограммы обработки прерывания - данные СК восстанавливаются, а также восстанавливается содержимое 4…7 разрядов PSW.

В МК48 предусмотрена возможность расширения внутренней памяти данных за счет внешней, максимальный размер которой 256 байт путем подключения внешних микросхем ОЗУ (см. раздел “Работа с внешней памятью”).

7.4.5. Каналы ввода – вывода

Каналы ввода/вывода служат для организации обмена информацией МК48 с внешними устройствами. Линии ввода/вывода организованы в три 8-ми-разрядных порта **P0, P1, P2**. Порты P1, P2 в режиме вывода фиксируют выводимые данные в триггерах – защелках. Каждая линия независима и имеет свой триггер – защелку. Эти данные сохраняют свое значение до следующей выдачи данных в порт.

В состояние ввода порты P1, P2 устанавливаются после сброса SR, а также если на разряды порта выдана “логич. 1”. После этого можно считывать содержимое выводов порта. Возможна смешанная настройка линий порта: одних на ввод, других, на вывод.

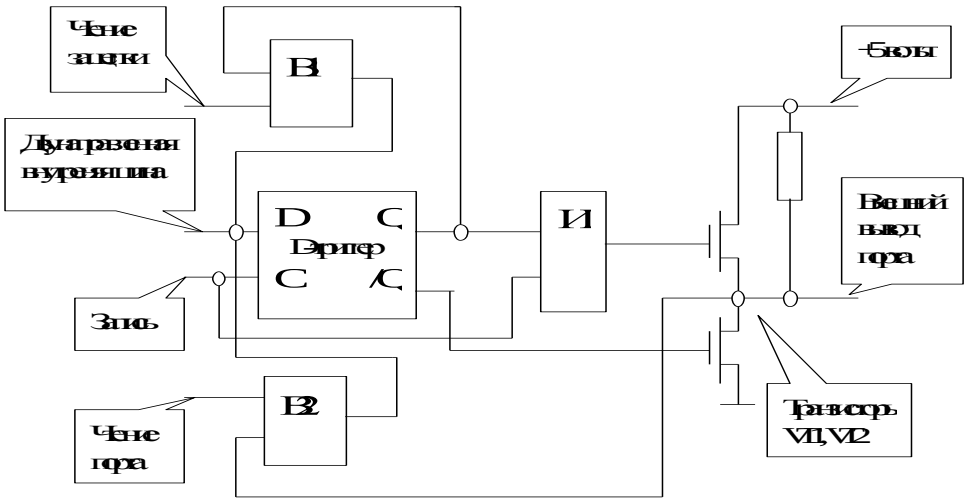


Рис. 7.4. Электрическая схема одной из линий порта P1, P2

Внутренняя шина – двунаправленная. D – триггер – триггер-защелка. Состояние триггера-защелки определяет состояние выхода. Если Q=1, /Q=0 то

транзистор VT2 закрыт, выход запитан через резистор R и находится в состоянии ”1”.

Если Q=0, /Q=1 , то транзистор VT2 открыт и на выходе устанавливается состояние ”0”.

Запись в триггер – защелку осуществляется через внутреннюю шину по внутреннему сигналу ”Запись”. Устройство В1 предназначено для реализации чтения состояния триггера – защелки, по внутреннему сигналу ”чтение защелки”. Устройство В2 предназначено для чтения состояния линии порта по внутреннему сигналу ”чтение порта”.

По схеме видно, что для того чтобы использовать вывод как вход, нужно, чтобы транзистор VT2 был закрыт, иначе он шунтирует вывод на общий провод. Запись в триггер – защелку ”1” позволяет использовать вывод как вход.

Устройство И обеспечивает включение транзистора VT1 на короткое время ≈ 0.4 мкс при изменении содержимого защелки с ”0” на ”1” для формирования фронта нарастания сигнала на выводах порта. После выключения транзистора уровень ”1” поддерживается на выводе порта за счет резистора R1. $R_{откр.\ тр\text{-}ра} \leq 5$ кОм. R1 = 50 кОм.

Рассмотрим работу схемы в режиме вывода в линию порта 0 или 1.

При выводе 1 уровень логической единицы помещается на внутреннюю шину, на линию “запись” выдается короткий положительный стробирующий импульс, который своим передним фронтом производит запись единицы в D – триггер. Выход Q триггера становится равным 1, выход /Q – 0. Сигнал с выхода Q разрешает прохождение импульса “запись” через логический элемент “И” на затвор полевого транзистора VT1, который открывается на время действия этого импульса. Сигнал 0 с выхода /Q, поступая на затвор полевого транзистора VT2, закрывает его. На линии порта устанавливается состояние логической единицы.

При выводе 0 транзистор VT1 не открывается, а открывается транзистор VT2, на линии порта устанавливается потенциал, близкий к нулевому.

При вводе информации из линии порта состояние линии порта поступает на устройство В2 и по сигналу “чтение порта” поступает на внутреннюю шину и далее в аккумулятор МК48.

Перечислим команды, которые работают с портами Р1, Р2.

OUTL P1, A

OUTL P2, A

По этим командам содержимое аккумулятора записывается в триггера-защелки портов и присутствует на выводах портов.

IN A, P1

IN A, P2

По этим командам в аккумулятор вводится состояние на выводах портов. Предварительно в триггеры-защелки портов должны быть записаны “лог. 1”.

Следующие команды – это команды “чтение – модификация – запись”.

ANL P1, #data8

ANL P2, #data8

Эта команда сложная, выполняется в три этапа. Во-первых, читается содержимое триггера – защелки по внутренней команде “чтение защелки”. Во-вторых, это значение логически умножается на константу #data8. В-третьих, полученное значение записывается в порт. Таким образом, можно модифицировать содержимое порта. Команду типа **ANL** удобно использовать, когда необходимо обнулить некоторые разряды порта. Для этого константа data8 должна иметь нули в разрядах, которые обнуляются. Например, имеем состояние порта P1, работающего на вывод, равное 01011111. Необходимо установить в нулевое состояние разряды 1 и 3 порта. Для этого необходимо выполнить команду **ANL P1, #11110101B**. При этом остальные разряды не меняют своего состояния. Продемонстрировать.

ORL P1, #data8

ORL P2, #data8

Команды, подобные предыдущим, только здесь используется логическое сложение. Заметим, что данную команду можно использовать для установки в “1” отдельных разрядов портов. Например, команда **ORL P2, #10000001B** установит в “1” 0 и 7 разряды порта P2.

Порт P0 (шина DB) в отличие от портов P1,P2 имеет дополнительные функции. Эти функции связаны с использованием микроЭВМ в расширенных системах, то есть когда используется внешняя память программ или данных. Если порт P0 используется как статический порт ввода/вывода, подобно портам P1,P2, то вывод из него выполняется по команде **OUTL BUS, A**, а ввод по команде **INS A, BUS**. В отличие от портов P1,P2 порт P0 допускает только побайтовый обмен, когда по всем линиям порта производится либо ввод, либо вывод. Кроме этого имеются команды типа “чтение – модификация – запись”.

ANL BUS, #data8

ORL BUS, #data8.

Отметим, что порт P0 все же своей основной функцией имеет функцию работы с внешней памятью, поэтому рассмотрим режимы работы микроЭВМ с внутренней и внешней памятью.

7.5. Режимы работы

7.5.1. Режим работы с внутренней памятью программ

Если внутренней памяти микроЭВМ достаточно для реализации возложенных на нее функций, то используется этот режим. Этот режим устанавливается в микроЭВМ при подаче напряжения “лог. 0” на вывод EMA. В этом режиме микроЭВМ не формирует никакие внешние управляющие сигналы, за исключением сигнала ALE. То есть все процессы проходят внутри микроЭВМ. Сигнал ALE вырабатывается в каждом машинном цикле, и имеет период 2,5 мкс. Здесь и далее имеется в виду, что частота кварцевого генератора равна 6 МГц.

Если микросхема имеет внутреннюю память программ и вывод EMA подключен к “лог. 0”, то обращение к внутренней памяти программ производится до последнего адреса внутренней памяти, далее автоматически начинается обращение к внешней памяти.

7.5.2. Режим работы с внешней памятью программ

При подключении вывода EMA к напряжению “лог. 1” обеспечивается режим работы только с внешней памятью программ.

Временная диаграмма работы микросхемы с внешней памятью программ:

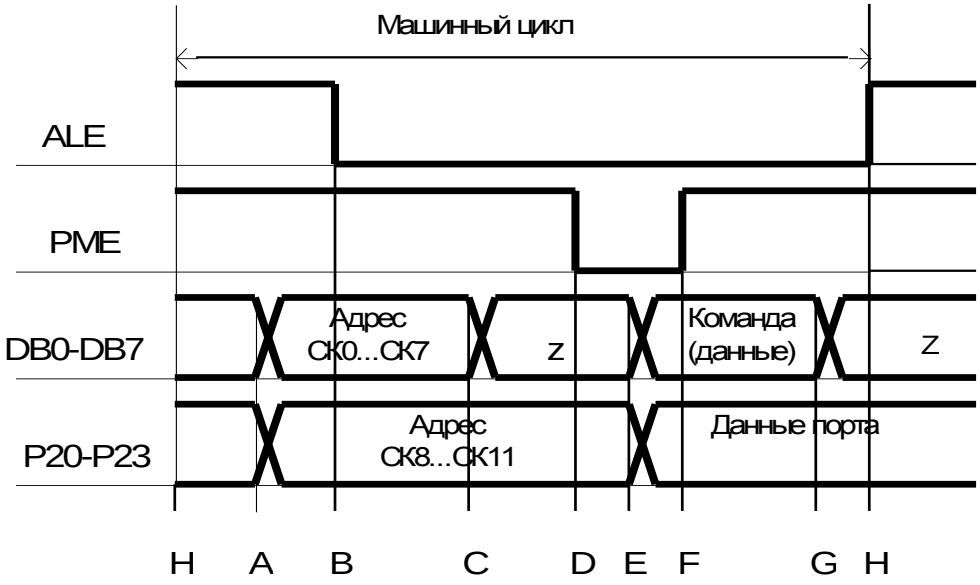


Рис. 7.5. Временная диаграмма работы микросхемы
с внешней памятью программ

Разъяснения по поводу временной диаграммы проведем после рассмотрения структурной схемы, реализующей режим работы с внешней памятью программ.

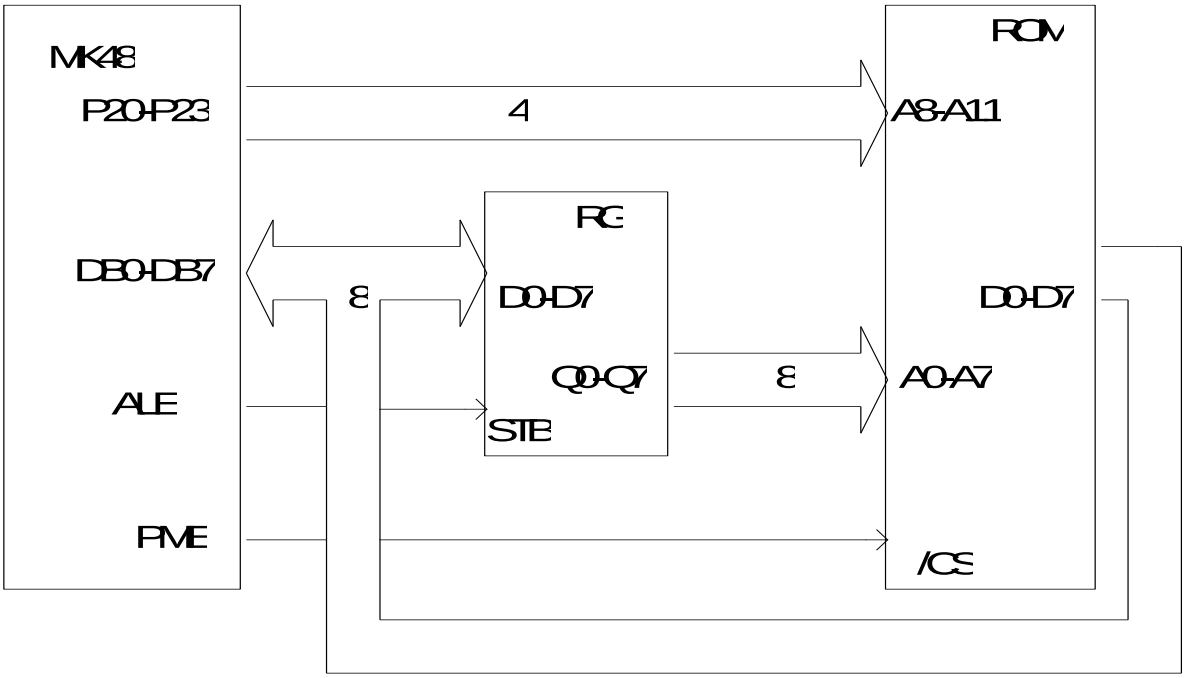


Рис. 7.6. Структурная схема, реализующей режим работы
с внешней памятью программ

Схема состоит из трех микросхем: микроЭВМ МК48, постоянного запоминающего устройства ROM и регистра RG типа регистра-защелки. Регистр-защелка работает следующим образом. Восьмиразрядная шина D0-D7 – вход регистра, шина Q0-Q7 – выход, вход STB управление режимом работы. Если на входе STB присутствует “лог. 1”, то данные, приходящие на вход регистра, беспрепятственно поступают на выход Q0-Q7. Любое изменение данных на входе вызывает аналогичное изменение данных на выходе. В этом режиме регистр работает как буфер. В момент перехода сигнала на входе STB из единичного состояния в нулевое данные защелкиваются в регистре, то есть то, что было на входе в данный момент, фиксируется на выходе и остается неизменным, пока STB=0.

ПЗУ работает следующим образом. Шина A0-A11 – входы адресных сигналов. Информация на этой шине однозначно определяет ячейку памяти, к которой происходит обращение. Данные из выбранной ячейки памяти поступают на шину D0-D7 – выход данных. Управление этим процессом происходит по входу /CS. При /CS=1 выборка данных из памяти запрещена, а шина D0-D7 находится в третьем состоянии (состояние Z). Выборка и появление данных на шине D0-D7 возможна только при /CS=0. Объем памяти ПЗУ выбирается из требований к устройству. Он не может быть больше 4Кб, это объем памяти, который может максимально адресовать МК48. Для того чтобы адресовать 4 Кб, требуется 12 адресных линий, и в ПЗУ это линии A0-A11. В МК48 тоже должны быть эти линии, через которые микроЭВМ могла бы послать адрес команды в ПЗУ. МК48 использует для формирования 12-разрядного адресного слова универсальную шину DB (8 линий) и 4 линии порта P2 (P20-P23).

Подробно опишем временную диаграмму.

Отрезок времени между двумя точками H – это машинный цикл. В течение этого времени происходит извлечение из памяти одного байта, байт из памяти поступает в микроЭВМ.

Точка **A** – микроЭВМ устанавливает на шинах DB0 - DB7 и P20 – P23 адрес ячейки памяти – 12 разрядов. Так как на входе STB присутствует “лог. 1”, адресное слово поступает на входы A0 – A11 ПЗУ. Но выборки данных из памяти не происходит, так как на входе /CS “лог. 1”.

B – МикроЭВМ снимает сигнал ALE, и младшие разряды адреса СК0 – СК7 фиксируются в регистре.

C – микроЭВМ убирает с шины DB0 - DB7 младшие разряды адреса, так как они уже зафиксировались в регистре. На шине DB0 – DB7 устанавливается третье состояние. Далее шина DB0 – DB7 будет использована для ввода в микроЭВМ данных из памяти.

D – МикроЭВМ сбрасывает сигнал PME, разрешая выдачу данных на выход ПЗУ.

E – на шине DB появляются данные с выхода ПЗУ.

F – микроЭВМ принимает данные с шины DB и устанавливает сигнал PME.

G – так как сигнал PME стал равным единице на выходе ПЗУ (на шине DB), устанавливается третье состояние.

Процесс чтения байта из памяти программ завершен.

7.5.3. Режим работы с внешней памятью данных

Память данных используется для временного хранения некоторой базы данных. Если объем одновременно хранимой базы данных больше объема внутренней памяти данных микроЭВМ (максимум 128 байт), то можно использовать внешнюю память данных, то есть ОЗУ, имеющую объем до 256 байт. Таким образом, объем внешнего ОЗУ может иметь максимальное значение, равное 256 байтам. Размер адресуемой памяти данных ограничивается разрядностью шины DB, посредством которой осуществляется обращение со стороны микроЭВМ к внешней памяти.

Если используется внешняя память данных, то обращение к ней производится с помощью команд типа **MOVX,** которые рассмотрим в дальнейшем. Поскольку внешняя память данных – это область ОЗУ, то есть область памяти, куда можно записать данные и откуда данные можно считать, то должны быть соответствующие команды для выполнения этих операций, а в МК48 предусмотрены специальные выводы для формирования сигналов, управляющих процессом чтения или записи.

Рассмотрим структурную схему подключения внешней памяти данных, объемом до 256 байт к МК48.

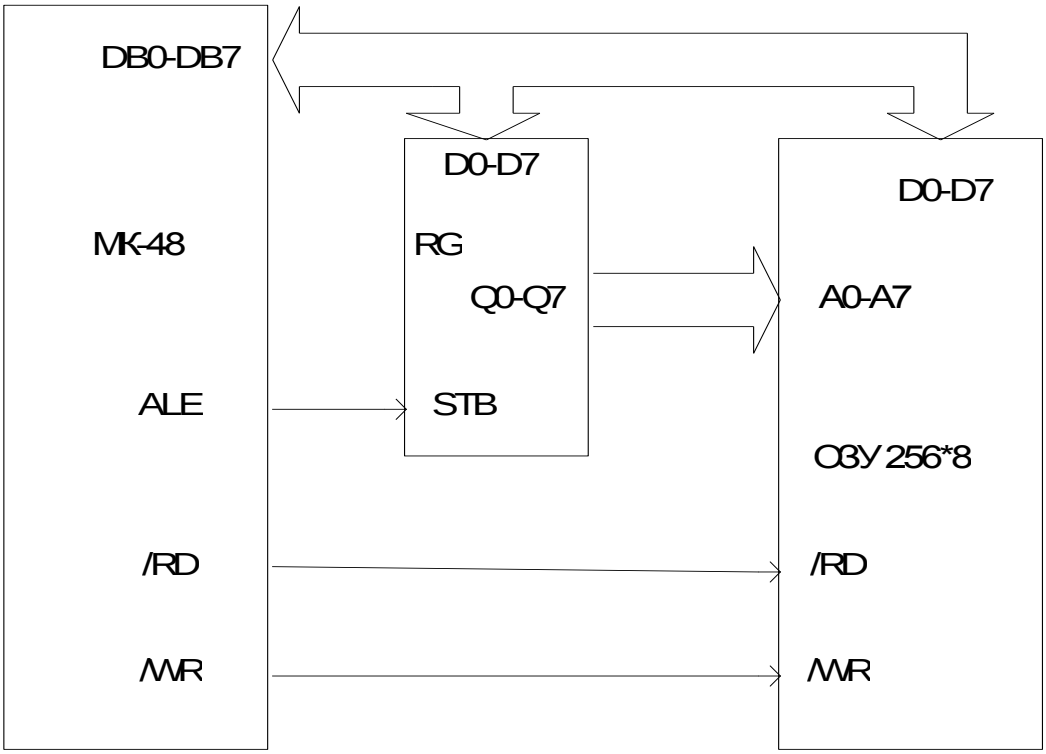


Рис. 7.7. Структурная схема подключения внешней памяти данных, объемом до 256 байт к МК48

Для работы с внешней памятью данных в микроЭВМ предусмотрены два управляющих сигнала: **/RD** – чтение внешней памяти данных, **/WR** – запись во внешнюю память данных.

Несколько замечаний по схеме. Так как шина DB используется в режиме временного мультиплексирования, то есть и для выдачи адреса ячейки памяти, и для выдачи при записи или приема при чтении данных, то ясна необходимость применения регистра, в котором по спаду сигнала ALE фиксируется адрес ячейки памяти, к которой происходит обращение. ОЗУ в режиме чтения выдает данные на шину DB из ячейки памяти, адрес которой оно имеет на адресных входах A0 – A7 по сигналу /RD. В режиме записи ОЗУ принимает данные с шины DB и записывает их в адресованную ячейку памяти по сигналу /WR. При отсутствии сигналов /RD и /WR выводы Q0 – Q7 ОЗУ находятся в третьем состоянии.

Временные диаграммы циклов чтения и записи внешней памяти данных.

Для работы с внешней памятью данных используются команды:

MOVX A, @R0

MOVX A, @R1 – две команды чтения внешней памяти данных.

MOVX @R0, A

MOVX @R1, A – две команды записи во внешнюю память данных.

Как видно, во всех командах участвует аккумулятор, а также регистры R0 или R1. Через регистры R0 или R1 реализована косвенная адресация внешней памяти данных, а именно в этих регистрах содержится адрес ячейки памяти.

Фрагмент программы, который производит запись числа 56 в ячейку внешней памяти данных, имеющую адрес 99.

MOV A, #56
MOV R0, #99
MOVX @R0, A

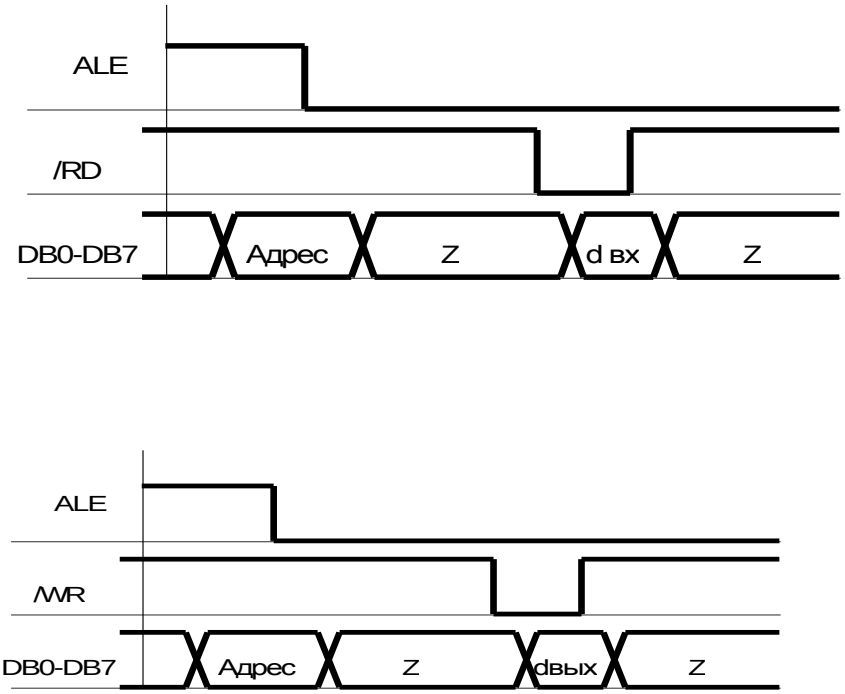


Рис. 7.8. Временные диаграммы циклов чтения
и записи внешней памяти данных

7.6. Таймер – счетчик

Таймер-счетчик – это устройство, заложенное в структуру МК48, позволяющее под управлением программы осуществлять временные выдержки в микропроцессорной системе и подсчитывать количество импульсов, если это необходимо, поступающих на вход Т1 микроЭВМ. Существует принцип организации временных выдержек, использующий то, что известно время выполнения каждой команды. Например, зная, что команда NOP выполняется за один машинный цикл, то есть 2,5 микросекунды, можно запрограммировать выдержку времени в 10 микросекунд, включив в программу 4 команды NOP. Более длинные временные интервалы можно реализовывать, используя команду цикла типа DJNZ. Например, следующий программный фрагмент реализует программную выдержку в 380 микросекунд.

```
        MOV R5, #50; 2 цикл=5 мкс
M1:     NOP          ; 1 цикл = 2, 5 мкс
        DJNZ R5, M1; 2 цикла=5 мкс
```

Как получились эти 380 микросекунд? Команда DJNZ уменьшает значение регистра R5 на единицу и проверяет, не равно ли новое значение нулю. Если не равно, то программа переходит на метку M1. Так как начальное значение R5 равно 50, то две последние команды будут выполняться 50 раз. Это даст программную выдержку в 375 микросекунд. Первая команда выполняется за 5 микросекунд. Суммарное время равно 380 микросекунд.

Такой метод организации временных выдержек нерационален, так как в течение всего времени процессор работает только на эту выдержку и больше ничего делать не может. Возникает вопрос: а нельзя ли освободить процессор от этой рутинной работы, возложив функции таймирования на некоторое другое устройство?

Таким устройством в микроЭВМ является таймер-счетчик. При организации программных выдержек роль процессора заключается теперь в загрузке таймера-счетчика некоторым числом, величина которого определяет величину программной выдержки времени, и запуску таймера на счет. Далее процессор освобождается, а таймер-счетчик прерывает работу процессора, когда закончит временную выдержку.

Разрядность таймера-счетчика равна 8, то есть максимальное число, которое можно загрузить в таймер-счетчик, равно 255. Команды загрузки и чтения таймера-счетчика **MOV T, A** и **MOV A, T.**

После загрузки таймера-счетчика каким-либо числом можно выполнить команду запуска таймера-счетчика на счет. Это команда **STRT T**. Счетчик работает на суммирование и каждые 80 микросекунд увеличивает свое значение на единицу. При переполнении, то есть когда его значение достигнет 255, таймер-счетчик сбрасывается, то есть обнуляется и продолжает счет с нуля. Этот момент фиксируется в триггере флага и триггере переполнения таймера-счетчика. Триггер флага можно опросить в любом месте программы и узнать, достиг ли таймер-счетчик переполнения. Это достигается выполнением команды **JTF ad8.**

Покажем на примере, как можно организовать выдержку времени с использованием таймера-счетчика методом опроса триггера флага. Пусть требуется сформировать импульс положительной полярности длительностью 800 микросекунд на выводе P10 порта P1.

Так как дискретность счета таймера-счетчика равна 80 микросекунд и счетчик работает на суммирование, то число, загружаемое в счетчик, определяется как 256 – 100=156, то есть к числу 156 нужно добавить 100 дискрет по 80 микросекунд, чтобы счетчик обнулился.

```
MOV A, #156
MOV T, A
ORL P1, #00000001B    ; Здесь присутствует погрешность в длительности
сформированного импульса.
STRT T                ;
M0: JTF M1
JMP M0
```

M1: ANL P1, #11111110B
STOP TCNT.

Максимальная выдержка времени получается при загрузке таймера-счетчика нулем и равна 256*80 микросекунд=20480 микросекунд.

Другой прием формирования импульса определенной длительности связан с использованием триггера переполнения. Когда возникают переполнения таймера-счетчика, последний формирует прерывание процессору, которое называется прерыванием от таймера-счетчика. Прерывания могут быть разрешены программно или запрещены. Команда **EN TCNTI** разрешает в дальнейшем прерывания, а команда **DIS TCNTI** запрещает. Если прерывания запрещены, то процессор никак не прореагирует на переполнение таймера-счетчика. Если прерывания от таймера-счетчика разрешены, то с приходом прерывания процессор микроЭВМ, закончив выполнения текущей команды, перейдет на выполнение команды, расположенной в памяти программ по адресу 0007H. Посмотрите карту распределения памяти программ. Адрес 0007H зарезервирован под вектор прерывания от таймера-счетчика. Далее приведена программа, которая использует прерывания от таймера-счетчика для формирования импульса в 800 микросекунд.

```
ORG 0
JMP START
ORG 7
ANL P1, #11111110B
STOP TCNT
RETR
START:
MOV A, #156
MOV T, A
EN TCNTI
ORL P1, #00000001
STRT T
```

Процессор запустил таймер-счетчик и теперь может бросить его и выполнять другую задачу. Больше ему не требуется возвращаться к таймеру-счетчику. Через 800 микросекунд установится триггер переполнения, сформирует прерывание от таймера-счетчика, процессор, что бы он ни делал в этот момент, вынужден будет прерваться и выполнить подпрограмму, начинающуюся с адреса 0007H и заканчивающуюся командой **RETR** – возврат из подпрограммы обработки прерывания.

Все, о чем говорилось раньше, в основном касалось режима работы таймера-счетчика в качестве таймера. Другой режим работы таймера-счетчика – это режим счетчика событий. По команде **STRT CNT** можно начать счет событий. Событием называется переход сигнала на внешнем выводе T1 от высокого уровня к низкому. Подсчет событий прекращается по команде **STOP TCNT**. При этом минимально допустимый период следования сигналов на входе T1 7.5 микросекунды.

Необходимо отметить, что нельзя одновременно использовать таймер-счетчик и в режиме таймера, и в режиме счетчика. Напоследок структурная схема включения таймера-счетчика.

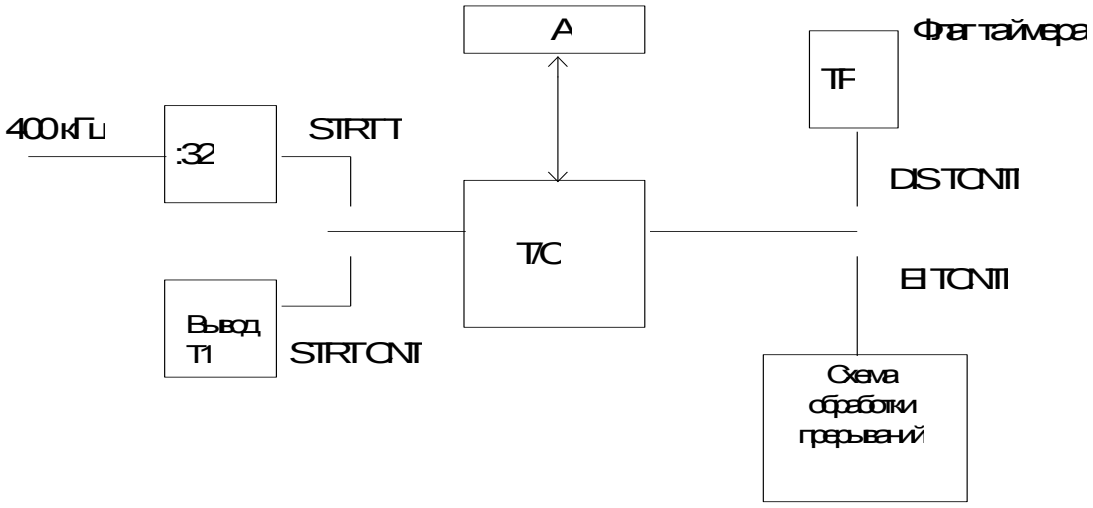


Рис. 7.9. Структурная схема включения таймера-счетчика

7.7. Система прерываний

В МК48 реализованы два вида прерываний:

- прерывание от внешнего источника;
- прерывание от внутреннего таймера-счетчика.

Прерывания могут быть избирательно разрешены или запрещены по командам EN и DIS соответственно.

EN I – команда разрешения внешнего прерывания по входу /INT. После выполнения этой команды низкий уровень на входе /INT воспринимается процессором как запрос на прерывание. В этом случае процессор заканчивает выполнение текущей команды и переходит на выполнение команды, находящейся по адресу 0003H. Именно с этого адреса должна начинаться подпрограмма обслуживания внешнего прерывания.

DIS I – команда запрещения внешнего прерывания. После выполнения этой команды любые изменения на входе /INT не прерывают работу процессора. После сброса микроЭВМ все виды прерываний запрещены.

Прерывания от таймера-счетчика были рассмотрены ранее.

Внешнее прерывание обладает более высоким приоритетом, то есть, если внешнее прерывание и прерывание от таймера-счетчика возникают одновременно, внешнее прерывание обслуживается в первую очередь. После обслуживания внешнего прерывания будет обслужено прерывание от таймера-счетчика.

Если в системе требуется использовать два внешних прерывания и таймер-счетчик свободен, то его можно использовать как источник второго внешнего прерывания. Для этого необходимо загрузить в таймер-счетчик число 255 и

запустить счетчик внешних событий командой **STRT CNT**. Тогда перепад сигнала из 1 в 0 на входе T1 будет восприниматься процессором как внешнее прерывание с выполнением программы с адреса 0007H.

Система прерываний одноуровневая. Это означает, что обслуживание вновь поступающих прерываний откладывается до конца обработки текущего. Обработка очередного прерывания может начаться только после завершения второго машинного цикла команды **RETR**. Сигнал INT низкого уровня должен быть снят до исполнения команды RETR, в противном случае процессор начнет повторное обслуживание данного запроса. В МК48 есть одна особенность, которую необходимо учитывать при организации прерываний.

При обслуживании прерываний старший разряд счетчика команд, независимо от состояния триггера DBF, аппаратно устанавливается в нуль всякий раз при переходе на подпрограмму обслуживания прерываний. Поэтому все программы обработки прерываний и все подпрограммы, вызываемые ими, должны располагаться в пределах нулевого банка памяти программ.

Структура программы с использованием прерываний:

```
ORG 0
JMP START
ORG 3
JMP INT
ORG 7
TEAM: ……..; подпрограмма обработки прерывания от T/C
      …….
      RETR
INT: ……..; подпрограмма обработки внешнего прерывания
      ……..
      RETR
START: …….; основная программа, начало
      …….
      …….; конец.
```

Имена меток взяты произвольно и могут быть другими.

7.8. Устройство управления и синхронизации

Устройство управления и синхронизации предназначено для выработки сигналов, обеспечивающих управление выполнением команд. Оно реализовано на кристалле микроЭВМ за исключением источника опорной частоты, в качестве которого используется кварцевый резонатор, LC – цепь или внешний источник синхроимпульсов. Варианты подключения источника опорной частоты.

Опорная частота f_{BQ1}(6 МГц) делится на 3 для получения внутренней частоты синхронизации f_{CLK} = 2 МГц. Сигналы CLK могут быть выведены на внешний вывод T0 по команде **ENTO CLK**.

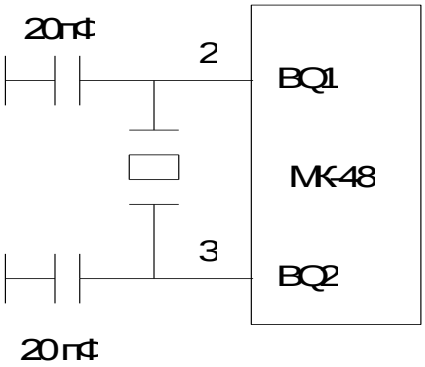


Схема включения кварцевого резонатора на 6 или 11 МГц. Этот вариант используется тогда, когда требуется получать временные соотношения с высокой точностью.

Рис. 7.10. Схема включения кварцевого резонатора на 6 или 11 МГц

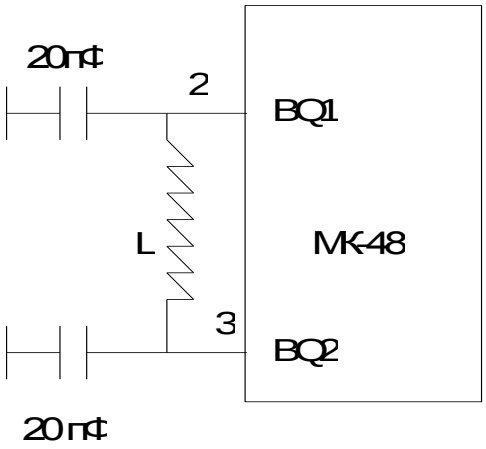


Схема с использованием LC - цепи. Более дешевая и менее качественная.

Рис. 7.11. Схема с использованием LC – цепи

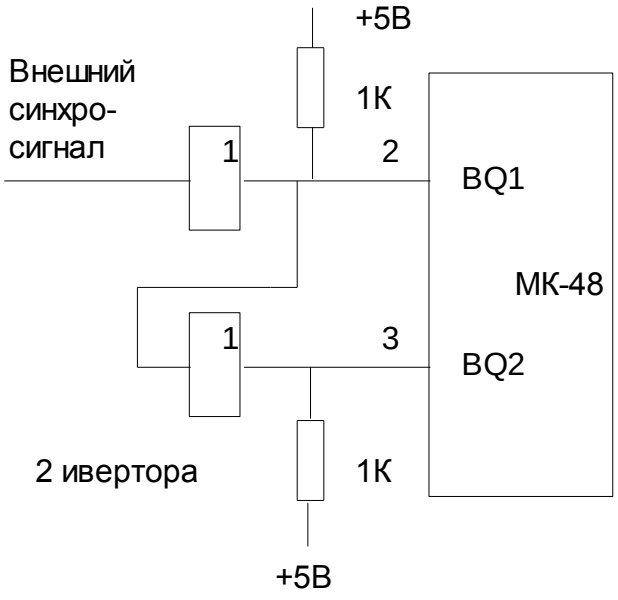


Рис. 7.12. Схема с внешним источником

Частота CLK делится на 5 в счетчике циклов для получения частоты, определяющей машинный цикл. $F_{мц}$ = 0, 4 МГц. Период машинного цикла = 2, 5 мкс. Эту же частоту имеет сигнал ALE.

Общий сброс микроЭВМ по включению питания реализуется подключением конденсатора к выводу /SR.

После сброса:

1. Обнуляются счетчик команд и указатель стека.
2. Порт P0 устанавливается в третье состояние, а порты P1,P2 в режим ввода.
3. Выбирается банк регистров RB0 и банк памяти программ MB0.
4. Запрещаются все прерывания.
5. Останавливается таймер – счетчик и блокируется выдача сигналов на вывод T0.
6. Сбрасываются флаги таймера – счетчики и флаги пользователя F0,F1.

7.9. Система команд

Система команд включает 96 команд, 68 из них однобайтные. В двухбайтных командах первый байт несет информацию о коде операции, второй байт представляет собой непосредственные данные или младшие разряды адреса следующей команды. Большинство команд (53) выполняется за один машинный цикл; 43 команды выполняются за два машинных цикла.

Команды по смыслу разбиты на группы:

1. Команды пересылок.
2. Команды арифметики.
3. Команды логики.
4. Команды передачи управления.
5. Команды управления режимами.

7.9.1. Команды пересылок

Во всех командах пересылок присутствует два операнда: источник и приемник байта пересылаемых данных. Все группы команд будем изображать по возможности в виде схем. В названии команды в скобках указаны количество байтов в команде и время выполнения в циклах.

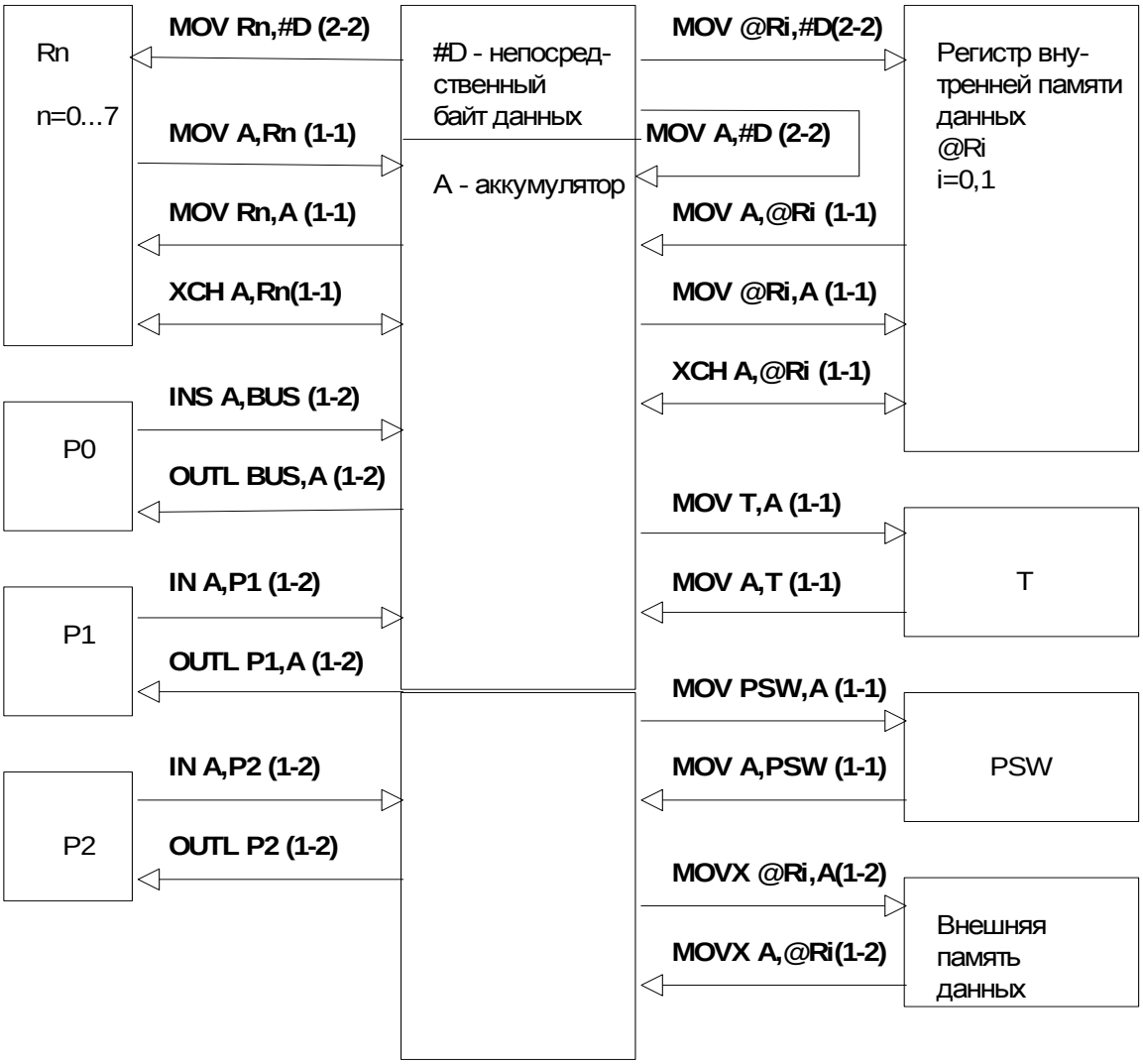

Rn
n=0...7
MOV Rn,#D (2-2)
MOV A,Rn (1-1)
MOV Rn,A (1-1)
XCH A,Rn(1-1)
#D - непосред-
ственный
байт данных
A - аккумулятор
MOV @Ri,#D(2-2)
MOV A,#D (2-2)
MOV A,@Ri (1-1)
MOV @Ri,A (1-1)
XCH A,@Ri (1-1)
Регистр вну-
тренней памяти
данных
@Ri
i=0,1
P0
INS A,BUS (1-2)
OUTL BUS,A (1-2)
P1
IN A,P1 (1-2)
OUTL P1,A (1-2)
P2
IN A,P2 (1-2)
OUTL P2 (1-2)
MOV T,A (1-1)
MOV A,T (1-1)
T
MOV PSW,A (1-1)
MOV A,PSW (1-1)
PSW
MOVX @Ri,A(1-2)
MOVX A,@Ri(1-2)
Внешняя
память
данных

Рис. 7.13

7.9.2. Команды арифметики

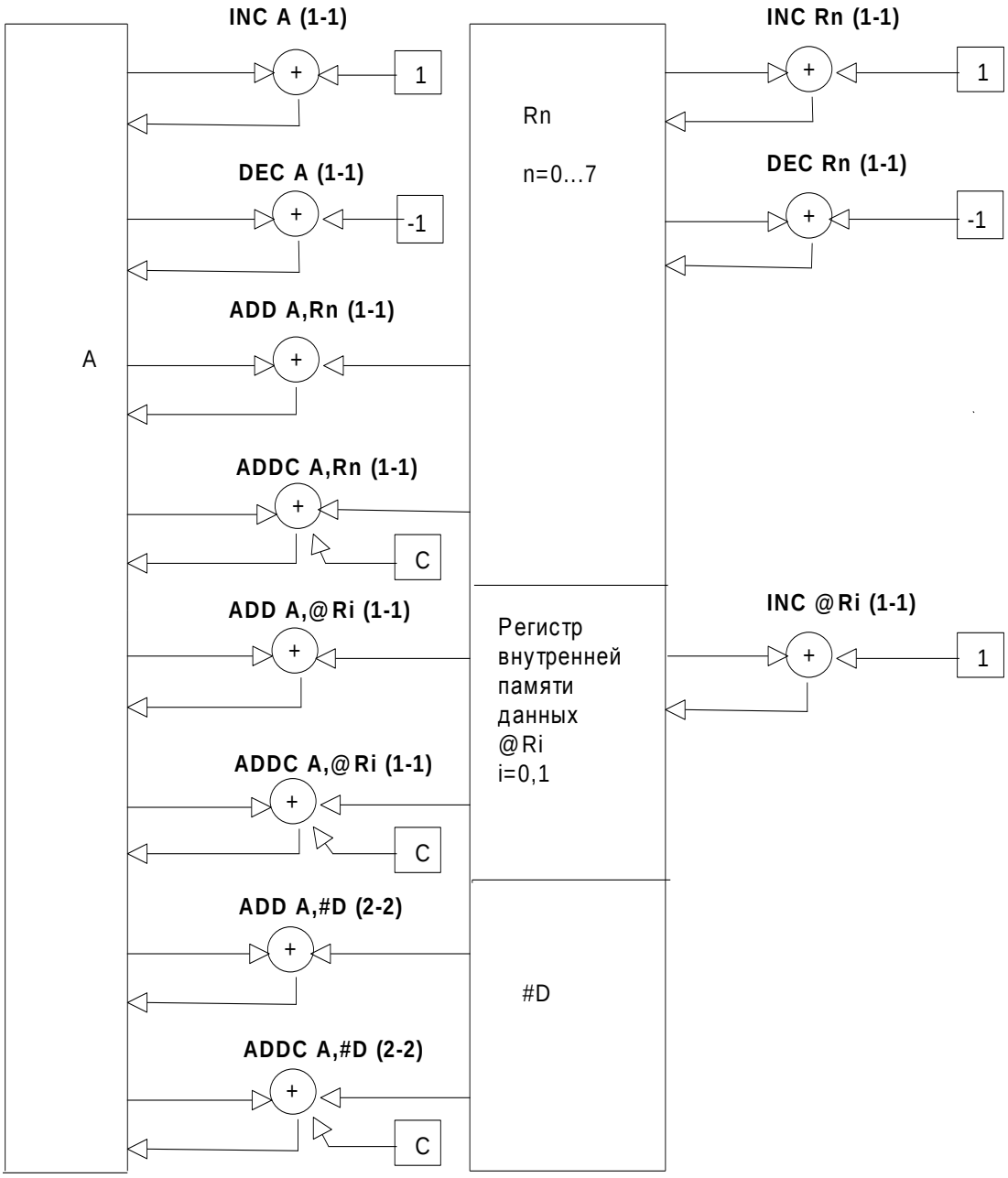


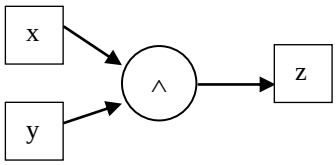
Рис. 7.14

Сюда же входит команда десятичной коррекции **DA A**.

7.9.3. Команды логики

Напомним, что логические операции выполняются над двумя 8-разрядными операндами поразрядно в соответствии с таблицами истинности для каждой логической операции.

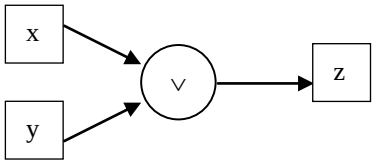
Логическое умножение. Обозначим значком $\wedge$.



X	y	Z
0	0	0
0	1	0
1	0	0
1	1	1

Рис. 7.15

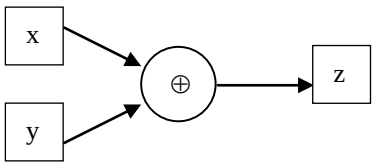
Логическое сложение. Обозначим значком $\vee$.



X	y	Z
0	0	0
0	1	1
1	0	1
1	1	1

Рис. 7.16

Исключающее ИЛИ. Обозначим значком $\oplus$



X	y	Z
0	0	0
0	1	1
1	0	1
1	1	0

Рис. 7.17

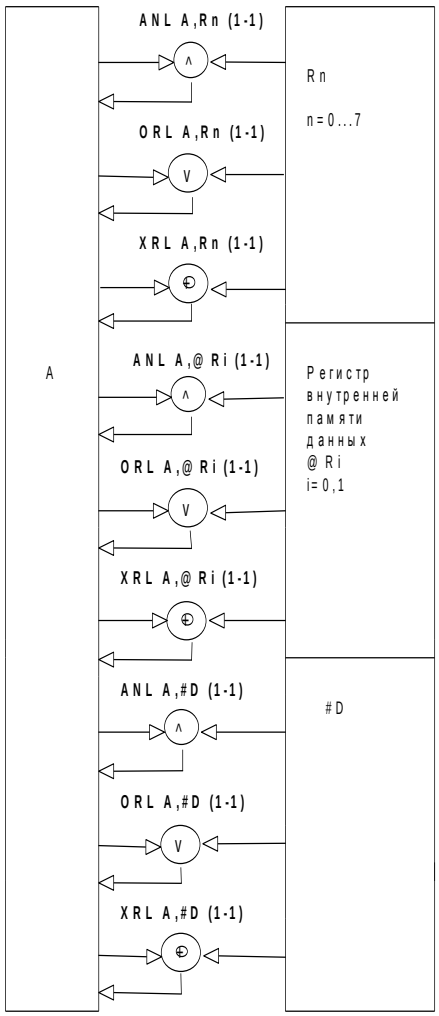
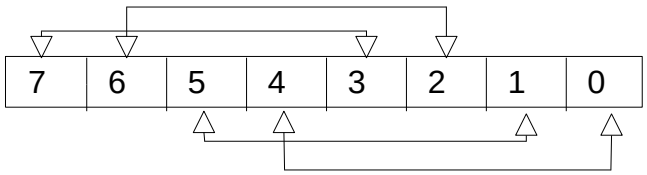


Рис. 7.18

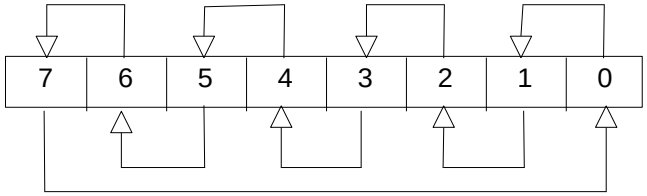
CLR A (1-1) ; (A)⇐0
CPL A (1-1) ; (A)⇐(/A)
SWAP (1-1) ; (A0…A3)⇔(A4…A7)

Все последующие логические операции выполняются над содержимым аккумулятора.

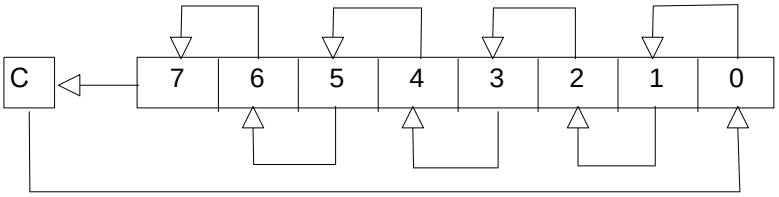
Схема команды **SWAP**



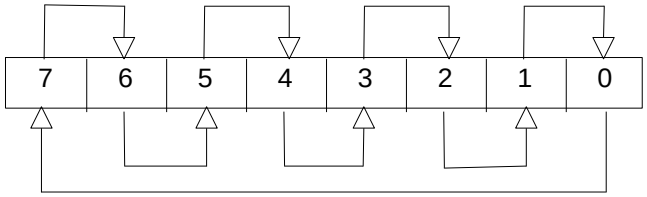
RL A (1-1) ; сдвиг влево



RLC A (1-1) ; сдвиг влево через бит переноса C



RR A (1-1) ; сдвиг вправо



RRC A (1-1) ; сдвиг вправо через бит переноса C

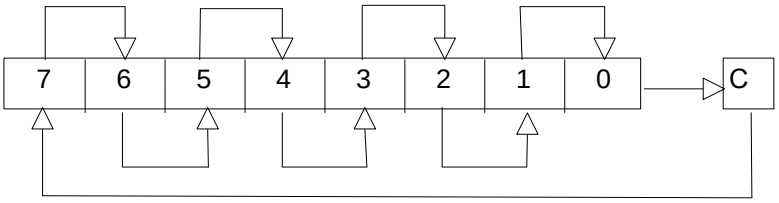


Рис. 7.19. Схемы команды **SWAP**

Далее следуют команды типа “чтение – модификация – запись”. При выполнении этих команд содержимое порта считывается, модифицируется и записывается в порт.

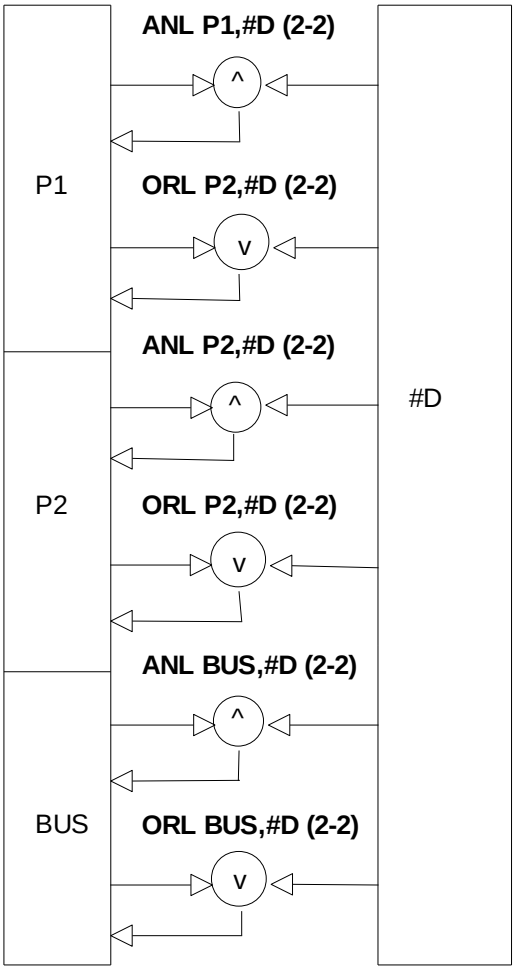


Рис. 7.20

Следующая группа команд модифицирует бит переноса C, флаги пользователя F0 и F1.

CLR C (1-1) ; (C) ⇐0
CPL C (1-1) ; (C) ⇐(/C)
CLR F0 (1-1) ; (F0) ⇐ 0
CPL F0 (1-1) ; (F0) ⇐ (/F0)
CLR F1 (1-1) ; (F1) ⇐ 0
CPL F1 (1-1) ; (F1) ⇐ (/F1)

7.9.4. Команды передачи управления

JMP ad11 (2-2); Безусловный переход по адресу (метке), указанному в команде в пределах выбранного банка памяти программ.

Примеры структур.

JMP LOOP23
….
….
LOOP23: ….
….

Или

….
….
LOOP7: …
….
JMP LOOP7
….

JC ad8 (2-2); Переход на метку в пределах страницы, если бит переноса С установлен. Иначе выполняется следующая команда. Этот принцип работает и в следующих командах условного перехода. Страница – 256 байт. Переход возможен на 127 байт вверх или на 128 байт вниз.

JNC ad8 (2-2); Переход на метку в пределах страницы, если бит С не установлен.

JZ ad8 (2-2)

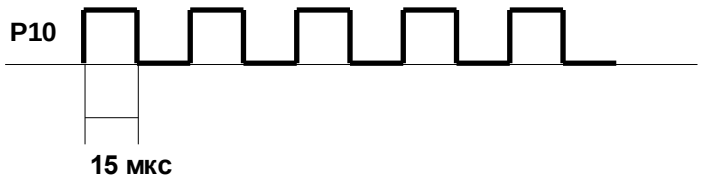
JNZ ad8 (2-2). Команды условного перехода по признаку нуля в аккумуляторе.

DJNZ Rn, ad8 (2-2); Команда цикла предназначена для организации программных циклов. Rn – счетчик повторений цикла. Ad8 – метка начала цикла. Команда выполняется в два приема:

1. Rn $\Leftarrow$ Rn-1
2. Если Rn $\neq$ 0, то переход на ad8, иначе – выход из цикла и выполнение следующей команды.

Пример формирования импульсной последовательности на выходе P10 порта P1.

MOV R1, #10
MOV A, #0
M0: XRL A, #1
OUTL P1, A
DJNZ R1, M0



Попутное замечание: Команда **XRL A,#1** может использоваться для инверсии конкретного бита, в нашем примере – бита 0.

JT0 ad8 (2-2)

JNT0 ad8 (2-2). Условный переход по состоянию сигнала на выводе T0.

JT1 ad8 (2-2)

JNT1 ad8 (2-2). Условный переход по состоянию сигнала на выводе T1.

JF0 ad8 (2-2)

JF1 ad8 (2-2). Условный переход, если установлен флаг пользователя F0 или F1.

JTF ad8 (2-2) Условный переход по переполнению таймер-счетчика.

JNI ad8 (2-2) Условный переход, если на входе INT низкий уровень.

JBb ad8 (2-2) Переход по метке, если бит b аккумулятора равен 1.

JB2 M1 (2-2) Переход на метку M1, если второй бит аккумулятора равен 1.

Команды для работы с подпрограммами.

CALL ad11 (2-2) Переход на подпрограмму, начало которой расположено по адресу ad11.

RET (1-2) Возврат из подпрограммы. Из стека восстанавливается только значение счетчика команд.

RETR (1-2) Возврат из подпрограммы обработки прерывания. Из стека восстанавливается значение счетчика команд регистра PSW.

7.9.5. Команды управления режимами

NOP (1-1) Холостая команда. Реализует временную выдержку в 2,5 микросекунды.

STRT T (1-1) Запуск таймера-счетчика в режиме таймера.

STRT CNT (1-1). Запуск таймера-счетчика в режиме счетчика событий.

STOP TCNT (1-1) Останов таймера-счетчика.

EN TCNTI (1-1) Разрешить прерывания от таймера-счетчика.

DIS TCNTI (1-1) Запретить прерывания от таймера-счетчика.

EN I (1-1) Разрешить внешние прерывания.

DIS I (1-1) Запретить внешние прерывания.

SEL RB0 (1-1)

SEL RB1 (1-1) Выбор банка рабочих регистров.

SEL MB0 (1-1)

SEL MB1 (1-1) Выбор банка памяти программ.

ENTO CLK (1-1) Разрешить выдачу синхросигнала на вывод T0.

Система команд МК48 имеет кроме вышеприведенных еще несколько команд, которые будут рассмотрены в примерах применения МК48.

7.10. Примеры практического применения МК48

7.10.1. Совместная работа с устройствами аналогового ввода-вывода

Рассмотрим подсистему аналогового ввода-вывода на основе микроЭВМ, имеющих внутреннюю память программ. Рассматриваемая подсистема имеет 8 аналоговых входов и один аналоговый выход.

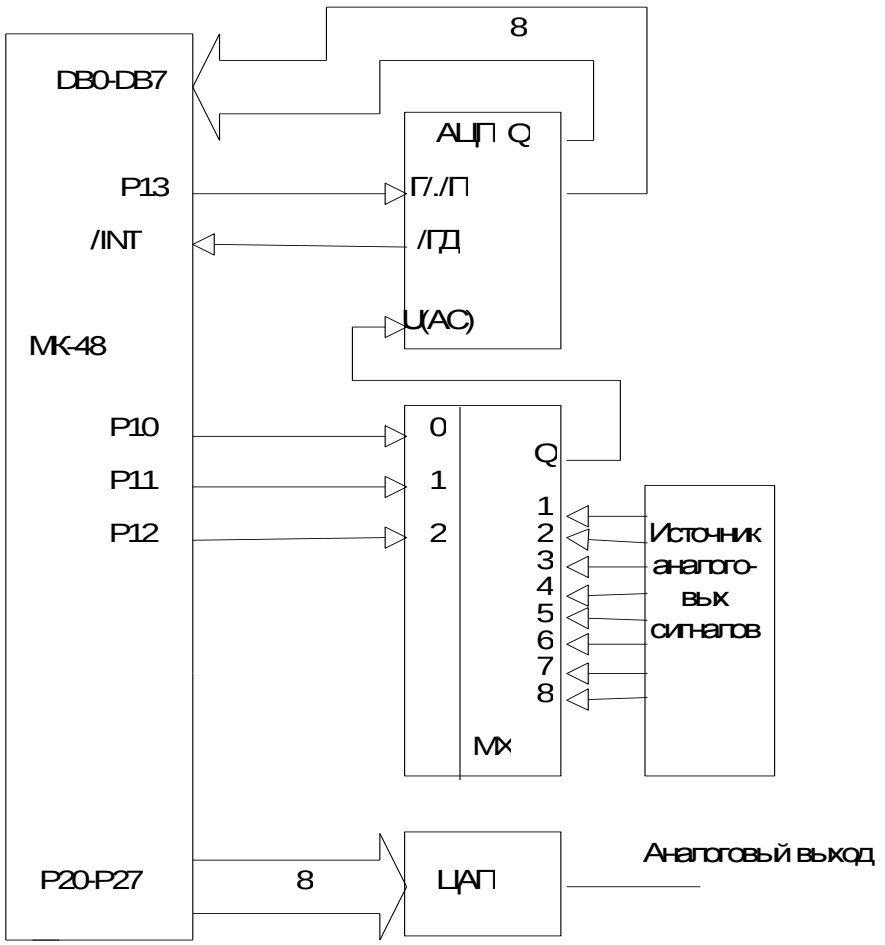


Рис. 7.21

Функциональная схема приведена для аналогового коммутатора КР590КН6 и АЦП К1113ПВ1.

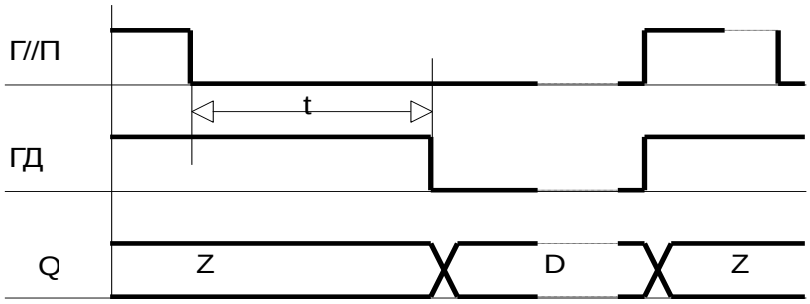
Здесь МХ – аналоговый мультиплексор или 8-канальный аналоговый коммутатор, подключает один из 8 аналоговых входов к аналоговому входу U(AC) АЦП. Номер подключаемого входа определяется кодом на входах 0,1,2 коммутатора и выбирается программно с помощью разрядов P10…P12 микроЭВМ в соответствии с таблицей 7.6.

Таблица 7.6

2	1	0	Номер канала
0	0	0	1
0	0	1	2
0	1	0	3
0	1	1	4
1	0	0	5
1	0	1	6
1	1	0	7
1	1	1	8

Сигналом запуска по входу Г//П (гашение и преобразование) запускается цикл преобразования АЦП. По окончании цикла на выходах Q АЦП появляется код, соответствующий напряжению на входе U(АС) АЦП, а на выходе ГД (готовность данных) АЦП устанавливается низкий логический уровень, означающий готовность данных на выходе АЦП и вызывающий прерывание микроЭВМ по входу /INT.

В подпрограмме обработки этого прерывания микроЭВМ вводит код с выходов АЦП через порт P0. Временная диаграмма, поясняющая принцип работы микроЭВМ и АЦП.



t - Время преобразования АЦП (30 микросекунд для 1113ПВ1)
D - Результат преобразования.

Рис. 7.22

Программа, реализующая непрерывный циклический процесс ввода аналогового сигнала с канала 1.

```
.ORG 0
EN I
JMP START
.ORG 3
INS A, BUS
MOV R0, A
CLR F1
RETR
START: CLR F1
CPL F1
MOV A, #08H
OUTL P1, A
ANL P1,#11110111B
M1:    JF1 M1
JMP START
```

Рассмотрим вариант, когда микроЭВМ не имеет внутренней памяти программ.

Прежде всего приведем часть общей схемы, обеспечивающей подключение внешней памяти программ.

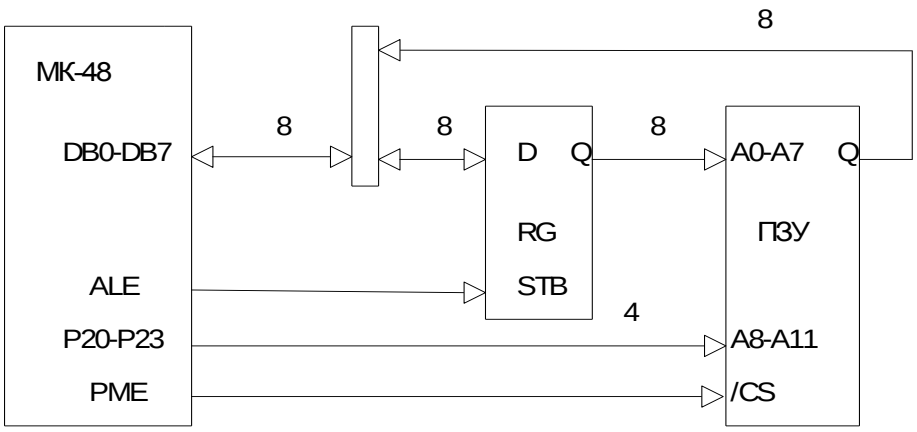


Рис. 7.23

Работа схемы была рассмотрена ранее. Из внешних ресурсов в нашем распоряжении остался порт P1 и половина порта P2 с выводами P24-P27. Используем их для подключения многоканального АЦП.

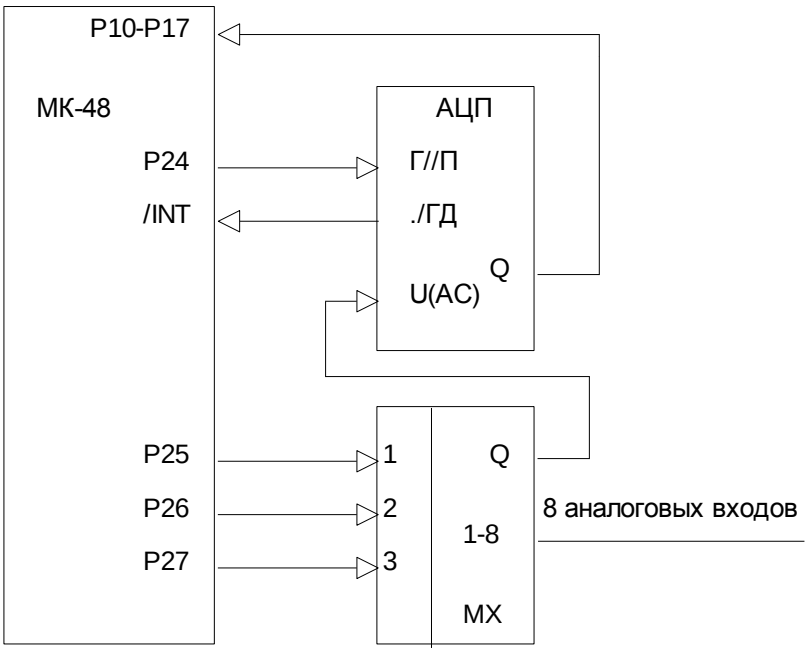


Рис. 7.24

Работа этой части схемы тоже рассматривалась ранее. Внешних ресурсов (портов) микроЭВМ не осталось. Подключение ЦАП вызывает проблему.

Используем возможность работы микроЭВМ с внешней памятью данных. Можно представить ЦАП как ячейку внешней памяти данных, в которую для формирования аналогового сигнала нужно уметь записывать байт данных так, как это делается при записи данных во внешнюю память. Известно, что по команде **MOVX @Ri , A** происходит пересылка содержимого аккумулятора во внешнюю память данных, адрес которой находится в регистре Ri. При этом на шину DB сначала выводится адрес ячейки внешней памяти данных и

фиксируется по сигналу ALE во внешнем регистре, выходы которого должны быть подключены к адресным входам ОЗУ. Этот адрес необходимо использовать, если количество подключаемых устройств больше одного. Но в нашем случае используется один ЦАП, его адресации не требуется, поэтому внешний регистр адреса не используется. Далее по алгоритму выполнения команды **MOVX @Ri, A** на шину DB помещаются данные и формируется сигнал /WR – запись. Необходимо зафиксировать эти данные в каком-то внешнем регистре по сигналу /WR и с выхода этого регистра подать на вход ЦАП. Тогда схема, реализующая подключение ЦАП, будет иметь вид.

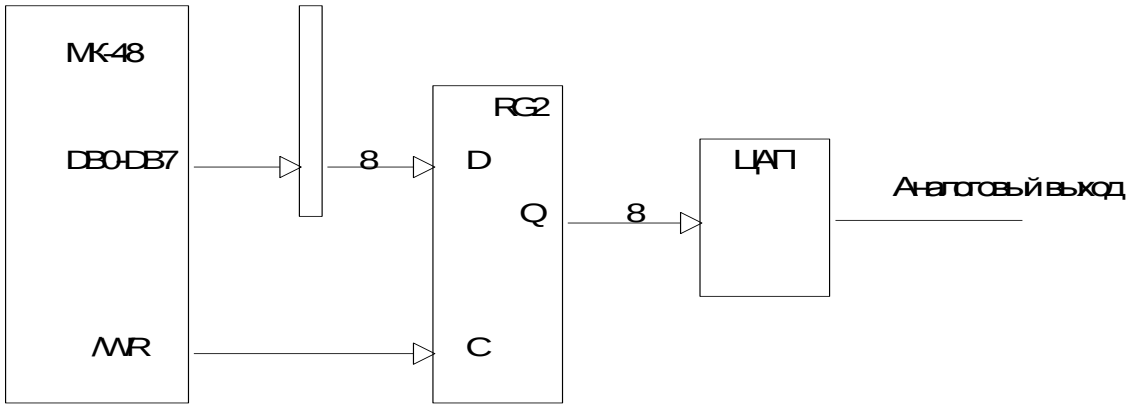


Рис. 7.25

Таким образом, к внутренней шине системы DB0-DB7 подключено несколько устройств: ПЗУ, RG1, RG2. В других системах шина DB0-DB7 может быть нагружена большим количеством устройств. Ограничение – в нагрузочной способности шины.

Необходимо понять, что прохождение данных по шине происходит последовательно. Выполнение команды **MOVX @Ri, A** разбивается на такие шаги:

1. На шине DB устанавливается адрес команды.
2. С шины DB снимается код команды.
3. На шине DB устанавливается адрес внешней ячейки памяти, куда микроЭВМ готовится записать данные.
4. На шину DB выдаются записываемые данные.

Команда **MOVX A, @Ri** выполняется аналогично за 4 шага, только вместо сигнала /WR – запись формируется сигнал /RD – чтение, и в последнем цикле производится чтение данных с шины DB.

7.10.2. Использование шины DB для расширения микропроцессорной системы

В предыдущем разделе на примере использования подсистемы аналогового ввода – вывода было рассмотрен вариант расширения системы с использованием универсальной шины DB. Рассмотрим общие принципы использования шины DB для расширения системы. Сам по себе МК48 обладает достаточными ресурсами для подключения различных устройств, но в некоторых особых случаях недостаточно и этих ресурсов. В этом случае используется возможность работы МК48 с внешней памятью данных. МК48 может обслуживать до 256 ячеек внешней памяти данных, то есть записывать в эти ячейки порции информации размером в байт или считывать из них байт информации. Процесс записи байта в ячейку с адресом А – это пересылка содержимого аккумулятора в эту ячейку. Этот процесс инициализируется по команде **MOV Ri,A**. По этой команде на шину DB вначале выдается адрес ячейки внешней памяти, который содержится в регистре Ri. Размер адресного слова – 1 байт, именно поэтому максимальное количество адресуемых ячеек равно 256. Это адресное слово должно быть зафиксировано (сохранено) во внешних схемах по спаду сигнала ALE, потому что далее на шину DB выдается байт данных, содержащихся в аккумуляторе, который сопровождается сигналом /WR . По команде **MOV A,Ri** фиксация адреса ячейки происходит аналогично команде **MOV Ri,A** , далее – же шина DB готова принять данные (настроена на прием), который реализуется сигналом /RD.

Исходя из этого, можно сделать вывод о том, что МК48 способна обмениваться информацией с 256 устройствами ввода – вывода с байтовой организацией, если рассматривать внешнее устройство как аналог ячейки ОЗУ.

7.10.3. Генерация сигналов различной формы

7.10.3.1. Импульсный генератор

Импульсный генератор, построенный на микроЭВМ, должен генерировать импульсную последовательность со скважностью 2, с максимальным периодом 40,96 мс, минимальным периодом 160 мкс. Управление внешнее от кнопочного пульта.

Подход к разработке схемы.

Используем микроЭВМ с внутренней памятью данных. Для связи с пультом управления в нашем распоряжении все порты микроЭВМ и входы тестирования Т0 и Т1. Для вывода импульсной последовательности можно использовать любую линию любого порта микроЭВМ. Пусть это будет вывод Р20. Для включения/выключения генератора имеется кнопка S0 с фиксацией.

Если она нажата, на выходе Р20 имеется импульсная последовательность, если отжата – на выходе имеется логический 0. Для задания периода

генерируемой импульсной последовательности имеются кнопки S1-S8, сигнал с которых нужно подать на один из портов микроЭВМ. Пусть это будет порт P1.

Тогда схема будет выглядеть следующим образом.

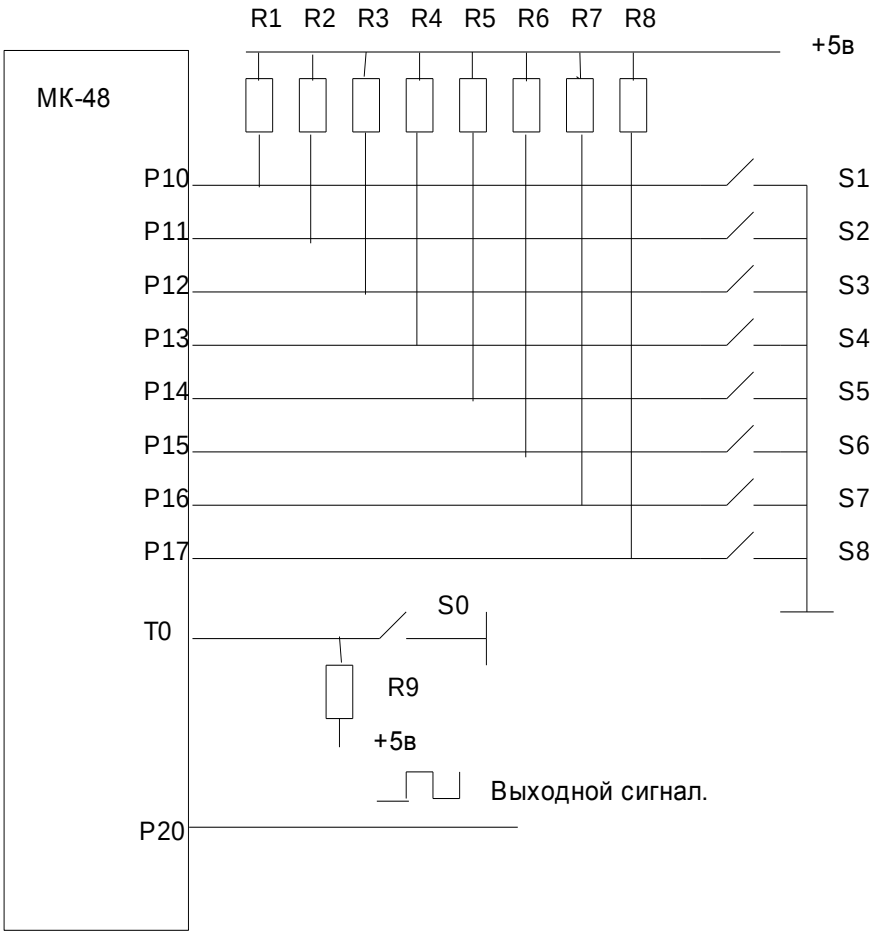


Рис. 7.26

Программа будет иметь вид.

```
.ORG 0
JMP START
.ORG 7
IN A, P2
XRL A, #1
OUTL P2, A
IN A, P1
MOV T, A
RETR
START: EN TCNTI
MOV A, #0FFH
OUTL P1, A
M3:    CLR A
OUTL P2, A
M0:  JT0, M0
```

IN A, P1
MOV T, A
STRT T
M1: JNT0 M1
STOP TCNT
JMP M3

7.10.3.2. Генератор синусоидальных сигналов

Для формирования сигнала такой сложной формы, как синусоида, требуется использование системы с аналоговым выводом. При этом вместо вычисления значений функции в реальном масштабе времени производится автономное вычисление этих значений и занесение их в память в виде таблицы. Одним из методов реализации синусоидальной функции может быть использование таймера событий для организации прерываний с фиксированной частотой, не более чем в 256 раз большей, чем требуемая выходная частота. В момент прерывания соответствующее значение синусоидальной функции может быть вычислено из ряда Маклорена:

$$\mathrm{Sin}\,(X) = X - X^3/3! + X^5/5! - X^7/7! \ldots((-1)^K \cdot X^{(2\cdot K+1)})/(2\cdot K+1)!$$

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

В системе команд микроЭВМ предусмотрена специальная команда **MOVP3 A, @A** для выбора данных из таблиц, если таблица помещена в последние 256 байт памяти программ или в третью страницу памяти программ. Для памяти программ размером 1 Кб страницы памяти программ имеют границы:

0 – 255 – 0 страница;
256 – 511 – 1 страница;
512 – 767 – 2 страница;
768 – 1023 – 3 страница.

Эта команда использует исходное содержимое аккумулятора для адресации ячейки третьей страницы памяти программ. Содержимое указанной ячейки считывается и помещается в аккумулятор. Если требуется таблица объемом меньше 256 байт, ее можно разместить на любой странице памяти программ и для нахождения данных в таблице использовать команду **MOVP A, @A**. Эта команда аналогична предыдущей, но она предполагает, что таблица содержится на текущей странице памяти программ.

Поскольку синусоида формируется цифровым способом, необходимо разбить четверть периода синусоиды на временные дискреты. Разобьем синусоиду на дискреты через один градус от 0 градусов до 90 градусов. Из таблицы синусов найдем значение функции Sin, соответствующее каждой дискрете:

Sin (0) = 0;

Sin (1) = 0, 0157;

Sin (2) = 0, 0314;

…

Sin (90) = 1.

Ответим на вопрос, в каком виде эти числа должны храниться в таблице памяти программ. ЦАП, на который будет подан код, соответствующий Sin (90), должен выдать на своем выходе максимальное значение по амплитуде, значит, код на 8-разрядном входе ЦАП должен быть равен 255. Составляя пропорцию Sin (90) – 255, а Sin (n) – x. Получим требуемое значение, которое округлим до целых. Например, для Sin (1) код ЦАП равен 4, для Sin (2) – 8 и так далее, для Sin (45) – 166.

Таким образом, мы имеем таблицу из 91 значения в диапазоне от 0 до 255, которое поместим во внутреннюю память программ, начиная с адреса 768 последовательно. Программа должна по каждому прерыванию от таймера выбирать очередное значение из этой таблицы и посылать его на ЦАП.

Таким образом, за 90 временных дискрет можно будет сформировать первую четверть периода синусоиды от 0 градусов до 90. Чтобы сформировать вторую четверть периода синусоиды, значения из таблицы должны выбираться в обратном порядке. Для формирования второй половины периода синусоиды нужно повторить предыдущую процедуру, сменив знак синусоиды. Это можно сделать, применив ЦАП с двухполярным питанием, имеющий вход выбора знака. Схема генератора с применением микроЭВМ с внутренней памятью программ приведена на рис. 7.27.

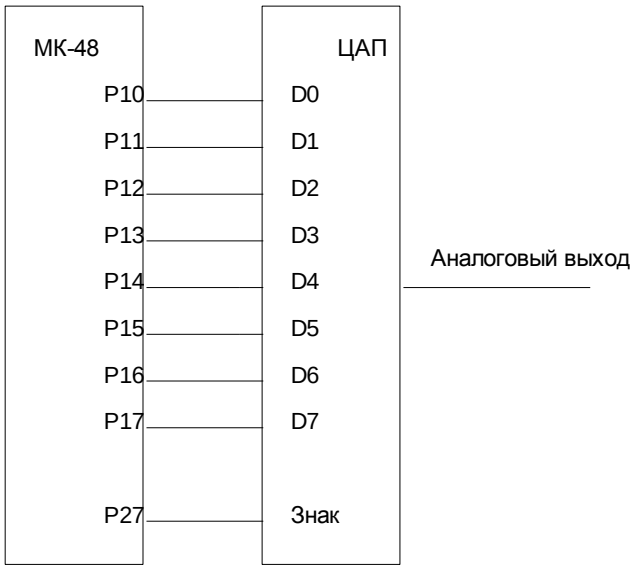


Рис. 7.27

Оговорим то, что для P27 = 0 имеем на выходе ЦАП положительный сигнал, для P27 = 1 – отрицательный сигнал. Поскольку предполагается использование внутреннего таймера, то период сформированного сигнала, который должен быть равен 360 дискретам времени, будет зависеть от величины этой дискреты в единицах времени. Как известно из ранее пройденного материала, таймер можно запрограммировать на время от 80 мкс до 20,48 мс. Тогда период сформированной синусоиды может быть в пределах от 80*360 = 28,8 мс до 20,48*360 = 7372,8 мс. Ниже приведена программа формирования сигнала синусоиды.

```
.ORG 0
JMP START
.ORG 7
JF0 M0
JF1 M1
MOV A, #0          ; формирование
OUTL P2, A         ; 1 четверти периода
MOV A, R2          ; F0=0
MOVP3 A, @A        ; F1=0
OUTL P1, A
INC R2
MOV A, R2
JNZ END
CLR F1
CPL F1
MOV R2, #254
JMP END
M1:  MOV A, R2       ; формирование
MOVP3 A, @A  ; 2 четверти периода
OUTL P1, A      ;  F0=0
DEC R2          ;  F1=1
MOV A, R2
JNZ END
CLR F0
CPL F0
CLR F1
JMP END
M0:  MOV A, #80H   ; формирование
OUTL P2, A      ; 3 четверти
JF1 M2          ;  F0=1
MOV A, R2       ;  F1=0
MOVP3 A, @A
OUTL P1, A
INC R2
MOV A, R2
```

```
JNZ END
CLR F1
CPL F1
MOV R2, #254
JMP END
M2: MOV A, R2       ; формирование
MOVP3 A, @A ;  4 четверти
OUTL P1, A      ;  F0=1
DEC R2          ;  F1=1
MOV A, R2
JNZ END
CLR F0
CLR F1
END:  RETR
START: EN TCNTI
MOV A, #0
OUTL P1, A
OUTL P2, A
MOV R2, #1
MOV A, #255
MOV T, A
STRT T
M3:  JMP M3
.ORG 768
DB 0
DB 4
DB 8
….
….
```

7.10.3.3. Измеритель временных интервалов

Рассмотрим практическую задачу измерения длительности импульса с применением таймера-счетчика. Пусть импульс приходит на вход Т0 микроЭВМ. Алгоритм заключается в опросе состояния этого входа до тех пор, пока он не станет равным 1. После этого запускается предварительно обнуленный таймер и продолжается опрос состояния входа Т0, пока оно не станет равным 0. После этого состояние таймера укажет на длительность импульса. Длительность импульса будет равна состоянию таймера, умноженному на 80 мкс. К сожалению, погрешность такого метода измерения составляет 80 мкс.

7.10.3.4. Измерение частоты

Принцип измерения частоты заключается в подсчете числа импульсов за определенный промежуток времени. В качестве этого промежутка времени удобнее взять значение 1 секунда, тогда подсчитанное число импульсов будет равно частоте в герцах.

Для формирования интервала времени 1 секунда используется таймер – счетчик. Удобно подать импульсы на вход внешних прерываний /INT, тогда окончание каждого импульса (переход из 1 в 0) будет вызывать прерывание МК48, по которому программно увеличивается счетчик импульсов. Счетчик импульсов реализуется на одном, а лучше двух регистрах, например, регистр R6 хранит младший байт числа, а регистр R7 – старший байт числа. По каждому внешнему прерыванию программа должна увеличивать значение регистра R6, проверять его переполнение и, если он переполнился (обнулился), увеличивать значение регистра R7.

Рассмотрим вопрос реализации времени выдержки в 1 секунду. При исходном состоянии таймера – счетчика, равном 0, он может обеспечить до своего обнуления интервал времени, равный 20,48 мс. Определим, сколько таких интервалов укладывается целиком в 1 секунде, используя формулу:

$$t_{1сек} = t_{20,48} \times n + \Delta t ,$$

где $t_{1\ сек}$ – интервал времени 1 секунда;

$t_{20,48}$ – интервал времени 20, 48 мс.;

n – число вложений интервалов 20, 48 мс. в интервале 1 сек.;

Δt – остаток.

По расчетам получилось: n = 48, Δt = 1696 мкс. Значит, таймер – счетчик должен 48 раз пройти полный (20,48 мс) цикл счета и останется еще 1696 мкс для реализации выдержки времени в 1 секунду. Так как присутствует остаток, то таймер – счетчик необходимо первоначально загрузить некоторым числом, чтобы этот остаток ликвидировать. Это число вычисляется по формуле:

$$n_{нач} = 256 - INT(\frac{\Delta t}{80}),$$

где INT(X) – целая часть числа X.

В результате расчета получилось, что $n_{нач}$=235. Таким образом, при загрузке первоначально в таймер – счетчик числа 235 до его первого обнуления пройдет 80 мкс * (266-235) = 1680 мкс. Имеем в виду, что остается еще 16 мкс, т.е. можно записать, что

$t_{1\ сек}$=20480 * n + 21 * 80 + 16 = 1000000 мкс.

Приведенная ниже программа реализует выдержку времени, равную 1 сек с применением таймера – счетчика.

```
.ORG 0
JMP START
.ORG 7
DJNZ R5,TEAM0   ; (2)
```

```
CLR F1              ; (1)
TEAM0:RETR      ; (2)
START:MOV A,#235
MOV T,A
MOV R5,#48
 CLR F1
EN TCNTI
STRT T
M1:    JF1 M1       ; (2)
STOP TCNT         ; (1)
.END
```

От команды **STRT T** до команды **STOP TCNT** пройдет время, равное 20480·48 + 21·80 + время на выполнение команд, отмеченных в комментариях цифрами. Цифры обозначают количество тактов, их сумма равна 8, а время выполнения равно 8·2,5=20 мкс. Таким образом, данная программа реализует временную выдержку, равную 20480·48 + 21·80 + 20 = 1000004 мкс, т.е. на 4 мкс больше необходимой. Очевидно, что такая погрешность допустима.

Импульсы, частота которых измеряется, поступают на вывод /INT. Подсчет этих импульсов ведется в подпрограмме обработки внешнего прерывания. Учитывая это, запишем программу полностью.

```
.ORG 0
JMP START
.ORG 3
JMP INT              ;(2)
.ORG 7
DJNZ R5,TEAM0   ;(2)
CLR F1               ;(1)
TEAM0:RETR       ;(2)
INT: MOV A,#1     ;(2)
ADD A,R6           ;(1)
MOV R6,A           ;(1)
MOV A,#0           ;(2)
ADDC A,R7          ;(1)
MOV R7,A           ;(1)
RETR               ;(2)
START:MOV R6,#0
MOV R7,#0
MOV A,#235
MOV T,A
MOV R5,#48
CLR F1
CPL F1
EN TCNTI
```

```
STRT T
EN I
M1:  JF1 M1
DIS I
STOP TCNT
JMP START
.END
```

Данная программа предъявляет ограничения на параметры входного импульса.

Ограничение на длительность импульса (см. рис. 7.28). Должно выполняться условие: $t_{имп} < t_{INT}$ означает, что длительность импульса должна быть меньше времени выполнения подпрограммы INT.

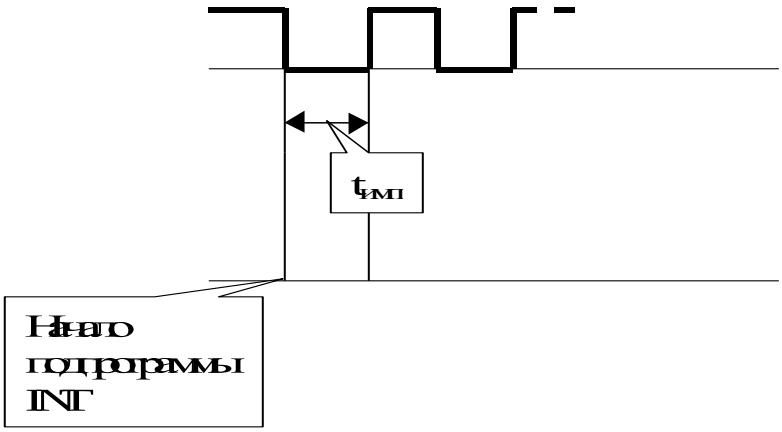


Рис. 7.28

Если в момент окончания подпрограммы INT сигнал на входе INT имеет еще нулевое значение, то подпрограмма будет выполняться еще один раз, а это значит, что вместо одного импульса будут зафиксированы два. Если же в момент окончания подпрограммы INT уровень сигнала на входе INT будет равным 1, то все нормально. Так как время выполнения подпрограммы INT равно 35 мкс, то $t_{имп} < 35$ мкс. Это ограничение для длительности импульса сверху. Существует ограничение для длительности импульса снизу. Так как сигнал на входе INT проверяется в каждом машинном цикле, то для того, чтобы сигнал не был пропущен, его длительность должна быть больше 2,5 мкс. Тогда можно записать $2{,}5\text{мкс} < t_{имп} < 35$ мкс.

Из рисунка 7.28 видно, что должны быть ограничения и на период входных импульсов. Если период входных импульсов так мал, что он укладывается по времени в интервал действия подпрограммы INT, то какие – то импульсы будут пропущены. Следовательно, должно быть $T_{имп} > t_{INT}$. Здесь надо учитывать то, что начало подпрограммы INT может быть задержано, если в момент прихода импульса (уровень 0) выполняется подпрограмма обработки прерывания от таймера. Время выполнения этой подпрограммы равно 15 мкс,

значит, ограничение на длительность периода входного сигнала выглядит так: $T_{имп} > t_{INT} +15$ мкс или $T_{имп} > 50$ мкс.

7.10.4. Реализация приема-передачи последовательного кода

7.10.4.1. Реализация приема последовательного кода

Последовательную передачу данных кодов КОИ8 можно программно реализовать на микроЭВМ. Формат передаваемых данных содержит стартовый бит, 8 бит данных, бит четности и два стоповых бита.

Бит старта D1	D2	D3	D4	D5	D6	D7	D8	Бит четности	Бит стоп	Бит стоп

В качестве вывода, через который будут передаваться данные, можно использовать любую свободную линию порта. Для управления одной линией порта (установка ‘1’ или ‘0’) необходимо использовать команды типа чтения – модификация – запись с использованием логических функций OPL или ANL.

Пример временной диаграммы принимаемой посылки, содержащей код числа 35h, показан ниже.

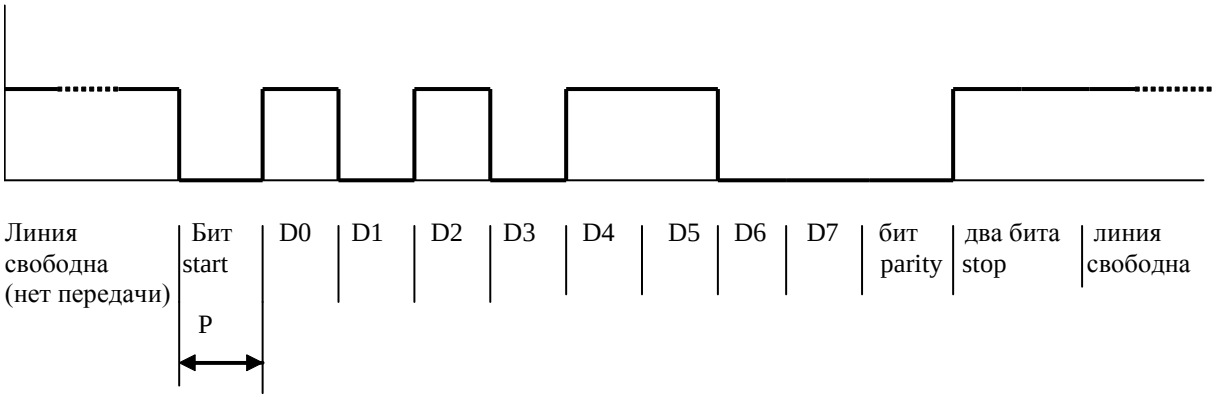


Рис. 7.29

Если в линии '1', значит, линия свободна от передачи. Посылка начинается нулевым битом start, далее следует посылка из 8 битов данных, начиная с младшего бита (в нашем примере – это число 35h). Далее следует бит четности (parity), который может принимать нулевое или единичное значение в зависимости от значения посылки данных. Бит четности дополняет до четного значения количество единиц в посылке данных. В нашем примере число единиц четно (4), значит, бит четности равен ‘0’. Бит четности защищает процесс передачи/приема от единичных помех. Контроль правильности посылки осуществляется на приемной стороне. Если сумма единиц в посылке

данных и бита четности – есть число нечетное, то приемная сторона делает вывод о том, что произошел единичный сбой при передаче, принятая посылка считается ошибочной и игнорируется. Скорость передачи – параметр, обратно пропорциональный периоду передачи одного бита Р. Период передачи одного бита – величина стабильная для выбранной скорости. Существует стандартный ряд скоростей передачи/приема. Некоторые значения из этого ряда: 1,2; 2,4; 4,8; 9,6; 19,2 килобод. Единицей измерения скорости передачи/приема является бод. 1 бод = 1 бит/сек.

Алгоритм программы регистрации принимаемых последовательных импульсов.

В начале выполнения программы биты принимаемых данных регистрируются без интервалов (один за другим), пока на линии не появится бит stop.

Далее начинается непосредственно прием байта. Схема приема ожидает появления бита start. Цель алгоритма заключается в возможно более точном обнаружении переднего фронта бита start. Этот момент времени будет использован как точка отсчета для регистрации всех последующих битов символа. После обнаружения переднего фронта бита start устанавливается задержка в течение половины длительности приема одного бита. Если на линии еще присутствует бит start, то он считается действительным и устанавливается временная задержка, равная Р. По окончании задержки Р обрабатывается первый бит данных и устанавливается новая задержка. Этот процесс повторяется до тех пор, пока не будут зарегистрированы все 8 бит данных и бит четности. Два последних принятых бита необходимо проверить, являются ли они битами stop. Если проверка подтверждает биты stop и проверка на четность дает положительный результат, считается, что символ достоверен, если нет, – произошла ошибка приема.

Рассмотренный метод требует значительных затрат мощности процессора на работу в цикле для реализации нужной временной задержки.

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

Эффективное распознавание стартового бита и увеличение точности синхронизации могут быть обеспечены в том случае, если система на базе микроЭВМ сможет определить передний фронт бита start как можно точнее. Достичь этого можно, подав принимаемые данные на вход Т1 микроЭВМ. При этом необходимо настроить таймер-счетчик на режим подсчета внешних событий и загрузить его числом FFH. При организации схемы приема, таким образом, передний фронт бита start будет вызывать прерывание по переполнению таймера-счетчика.

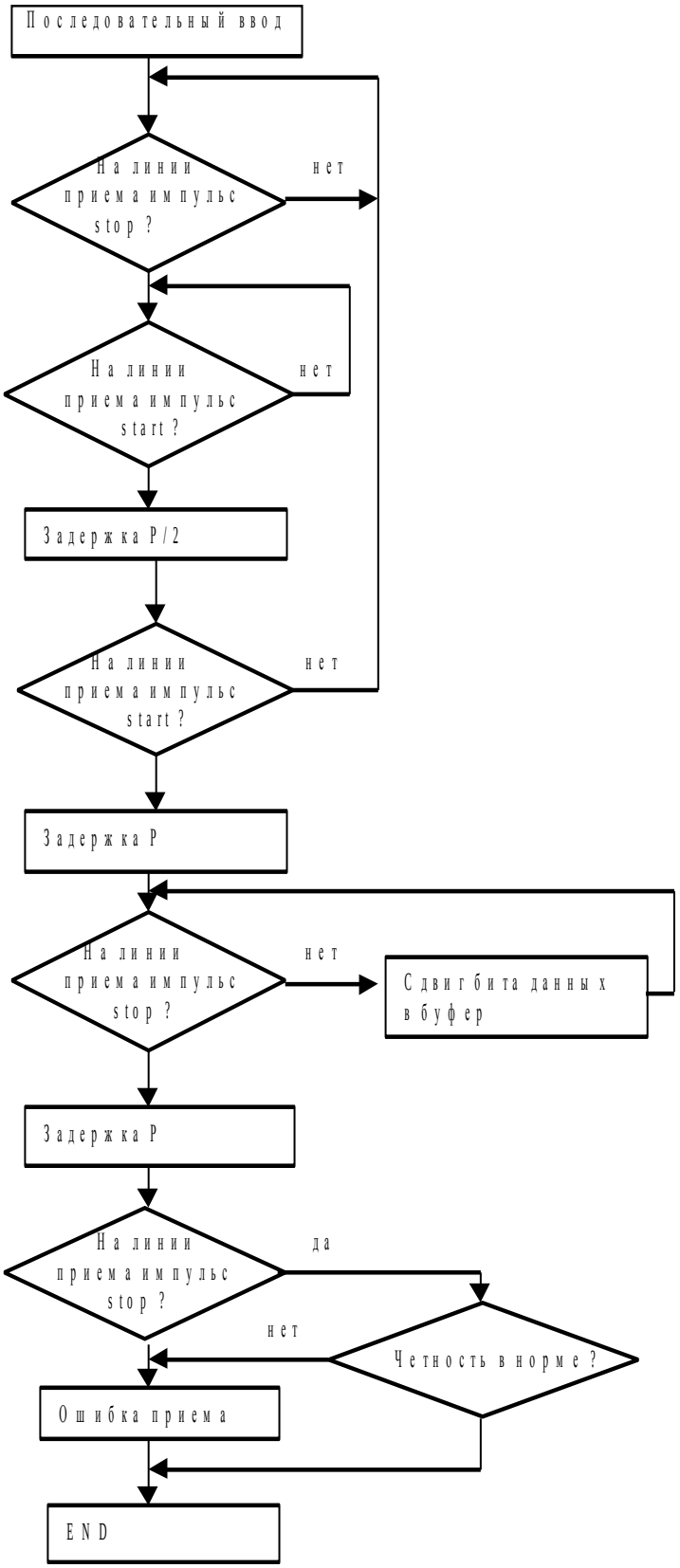

Последовательный ввод
На линии приема импульс stop ?
нет
На линии приема импульс start ?
нет
Задержка P/2
На линии приема импульс start ?
нет
Задержка P
На линии приема импульс stop ?
нет
Сдвиг бита данных в буфер
Задержка P
На линии приема импульс stop ?
да
нет
Четность в норме ?
Ошибка приема
END

Рис. 7.30

7.11.4.2. Реализация передачи последовательного кода

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

ПОСЛЕСЛОВИЕ

Персональный компьютер (ПК) стал обязательным атрибутом в любом современном офисе. Это основная техническая база информационной технологии. Профессионалы, работающие вне компьютерной сферы, считают непременной составляющей своей компетентности знание аппаратной части персонального компьютера, хотя бы его основных технических характеристик. Особенно велик интерес к компьютерам среди студентов, широко использующих их для своих целей.

Возможности ПК определяются характеристиками его функциональных блоков. Замена одних блоков на другие в настоящее время не представляет особой проблемы, и при необходимости можно достаточно быстро произвести модернизацию ПК. Однако современный рынок компьютерной техники столь разнообразен, что довольно не просто выбрать нужный блок, определить конфигурацию ПК с требуемыми характеристиками. Без специальных знаний здесь практически не обойтись.

В пособии автор попытался дать основное представление о структуре и функции основных частей ПК, помочь студентам сориентироваться на рынке технических средств компьютерной индустрии.

СПИСОК ЛИТЕРАТУРЫ

1. Задоя Н.И. Основы микропроцессорной техники: Уч. пособие / Рубцовский индустриальный институт. – Рубцовск, 2010. – 105 с.

2. Водовозов А.М. Элементы систем автоматики: Уч. пособие. – М.: Академия, 2006. – 224 с.

3. Лагин В.И. Электроника и микропроцессорная техника. Ростов на Дону: Феникс, 2007. – 576 с.

4. Безуглов Д.А. Цифровые устройства и микропроцессоры: Уч. пособие. – Ростов на Дону: Феникс, 2006. – 480 с.

5. Угринович Н.И. Информатика и информационные технологии: Учебное пособие. – М.: БИНОМ, 2001. – 464 с.

Задоя Николай Иванович

ЭЛЕМЕНТЫ ЦИФРОВОЙ АВТОМАТИКИ

Учебное пособие для бакалавров направления
«Электроэнергетика и электротехника»

Редактор Е.Ф. Изотова

Подписано в печать 09.10.14. Формат 60х84 /16.
Усл. печ. л. 6,56. Тираж 100 экз. Заказ 14 1301. Рег. №155.

Отпечатано в РИО Рубцовского индустриального института
658207, Рубцовск, ул. Тракторная, 2/6.