Лекции по предмету
Основы микропроцессорной техники (ОМПТ)

Тольяттинский государственный университет, 2006г.

Преподаватель: Прядилов Алексей Вадимович
Составитель: Кудинов Андрей Константинович

Лекция 1. (Вводная.)

Краткая история развития вычислительной техники.
Основные термины и понятия.

Достижения микропроцессорной техники широко используются практически во всех сферах человеческой деятельности. В связи с этим знание современными специалистами микропроцессорной техники становится обязательным условием успешной производственной деятельности. Чтобы овладеть знаниями необходимо, кроме усвоения теории, научиться решать конкретные задачи, связанные с выбором того или иного элемента, той или иной схемы микропроцессорной системы, овладеть хотя бы простейшими расчётами, понять физическую сущность явлений и процессов, происходящих в микропроцессорных системах, освоить элементарные приёмы программирования. Именно эту цель мы будем преследовать при изучении МП и МПУ.

1. АРХИТЕКТУРА МИКРОПРОЦЕССОРНЫХ СИСТЕМ

1.1. Краткие теоретические сведения

Первая электронная цифровая вычислительная машина (ЭВМ) была построена в США в 1946 году под руководством американского учёного Джона Неймана. Она содержала около 18000 ламп, которые использовались для построения арифметического и запоминающего устройств, основу которых составляли триггерные ячейки.

В 1950 году под руководством академика Сергея Алексеевича Лебедева была создана первая отечественная ЭВМ - малая электронная счётная машина (МЭСМ). Позже им же был создан ряд быстродействующих ЭВМ – "БЭСМ". Несмотря на своё несовершенство, первые ЭВМ продемонстрировали грандиозные возможности цифровой *вычислительной техники*, под которой подразумевают совокупность средств, предназначенных для ускорения и автоматизации процессов, связанных с вычислениями. Для автоматического выполнения сложных математических расчётов требуются не отдельные вычислительные устройства, а целые комплексы вычислительных и вспомогательных устройств с общей системой управления, которые и принято называть *ЭВМ* или *компьютерами*.

На рис. 1.1 представлена простейшая структурная схема ЭВМ.

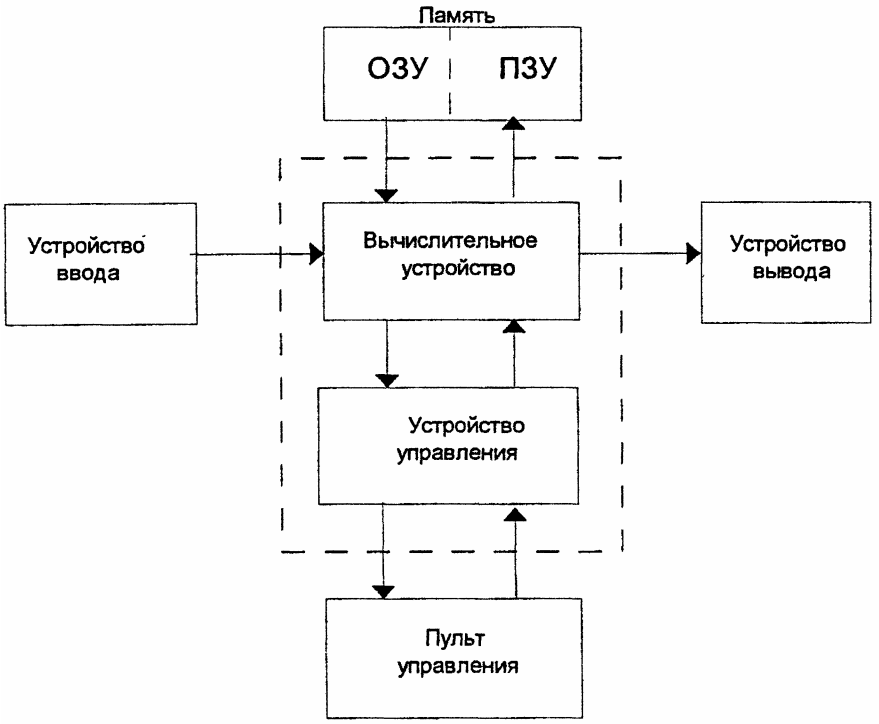


Рис. 1.1 Структурная схема ЭВМ (радиальная архитектура)

В общем случае вычислительная машина состоит из собственно *вычислительного устройства*, где выполняются арифметические и логические действия; *устройств ввода*, предназначенных для ввода исходных данных; *устройств вывода*, предназначенных для представления результатов вычислений в виде, удобном для практического использования; *устройства управления*, осуществляющего координацию работы всех устройств машины.

Вычислительное устройство совместно с устройством управления называют ***процессором***.

Память служит для приёма, хранения и выдачи чисел. Память делится на оперативное (ОЗУ) и постоянное (ПЗУ) запоминающие устройства. ОЗУ осуществляет приём, хранение и выдачу промежуточных результатов, а также данных и программ, участвующих в ближайшем ряде операций. ПЗУ хранит постоянные величины (константы), а также программы по которым выполняются часто встречающиеся (периодические) действия.

Управление режимами работы машины осуществляется с *пульта (терминала).*

Любое внешнее устройство, совершающее по отношению к микропроцессору операции ввода-вывода, можно назвать *периферийным*.

Управление реальными объектами осуществляется *контроллерами* - управляющими вычислительными машинами. Контролеры отличаются от обычных ЭВМ главным образом своими входными и выходными устройствами, а также режимом работы (не предусматривается возможность создания и модернизации программ). В качестве входных устройств здесь используются устройства связи с объектом (в простейшем случае - аналогово-цифровые преобразователи (АЦП)), а в качестве выходных -

преобразователи сигналов (в простейшем случае - цифро-аналоговые преобразователи (ЦАП)). Если объект сам представляет собой цифровую систему, до для связи с ним используются порты ввода- вывода.

Порт — это некая схема сопряжения, обычно включающая в себя один или несколько регистров ввода-вывода и позволяющая подключить периферийное устройство или управляемый объект к внешним шинам микропроцессора. Практически каждая микросхема (микропроцессор или микроконтроллер) использует для различных целей несколько портов ввода-вывода. Например, в персональном компьютере каждый порт имеет свой уникальный номер. Заметим, что номер порта — это, по сути, адрес регистра ввода-вывода, причем адресные пространства основной памяти и портов ввода-вывода не пересекаются.

Регистр представляет собой совокупность бистабильных устройств (то есть имеющих два устойчивых состояния), предназначенных для хранения информации и быстрого доступа к ней. В качестве таких устройств в интегральных схемах используют триггеры. Триггер в свою очередь выполнен на транзисторных переключателях (электронных ключах). В регистре из N триггеров можно запомнить слово из N бит информации.

Приведённая на рис. 1.1 структурная схема вычислительной машины носит название радиальной, а машины, построенные по такой схеме, - ЭВМ с радиальной архитектурой.

Архитектура ЭВМ - это искусство организации вычислительной машины, определяющее процесс обработки данных в конкретной вычислительной машине и включающее методы кодирования данных, состав, назначение, принципы взаимодействия технических средств и программного обеспечения. Существует большое многообразие ЭВМ с различными архитектурами, сущность которых определяется талантом и изобретательностью их разработчиков.

Вводная лекция (сокр. – для вечерников)

Микропроцессор – БИС, предназначенная ля обработки и манипуляции данных согласно инструкциям (командам).

Последовательность инструкций, подчиненная единому замыслу (алгоритму) называется программой.

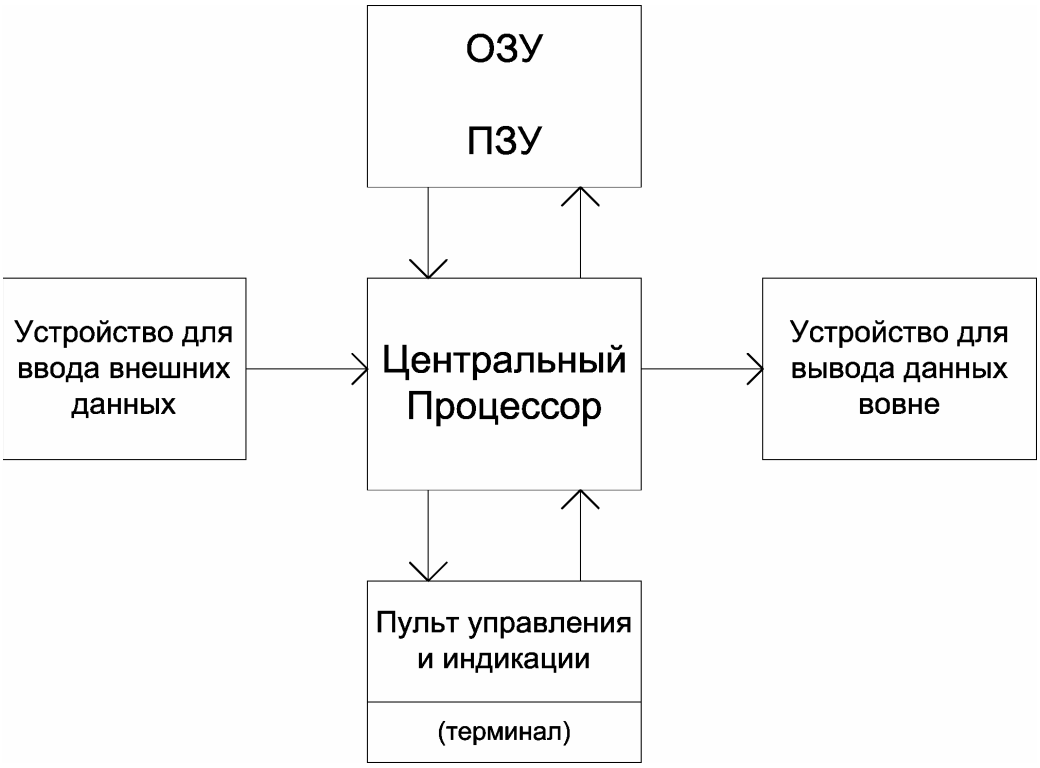
Программа хранится в специальном запоминающем устройстве – программной памяти.

Т.о. микропроцессор имеет в своем составе все устройства, необходимые для извлечения инструкций из памяти программ, их декодирования (расшифровки) и выполнения.

Для оперативного размещения промежуточных данных служит память данных. Работа МП с памятью данных ничем не отличается от работы с ПП.

Микро ЭВМ – набор БИС с общей системой управления, содержащей ЦП, память, устройство ввода-вывода, и устройства связи с внешними объектами, позволяющей выполнять, создавать и модифицировать программы.

Если все эти устройства выполнены на одном кристалле, ЭВМ называется однокристальной.

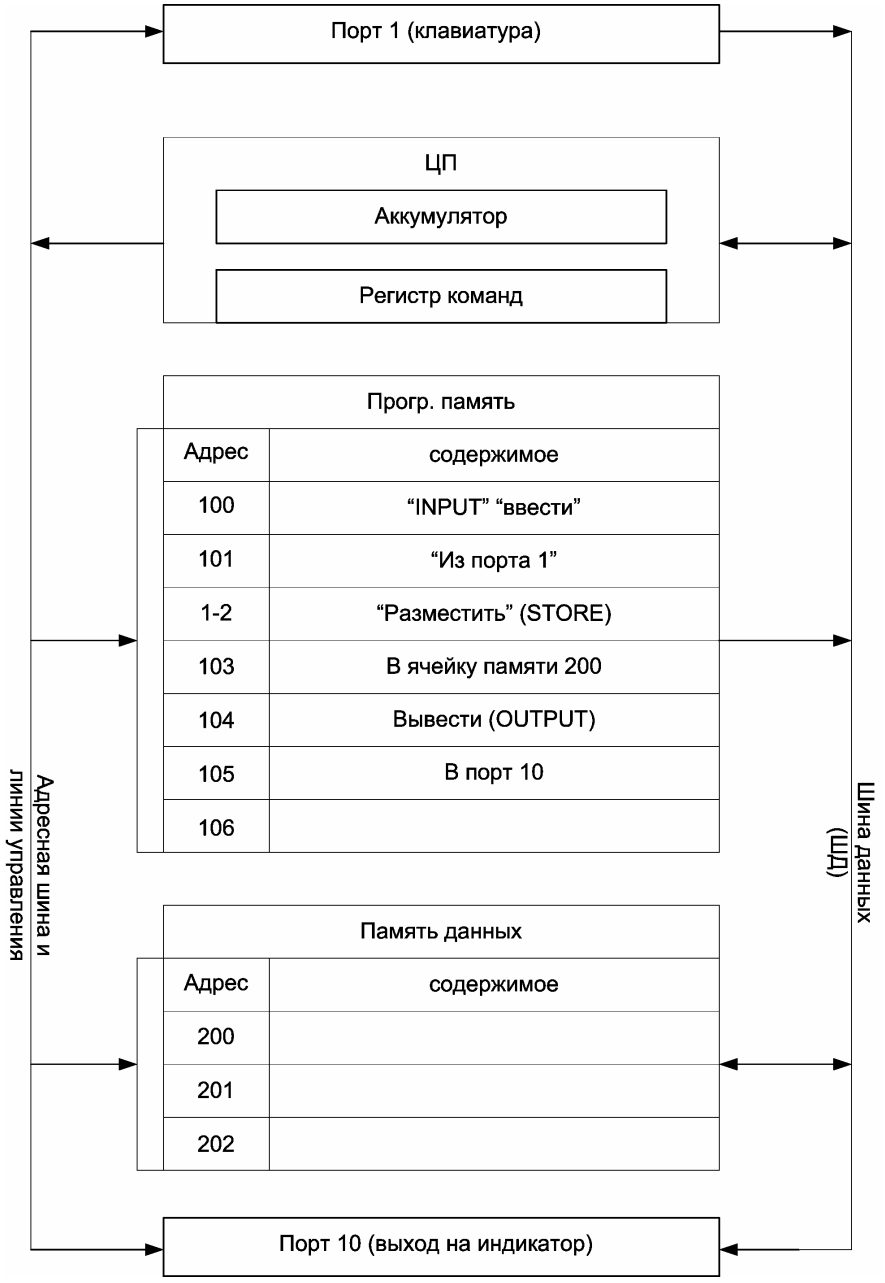


Структурная схема ЭВМ

Рассмотрим функционирование ЭВМ при выполнении обычного набора операций “ввод – размещение - вывод”. Такие операции выполняются, например, при нажатии клавиши на клавиатуре терминала.

Допустим, при нажатии клавиши должны выполниться команды согласно следующего алгоритма:

1. Ввести (INPUT) данные через порт ввода 1. (с клавиатуры)
2. Разместить (STORE) данные, поступившие из порта 1, в ячейке памяти данных 200.
3. Вывести (OUTPUT) данные через порт вывода 10. (на индикатор)



Последовательность событий, происходящих в микроЭВМ, будет такой:

1. МП выставляет адрес 100 на адресную шину и по соответствующей линии управления активизирует ПП. Данные из ячейки с адресом 100 поступают на ШД.
2. В специальный регистр (регистр команд) МП считывает информацию с ШД.
3. МП декодирует данные в регистре команд, выясняем, что это команда INPUT, и ей требуется указание номера устройства ввода.
4. МП выставляет на ША адрес 101 и по линии управления активирует ПП. В результате ячейка 101 соединяется с ШД.
5. МП считывает информацию (номер порта) с ШД и помещает ее в РК. Теперь он может декодировать команду полностью.
6. МП выставляет на ША адрес порта 1 и через линию управления активирует его (т.е. информация из порта 1 попадает на ШД).
7. МП считывает информацию с ШД (код нажатой клавиши) и размещает ее в специальном регистре – аккумуляторе. На этом выполнение первой команды закончено.
8. МП выставляет на ША адрес 102 и по линии управления активизирует ПП. Содержимое ячейки 102 (STORE) попадает на ШД и размещается микропроцессором в РК.
9. МП декодирует эту команду и определяет, что нужен операнд. Он выставляет на ША следующий адрес 103 и активизирует ПП. В результате на ШД появляется операнд (адрес ячейки памяти 200).
10. МП считывает операнд и декодирует команду. Далее начинается процесс ее выполнения:
11. МП выдает на ШД информацию из аккумулятора (код нажатой клавиши), выставляет на ША адрес ячейки ОЗУ – 200, и по линии управления активизирует вход записи микросхемы. Данные с IL записываются в ячейку с адресом 200. На этом выполнение второй команды заканчивается.
12. Третья команда...

Литература:

1. Ч. Гилмор. Введение в микропроцессорную технику: Пер. с англ. – М: Мир, 1984. – 334с.
2. Р. Токхайм. Микропроцессоры: курс и упражнения: Пер. с англ. – М: Энергоатомиздат, 1987. – 336с.
3. В.-Б.Б. Арбайшис и др. Микропроцессоры и микропроцессорные комплекты интегральных микросхем: Справочник в 2-х томах. – М., Р. И С. 1988.

Лекция 2 Магистральная архитектура

Структурная схема ЭВМ с магистральной стуктурой приведена на рис.1

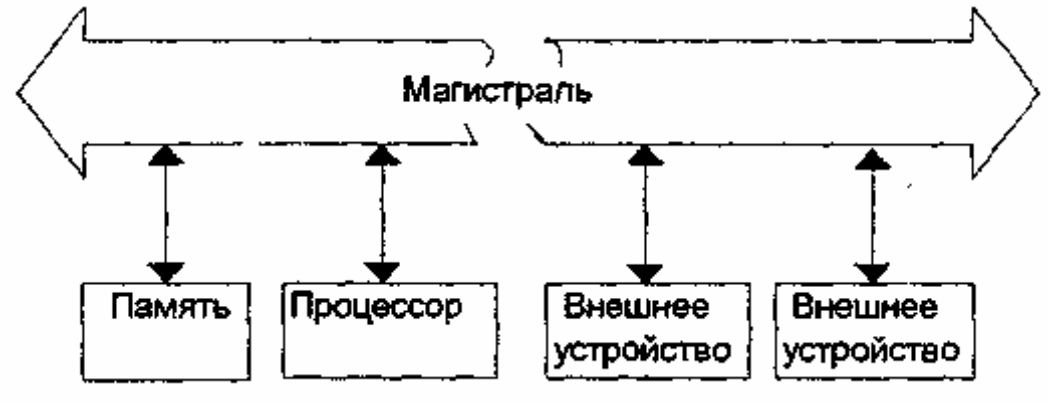


Рис 1. Структурная схема ЭВМ с магистральным принципом построения

Магистраль (шина) - это группа линий передачи информации, объединённых общим функциональным признаком (например, магистраль данных, адресов, управления). Магистраль представляет собой набор проводников, к которым параллельно подсоединены все компоненты ЭВМ. Посылая по магистрали электрические сигналы, любая компонента ЭВМ может передавать информацию другим компонентам. В определённый момент времени обмен информацией может происходить только между определёнными двумя компонентами ЭВМ.

В современных ЭВМ обмен информацией происходит на частотах десятков мегагерц, что позволяет рассматривать магистраль как длинную линию, т.е. такую линию, физические размеры которой сравнимы с длинной волны передаваемого по ней сигнала.

В длинных линиях время распространения тока и напряжения получается (такого же порядка, что и период передаваемых сигналов. Вследствие этого в таких линиях процессы рассматриваются не только во времени, но и в пространстве.

Когда к длинной линии подключен генератор переменной ЭДС, то вдоль линии движется волна, представляющая собой распространение электромагнитного поля в одном направлении, в данном случае от генератора к концу линии. Для каждой линии отношение амплитуды напряжения волны к амплитуде тока является постоянной величиной и называется волновым вопротивлением линии ρ.

Волновое сопротивление линии зависит от её конструкции и определяется по формуле

$$\rho = \sqrt{\frac{L}{C}},$$

где L и C соответственно распределенные индуктивность и емкость линии.

Движущаяся волна, дойдя до конца линии, отражается и двигается

обратно к генератору. Таким образом, в линии распространяются две волны: падающая - движется от генератора и отраженная - движется в обратном гаправлении. Сложение этих двух волн приводит к искажению передаваемого сигнала и потере работоспособности магистрали.

Для характеристики режима линии пользуются коэффициентом бегущей юлны (кбв), введенным А.А. Пистолькорсом в 1927 г.:

$$кбв = \frac{\rho}{R}$$

При ρ= R , кбв=1. Другими словами, отраженной волны не будет, если линию нагрузить активным сопротивлением R, равным волновому сопротивлению линии ρ.

Для образования режима бегущей волны в линии (т.е. такого режима, когда отраженная волна отсутствует) магистраль должна быть согласована на концах. Согласование магистралей производят с помощью заглушек, схема которых показана на рис. 2

Кроме согласования волнового сопротивления линии, заглушка должна обеспечить требуемый уровень напряжения на линии.

Поскольку шина питания и общий провод в заглушке соединены между собой конденсаторами, сопротивления которых на высоких частотах порядка 10 МГц в соответствии с

$$Z_c = \frac{1}{2\pi \cdot fC},$$

где f - частота в Гц, С - емкость в Ф, становится пренебрежимо малым, эквивалентная схема заглушки на рабочих частотах будет представлять параллельное соединение сопротивлений (рис. 3).

Рис.2 Схема согласования линии магистрали

Рис. 3 Эквивалентная схема согласования магистрали

Таким образом, для заглушки, выполненной, как показано на рис. 2, условием согласования линии магистрали является равенство

R1||R2=ρ.

В современных ЭВМ за каждой линией закреплен соответствующий

сигнал, причем конструктивно за каждым сигналом закреплен один и тот же контакт на всех разъемах, предназначенных для подключения внешних устройств к магистрали. Последнее позволяет гибко менять состав и структуру ЭВМ, осуществлять наращивание ее аппаратных средств в зависимости от требования решаемого круга задач.

Связь между блоками, подключенными к магистрали, осуществляется, как правило, с помощью трех шин (групп линий магистрали): шины адресов, шины данных и шины управления.

Шина данных - это двунаправленные линии, по которым передается информация между устройствами ЭВМ. Процесс однократного обращения к шине данных называется **шинным циклом**. Шинный цикл обычно состоит из нескольких периодов - машинных тактов и сопровождается сигналами управления, определяющими тип цикла (цикл чтения данных, цикл записи данных и т.д.). Если одним из устройств, участвующих в обмене данными, является процессор, то сигналы на линии управления поступают от процессора.

Кроме сигналов управления процессор выставляет сигналы адреса на линии **шины адреса**, определяющие номер устройства, с которым в данный момент происходит обмен данными.

Кроме процессора, сигналы для линий управления и линий адреса могут формироваться некоторыми устройствами, так называемыми контролерами прямого доступа к памяти.

Рассмотрим подробнее работу и описание функций отдельных линий магистрали на примере системной шины ЭВМ типа IBM PC ISA-8

ISA Bus (Industry Standard Architecture) — шина расширения, применявшаяся с первых моделей PC и ставшая промышленным стандартом. В компьютере XT применялась шина с разрядностью данных 8 бит и адреса — 20 бит. В компьютерах AT шину расширили до 16 бит данных и 24 бит адреса. В таком виде она существует и поныне как самая распространенная шина для периферийных адаптеров. Конструктивно шина выполнена в виде двух щелевых разъемов (слотов) с шагом выводов 2,54 мм (0,1 дюйма), вид которых изображен на рис4. Подмножество ISA-8 использует только 62-контактный слот (ряды A, B), в ISA-16 применяется дополнительный 36-контактный слот (ряды C, D).

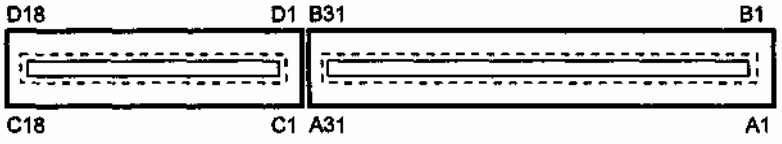


Рис. 4 Внешний вид слота ISA

Назначение контактов слотов шины ISA приведено в табл. 1.

Таблица 1. Разъемы шин ISA-8

Ряд B	N#	Ряд A
GND	1	IOCHK
Reset	2	Data7
+5 B	3	Data 6
IRQ 2/9[1]	4	Data 5
-5B	5	Data 4
DRQ2	6	Data 3
-12B	7	Data 2
OWS#[2]	8	Data 1
+12 B	9	Data 0
GND	10	IOCHRDY
MemWR#	11	AEN
MemRD#	12	Addr 19
IOWR#	13	Addr 18
IO RD#	14	Addr 17
DACK 3#	15	Addr 16
DRQ3	16	Addr 15

Ряд B	N#	Ряд A
DACK 1#	17	Addr 14
DRQ1	18	Addr 13
Refr#	19	Addr 12
Bclock	20	Addr 11
IRQ7	21	Addr 10
IRQ 6	22	Addr 9
IRQ 5	23	Addr 8
IRQ 4	24	Addr 7
IRQ3	25	Addr 6
DACK 2#	26	Addr 5
TC	27	Addr 4
ALE	28	Addr3
+5 B	29	Addr 2
Osc.	30	Addr 1
GND	31	Addr 0

1 B4: XT=IRQ2, AT=IRQ9.

2 B8: XT=Card Selected.

Сигналы шины ISA имеют корни в шинах Microbus и Multibus, они естественны для периферийных микросхем фирмы Intel семейств 8080 и 80x86/88. Набор сигналов 8-битной шины ISA предельно прост. Непосредственно к программному обращению к ячейкам памяти и пространства ввода/вывода относятся следующие сигналы:

Data[7:0] — шина данных.

Addr[19:0] — шина адреса.

AEN — Adress ENable, разрешение адресации портов (запрещает ложную дешифрацию адреса в цикле ПДП (DMA). Этот сигнал выдается контроллером ПДП, и указывает, что идет выполнение цикла ПДП.

Прямой доступ к памяти (ПДП) -это способность системы передавать или принимать данные непосредственно от устройств ввода-вывода в ОЗУ или из ОЗУ без использования процессора. Для инициирования ПДП, контролер по одной из линий шины управления, выделенной для этой цели, выставляет запрос ПДП в процессор. Процессор, получив запрос ПДП, отключается от линий магистрали (переводит свои выводы в состояние с высоким сопротивлением (так называемое высокоимпедансное состояние) и выдаёт на шину управления подтверждение ПДП. Получив подтверждение ПДП, контроллер ПДП захватывает магистраль, т.е. становится задатчиком сигналов на шине управления и шине адреса. Закончив обмен данными, контролер ПДП снимает запрос ПДП, и процессор возобновляет свою работу. ПДП позволяет увеличить скорость обмена данными в системе, поскольку в режиме ПДП обмен данными между элементами системы происходит

непосредственно, минуя процессор.

DRQ[1:3] — Data ReQuest, запросы 8-битных каналов DMA (положительным перепадом).

DACK[1:3]# — подтверждение запросов 8-битных каналов DMA.

TC — признак завершения счетчика циклов DMA.

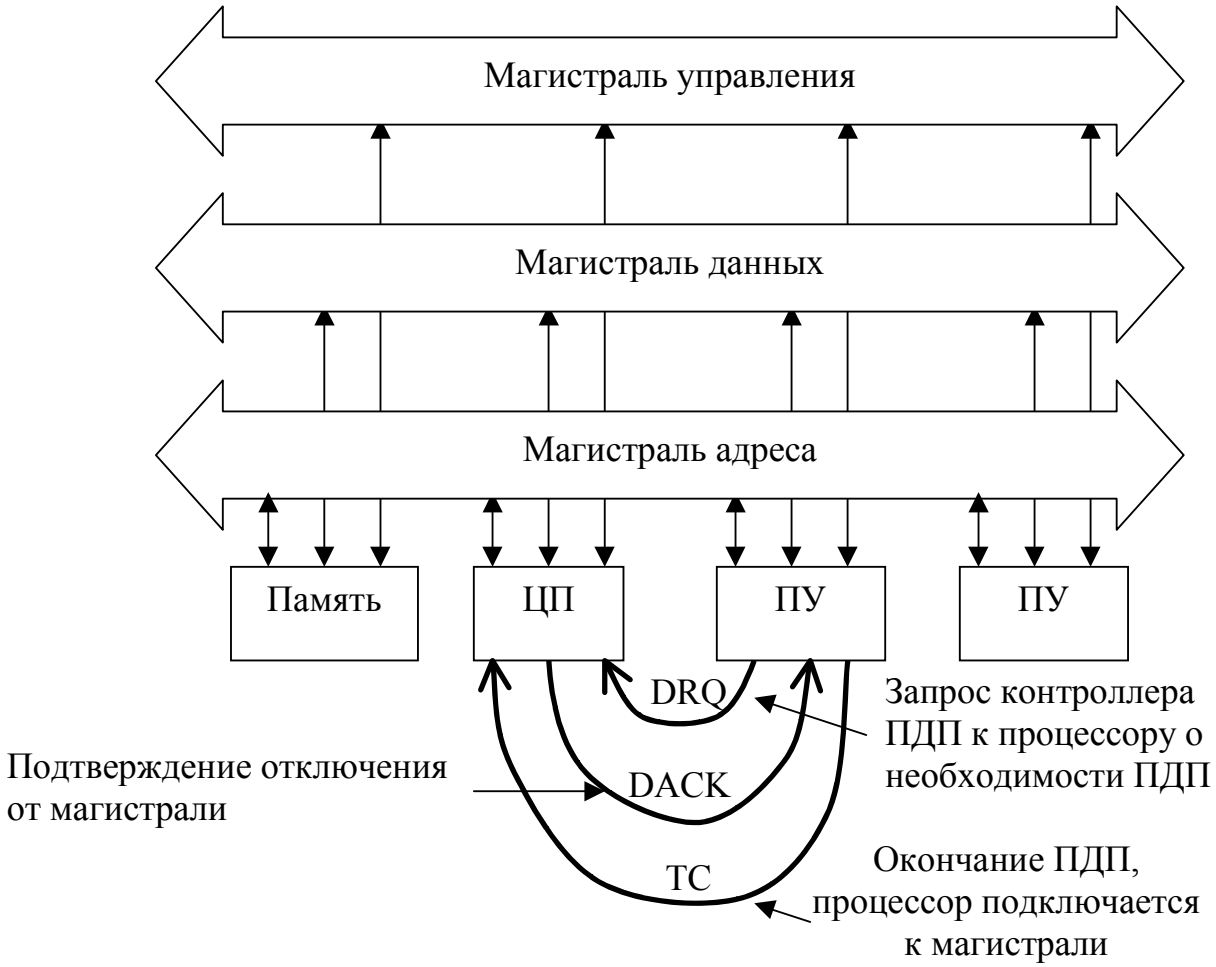


Рис.5 Порядок захвата и освобождения шин в цикле ПДП.

IOWR# — запись в порт.

IORD# — чтение порта.

MemWR# — запись в память (в диапазоне адресов 0-FFFFFh).

MemRD# — чтение памяти (в диапазоне адресов 0-FFFFFh).

Шина имеет и несколько служебных сигналов синхронизации, сброса, регенерации памяти, установленной на адаптерах:

IOCHRDY — I/O Channel Ready, "готовность устройства", низкий уровень удлиняет текущий шинный цикл (не более 15 мкс) при работе с медленными запоминающими и внешними устройствами.

ALE — Adress Lock Enable, разрешение защелки адреса. После его спада в каждом цикле процессора линии Addr 0-19 гарантированно содержат действительный адрес. Этот сигнал инициируется процессором и указывает на начало шинного цикла (по фронту).

Refr# — цикл регенерации памяти (в XT он называется DACK 0#). Сигнал появляется каждые 15 мкс, при этом шина адреса указывает на очередную регенерируемую строку памяти.

IOCHK — контроль канала, низкий уровень информирует процессор о том, что в данных, поступивших из памяти или устройства вв./вывода содержится ошибка контроля четности.

Reset — сигнал аппаратного сброса (активный уровень — высокий).

BClock — Bus Clock, синхронизация шины с частотой около 8 МГц. Периферийные устройства могут и не использовать этот сигнал, работая только по управляющим сигналам записи и чтения.

OSC — Oscilliate, несинхронизированная с шиной частота 14,431818 МГц (использовалась старыми дисплейными адаптерами).

Обобщенные временные диаграммы циклов чтения или записи памяти или ввода/вывода приведены на рис. 6. Здесь условный сигнал CMD* изображает один из сигналов:

MEMRD# — в цикле чтения памяти;

MEMWR# — в цикле записи памяти;

IORD# — в цикле чтения порта ввода/вывода;

IOWR# — в цикле записи порта ввода/вывода.

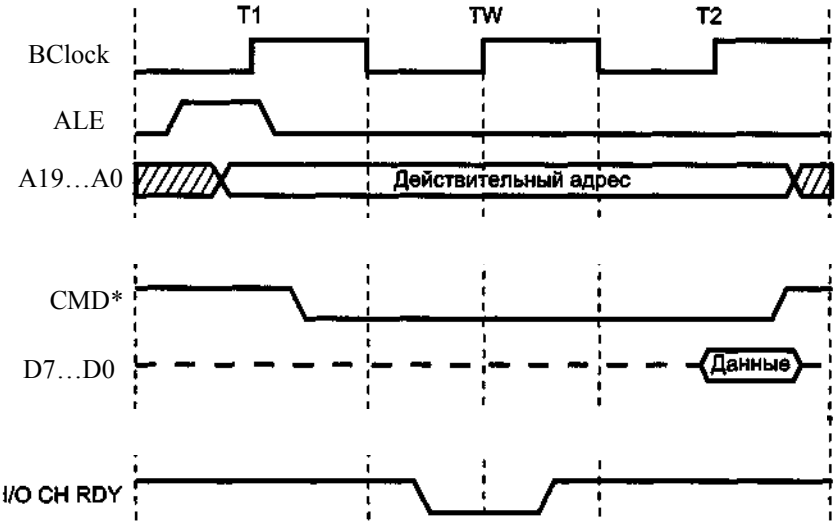


Рис. 6. Временные диаграммы циклов чтения или записи на шине ISA

В каждом из рассматриваемых циклов активными (с низким уровнем) могут быть только сигналы (сигнал) лишь из одной строки данного списка.

По адресованному ему спаду сигнала чтения устройство должно выдать на шину данных содержимое адресуемой ячейки и удерживать его, пока не произойдет подъем данного сигнала. Во время циклов записи процессор выставляет действительные данные несколько позже начала (спада) сигнала записи, и устройство должно для себя фиксировать эти данные в конце цикла по подъему сигнала записи. Обращение к портам ввода/вывода аналогичное.

Минимальная длительность цикла определяется чипсетом и может программироваться опциями BIOS Setup через количества тактов ожидания. При этом циклы обращения к памяти обычно короче циклов обращения к портам ввода/вывода. Если устройство не вписывается в заданные циклы, оно может вводить дополнительные **такты ожидания**, используя сигнал IOCHRDY, но при этом недопустимо удлинять цикл более, чем на 15

микросекунд.

0WS – сигнал от устройства, разрешающий системной шине ускорить текущий цикл (устранить такты ожидания)

Иногда в процессе работы системы, в моменты времени, заранее не известные, может возникнуть некоторое внешнее условие, требующее обработки (обслуживания) специальной программой обработки (подпрограммой). Реакция процессора на это условие называется **прерыванием.**

IRQ2/9, IRQ[3:7] — запросы прерываний. Положительный перепад сигнала вызывает запрос аппаратного прерывания. Для идентификации источника высокий уровень должен сохраняться до подтверждения прерывания процессором, что затрудняет разделяемое использование линий запроса.

Линия IRQ2/9 в шинах XT вызывает аппаратное прерывание с номером 2, а в AT — с номером 9.

После подтверждения прерывания устройство, требовавшее обслуживания, выставляет на шину данных так называемый вектор прерывания (номер(адрес) ячейки памяти, с которой начинается подпрограмма обслуживания прерывания). Процессор считывает вектор прерывания и переходит к выполнению подпрограммы обслуживания прерывания. По окончанию обслуживания процессор возобновляет свою работу по основной программе с места, в котором произошло прерывание.

Чтобы возобновить работу с того места, в котором произошло прерывание, необходимо запомнить промежуточные результаты работы процессора, а также адрес ячейки ОЗУ, в которой находится команда, которая бы выполнялась следующей, если бы прерывание не произошло. Для этих целей в ОЗУ выделяется некоторое количество ячеек, называемых **стеком.**

Стек - это структура данных или устройство памяти, организованное по принципу "последним вошёл - первым вышел", т.е. данные, поступившие в стек, будут извлекаться из него в обратном порядке.

Для организации стека в процессоре предусмотрен специальный указатель стека. Указатель стека - это регистр-счётчик, содержимым которого всегда является адрес. Указатель стека загружается адресом, представляющим собой вершину стека (точку входа в стек). Рассмотрим функционирование стека на следующем примере. Пусть в момент возникновения прерывания процессор должен был бы приступить к выполнению команды с адресом в ОЗУ 800AH, при этом промежуточные результаты, которые необходимо сохранить, обозначим как A, а указатель стека пусть содержит 220AH, что на единицу старше первой ячейки памяти стека 2209H. Последовательность событий при работе со стеком здесь может быть следующей.

1. Указатель стека декрементируется (уменьшается на единицу) от 220AH до 2209H.

2. В ячейку памяти с адресом с адресом 2209H помещаются данные

800АН.

3. Указатель стека снова декрементируется от 2209Н до 2208Н.

4. В ячейку памяти с адресом 2208Н загружаются данные А.

Стек может продолжать расти, если в процессе обслуживания прерывания возникнет следующее прерывание (так называемое вложенное прерывание) и т.д. Стек не имеет ограничений за исключением тех, которые обусловлены наличием других программ в ОЗУ (т.е. стек не должен перекрывать ячейки ОЗУ, содержащие другие программы и их данные).

По окончании обслуживания прерывания работа со стеком будет следующей.

1. Указатель стека указывает на ячейку 2208Н.

2. Из ячейки с адресом 2208Н выбираются данные А.

3. Содержимое указателя стека инкрементируется (увеличивается на единицу) с 2208Н до 2209Н.

4. Из ячейки памяти с адресом 2209Н выбираются данные 800АН.

5. Содержимое указателя стека снова инкрементируется от 2209Н до 220АН.

Механизм прерывания является одной из важнейших характеристик компьютера, предоставляющий системе эффективное средство быстрого отклика на непредсказуемые события. Обработка прерываний повышает пропускную способность вычислительной системы, позволяет периферийным устройствам выдавать на микропроцессор запросы на обслуживание в тех случаях, когда они в нем нуждаются. Это гораздо эффективнее опроса периферийных устройств с целью выявления того, необходимо ли им обслуживание.

Кроме логических сигналов шина имеет контакты для разводки питания +5, -5, +12 и -12 В.

Лекция 3 Микропроцессоры и микропроцессорные комплекты

Микропроцессор — это программно управляемое, функционально законченное устройство, предназначенное для обработки цифровой информации и построенное на одной или нескольких больших интегральных схемах (БИС).

Совокупность совместимых по основным параметрам БИС, из которых можно строить различные по сложности и назначению микропроцессорные системы, включая микроконтроллеры и микро-ЭВМ, называют **микропроцессорным комплектом**. В этот комплект, кроме основного микропроцессорного элемента, выполняющего функцию центрального процессора, входят также интерфейсные БИС, позволяющие организовать прием и выдачу цифровой информации, БИС памяти для хранения информации и программ и некоторые вспомогательные микросхемы, служащие для ускорения процесса обработки информации и повышающие организационную эффективность вычислительной системы.

Краткая история развития МПК

Технология МП коплектов развивается в основном, по двум направлениям – МОП (МДП) и биполярные технологии.

Первые МП строились по p-МОП технологии, что обеспечивало простоту изготовления и низкую стоимость. Недостатки – принципиальное ограничение по быстродействию. Пример – К145, КР1814.

n-МОП:

- быстродействие – выше на порядок, чем у p-МОП
- больше плотность упаковки
- относительная простота изготовления
- невысокая стоимость

Пример – КР580, КР1801, КР1810 (i8088)

к-МОП:

- малое энергопотребление
- высокая помехоустойчивость
- надежная работа в широком интервале температур и питающих напряжений.

Пример – КР588

Биполярные технологии обеспечивают более высокое быстродействие и меньшее энергопотребление (на высоких частотах).

ТТЛШ (по отношению к ТТЛ):

- низкя потребляемая мощность
- выше быстродействие

Пример – К589, КР1802, КМ1804

ЭСЛ: самое высокое быстродействие.

Пример – К1800

Процессоры до i486 выполнялись по МОП технологии, i486 – с элементами биполярных технологий, i-Pentium – биполярная технология.

Характеристики МП как вычислительной машины

С точки зрения программиста МП характеризуется разрядностью, адресным пространством и системой команд.

1. **Разрядность.**

Каждый МП оперирует данными, представленными словами определенной длины. В настоящее время типичными являются слова длиной 4, 8, 12, 16, 32 бит (или разрядов). Например, 8-битовое слово – байт:

10011011

Байт – $8^{ми}$-битовый элемент данных, обрабатываемый как единое целое и имеющий один адрес.

Разрядность вычислительной машины определяет точность вычислений (или производительность при заданной точности). Существуют МП с фиксированной и наращиваемой разрядностью.

Фиксированная: КР580, КР588, КР1801, КР1810, и др.

Наращиваемая: К589, К1800, К1802, К1804.

МП с наращиваемой разрядностью строятся на основе секций, например, по 4 разряда, путем установки соответствующих перемычек на плате.

Адресное пространство определяется числом адресных линий n и определяет максимальный размер M непосредственно адресуемой памяти

$M=2^n$

Например, МП КР580ВМ80А имеет n=16 линий адреса, что позволяет ему непосредственно адресоваться к 2^{16}=65536 ячейкам (байтам) памяти.

Система команд – характеристика, которая определяется совокупностью операций, обеспечивающих выполнение программы в соответствии с заданным алгоритмом.

Программа – это последовательность предписаний (команд), определяющая порядок выполнения операций при реализации заданного алгоритма.

В систему команд входят:

- форматы команд и обрабатываемых данных;
- число команд;
- способы адресации данных;
- длина адреса (см. адресное пространство);
- объем и организация стека;
- способы обработки прерываний;

- организация ввода-вывода.

Формат команды – это схема расположения двоичных цифр или знаков команды, позволяющая различать ее составные части (поля), определять их разрядность и функциональное назначение.

Выбор МП

При выборе МП как правило, руководствуются его быстродействием и потребляемой мощностью и стоимостью.

Быстродействие – характеристика, которая определяется схемотехническими и архитектурными возможностями МП комплекта. Количественно выражается числом выполняемых в секунду операций типа:

регистр - регистр (пересылка)

регистр - память (чтение - запись)

сложение, умножение (арифметические операции)

Однако, ни одна из них не определяет в полной мере быстродействие всей вычичлительной системы. Наиболее комплексной и объективной оценкой быстродействия МП или комплекта является способ эталонного программирования. Здесь для определенного набора эталонных задач, отражающих специфику той или иной области, для которой проектируется микропроцессорная система, производится пробное программирование. Исходя из времени решения эталонного пакета программ, делается вывод о быстродействии и осуществляется выбор МП комплекта.

Потребляемая мощность – важный параметр, определяющий сферу применения МП комплекта (стационарные либо мобильные системы).

Все выпускаемые в настоящее время микропроцессоры по способу обработки информации можно разделить на два класса: с аппаратной и микропрограммной обработкой информации. Ниже рассмотрены серийно выпускаемые микропроцессорные БИС из тех комплектов, где центральный процессорный элемент выполняет обработку информации аппаратным способом.

Отличительными особенностями микропроцессоров с аппаратной обработкой информации являются фиксированная разрядность передаваемых, принимаемых и обрабатываемых в одном такте информационных слов, а также фиксированный перечень аппаратно выполняемых команд. Эти микропроцессоры имеют весьма сложную внутреннюю структуру.

МП комплект серии КР580

Данный комплект выполнен по n-МОП и ТТЛШ технологии. Характеризуется архитектурным единством, унификацией интерфейса, программируемостью микросхем, их логической и электрической

совместимостью. Комплект предназначен для работы в диапазоне температур от -10 до +70°С.

Состав комплекта:

КР580ВМ80А – однокристальный $8^{ми}$-разрядный МП, n-МОП;

ВВ51 – программируемый синхронно- асинхронный приемопередатчик для каналов последовательной связи, n-МОП;

ВВ55 – программируемое устройство ввода - вывода параллельной информации, n-МОП;

ВТ57 – программируемый четырехканальный контроллер ПДП, n-МОП;

ВИ53 – программируемый таймер, n-МОП;

ВН59 – программируемый контроллер прерываний, n-МОП;

ВГ75 – программируемый контроллер видеотерминала, n-МОП;

ВД79 – контроллер клавиатуры/ дисплея, n-МОП;

ГФ24 – формирователь тактовых импульсов с несовпадающими фазами, n-МОП;

ВА86 – двунаправленный шинный формирователь с прямым выходом, n-МОП;

ВА87 – двунаправленный шинный формирователь с инверсным выходом, n-МОП;

ВК28 – системный контроллери шинный вормирователь, ТТЛШ;

ВК38 – системный контроллери шинный вормирователь, ТТЛШ;

ИР82 – буферный регистр, ТТЛШ;

ИР83 – буферный регистр, ТТЛШ;

Лекция 4

Микросхема КР580ВМ80А —функционально законченный однокристалльный микропроцессор с фиксированной системой команд. Применяется в устройствах обработки данных и управления.

Микропроцессор имеет раздельные 16-разрядный канал адреса и 8-разрядный канал данных. Канал адреса обеспечивает прямую адресацию к внешней памяти объемом до 65 536 байт и 256 устройствам ввода и вывода.

Условное графическое обозначение микросхемы показано на рисунке 1, назначение выводов указано в таблице 1. Структурная схема микропроцессора изображена на рисунке 2.

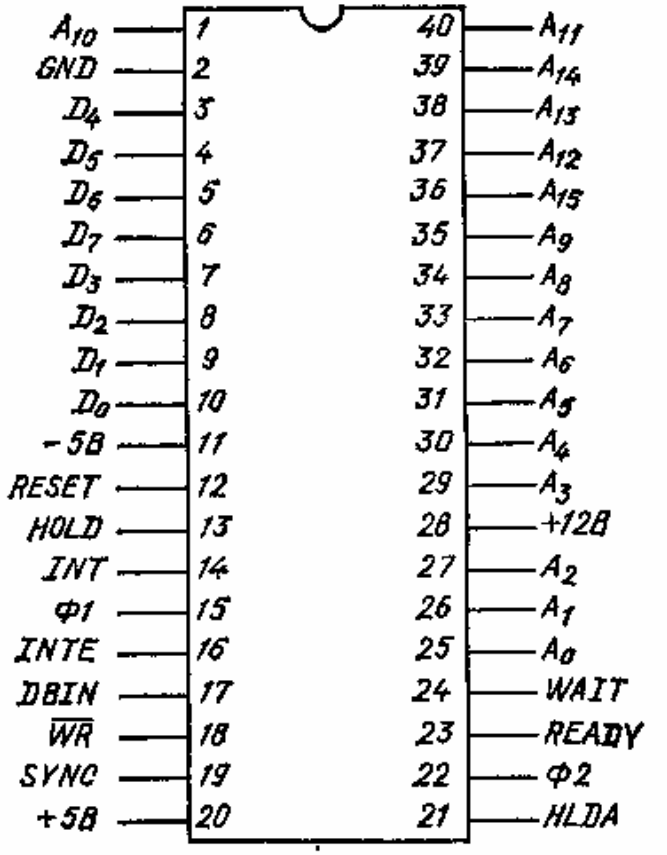


Рис.1 Условное графическое обозначение КР580ВМ80А

Таблица 1

Выводы	Назначение	Вход или выход
GND, +5В, –5В, +12В	Питание	Входы
Ф1, Ф2	Тактовые импульсы	Входы
$D_0 – D_7$	Линии данных	Двунаправленные
$A_0 – A_{15}$	Линии адресов	Выходы
SYNC	Синхронизация	Выход
DBIN	Строб входных данных	Выход
WAIT	Ожидание	Выход
WR#	Строб записи	Выход
HLDA	Подтверждение захвата	Выход
INTE	Разрешение прерывания	Выход

READY	Готовность ввода данных	Вход
HOLD	Захват	Вход
INT	Требование прерывания	Вход
RESET	Сброс	Вход

МП содержит следующие основные функциональные блоки:

1. Устройство обработки информации (УОИ), которое, в свою очередь, включает в себя арифметико- логическое устройство (АЛУ); накопительный регистр – аккумулятор (А); буфер аккумулятора (БА); оперативный регистр временного хранения данных (ОРВХ); регистр признаков F (или флагов) – регистр (РП) и десятичный корректор (КД).

2. Блок регистров (БР), состоящий из программного счетчика адреса (*PC*– от Program counter); указателя адреса последней занятой ячейки стека (*SP* – от Stack pointer) и регистров общего назначения (РОН) B, C, D, E, H, L, вспомогательных регистров W, Z, адресного регистра/защелки (РА) шины адреса с инкрементором - декрементором (И-Д) и мультиплексора (МПЛ).

3. Устройство управления и синхронизации (УУС), которое координирует работу всех функциональных блоков МП, формируя необходимые команды и машинные циклы в зависимости от сигналов внешних устройств (ВУ), и сигналов, поступающих от ДК. УУС вырабатывает также сигналы управления для ВУ. Для этого УУС связано с 12-разрядной шиной управления (ШУ), в которой 6 цепей являются входными и столько же выходными.

4. Буферный регистр данных (БРД) и регистр - защелка, предназначенные для фиксации информации и логического электрического разделения внутренней шины данных от внешней системной ШД. БРД представляет собой комплекс двунаправленных схем с тремя устойчивыми состояниями (односторонняя проводимость в прямом и обратном направлении, полное электрическое отключение).

5. Буферный регистр адреса (БРА): в отличие от БРД является однонаправленным. Он предназначен для передачи адресов команд и данных из МП в память, а также номеров внешних устройств в систему.

Кроме того, ЦП имеет внутреннюю ШД, через которую происходит обмен данными между внутренними регистрами МП. Внешняя шина данных предназначена для двустороннего обмена информацией между МП и другими устройствами системы. Шина адреса ША обеспечивает адресацию памяти и устройств ввода- вывода.

АЛУ является параллельным 8-разрядным устройством. Оно обеспечивает выполнение основных операций по обработке данных, а также осуществляет простейшие арифметические (сложение, вычитание) и логические (И, ИЛИ, исключающее ИЛИ, НЕ, сдвиги) действия. Данные в АЛУ могут поступать из оперативного регистра вр. хр. ОР, буфера аккумулятора БА, регистра признаков РП. ОР и БА предназначены для приема и хранения информации на время выполнения команды 8-разрядных операндов.

Результаты операций передаются либо на внутреннюю ШД с последующей передачей в блок регистров БР или БРД, либо в А.

АЛУ и А также связаны с регистром признаков РП. Последний используется для индикации результатов операций.

На рис. 3 представлены пять индикаторов МП КР580. Индикатор переноса CY устанавливается или сбрасывается в результате выполнения арифметических операций. Его состояние проверяется командами программы. Переполнение 8 бит при сложении устанавливает 1 в CY; в случае вычитания, когда CY установлен, это указывает, что вычитаемое больше уменьшаемого.

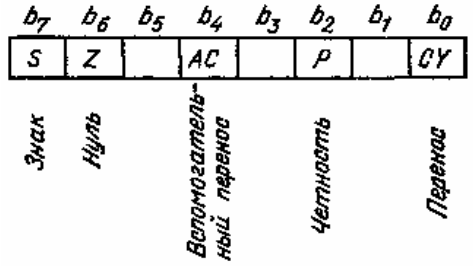


Рис. 3 Регистр признаков.

Индикатор нуля Z устанавливается, когда результатом некоторых операций является 0, в противном случае он сбрасывается.

Индикатор знака S устанавливается в зависимости от состояния наиболее значимого бита после выполнения арифметических или логических команд. Эти команды используют самый старший бит данных для того, чтобы представить знак числа, содержащегося в аккумуляторе. Установленный индикатор соответствует отрицательной величине, сброшенный — положительной.

Индикатор вспомогательного переноса AC показывает переполнение или перенос в третьем разряде аккумулятора таким же образом, как индикатор переноса показывает переполнение или перенос в седьмом разряде. Этот индикатор используется в ходе выполнения операций двоично-десятичной арифметики.

Индикатор четности P проверяет число бит единиц в аккумуляторе. Если это число четное, он показывает, что паритет четный, и тогда в индикаторе паритета устанавливается 1; если число нечетное, паритет нечетный, и индикатор сбрасывается в 0. Например, если команда ADD дает результат в аккумуляторе 0011 0011_2, в индикаторе паритета будет установлена 1, так как число единиц (4) четно. Если в аккумуляторе—1010 1110_2, индикатор P будет сброшен в 0, потому что число бит единиц (5) нечетно.

Так как все эти индикаторы содержат только единственный бит, их иногда называют не индикаторами, а битами; мы часто будем встречать выражения: бит переноса, нуля, вспомогательного переноса, паритета (в регистре состояния).

Биты 5 и 3 всегда равны 0, а 1-й единице. РП позволяет реализовать программные переходы в зависимости от результатов операции.

ОР получает информацию от внутренней ШД и может передавать ее в АЛУ, на внутреннюю ШД или в РП. Аккумулятор А может загружаться из АЛУ и внутренней ШД, а передавать данные в БА и внутреннюю ШД. В состав АЛУ входит КД. Он представляет собой комбинационную схему и осуществляет приведение двоичного результата к двоично- десятичному представлению.

Счетчик команд *PC* используется для выработки и хранения текущего 16- разрядного адреса команды. Он автоматически увеличивается на 1, 2 или 3 в зависимости от длины команды в байтах.

Стековая память реализована в ОЗУ (внешней по отношению к МП памяти). Она используется при выполнении подпрограмм и обслуживания прерываний. Указатель стека *SP* предназначен для приема и хранения 16-разрядного адреса ячейки стека, к которой было произведено последнее обращение. При каждом обращении к стеку и занесении в него слова от содержимого *SP* отнимается единица, а при извлечении слова – прибавляется 1 (т.н. "перевернутый стек").

РОН B,C,D,E,H,L – 8-разрядные регистры. Они выполняют функции сверхоперативной памяти. Их можно использовать как отдельные регистры либо как 16-разрядные указатели адреса, образуемые парами B-C, D-E, H-L. Они доступны программисту. Вспомогательные регистры W,Z предназначены для приема и временного хранения 2-го и 3-го байтов команд, передаваемых в ДК.

Эти регистры программно недоступны.

Регистр адреса РА используется для приема и хранения 16-разрядного адреса команды или операнда и выдачи его на БРА. И-Д позволяет увеличивать или уменьшать на 1 содержимое регистров в процессе межрегистровых пересылок. При этом совмещаются во времени процедура изменения адреса и выполнение операции в АЛУ. Мультиплексор МПЛ обеспечивает передачу информации между внутренней ШД о РОН. РК предназначен для приема первого байта команды, содержащего код операции, и его хранения во время выполнения команды. ДК расшифровывает код команды и вырабатывает микрокоманды в соответствии с микропрограммой ее выполнения. ДК представляет собой программируемую логическую матрицу.

Л 5. Функционирование МП.

Понятие о машинных циклах и тактах.

В ходе выполнения операций МП работает как программный автомат, который под действие тактовых импульсов Ф1 и Ф2 переходит из одного внутреннего состояния в другое. Находясь в каждом внутреннем состоянии, МП выполняет ту или иную микрооперацию. Внутреннее состояние микропроцессора называется тактом и обозначается Т.

Каждое обращение МП–ра к памяти или к элементам МПС требует нескольких тактов.

Процесс однократного обращения МП через шину данных к элементу МП системы и обработки одного байта называется циклом и называется М (машитным циклом).

Число различающихся по типу циклов не велико и для МП КР580ВМ80А ровно 10. Действия, выполняемое МП в конкретном машинном цикле определяются 8-разрядной информации состояния, которая выдается на шину данных в первом такте каждого машинного цикла. Начало каждого машинного цикла сопровождается выдачей сигнала «SYNC», на протяжении этого сигнала на шину данных и удерживается информация состояния (слово состояния).

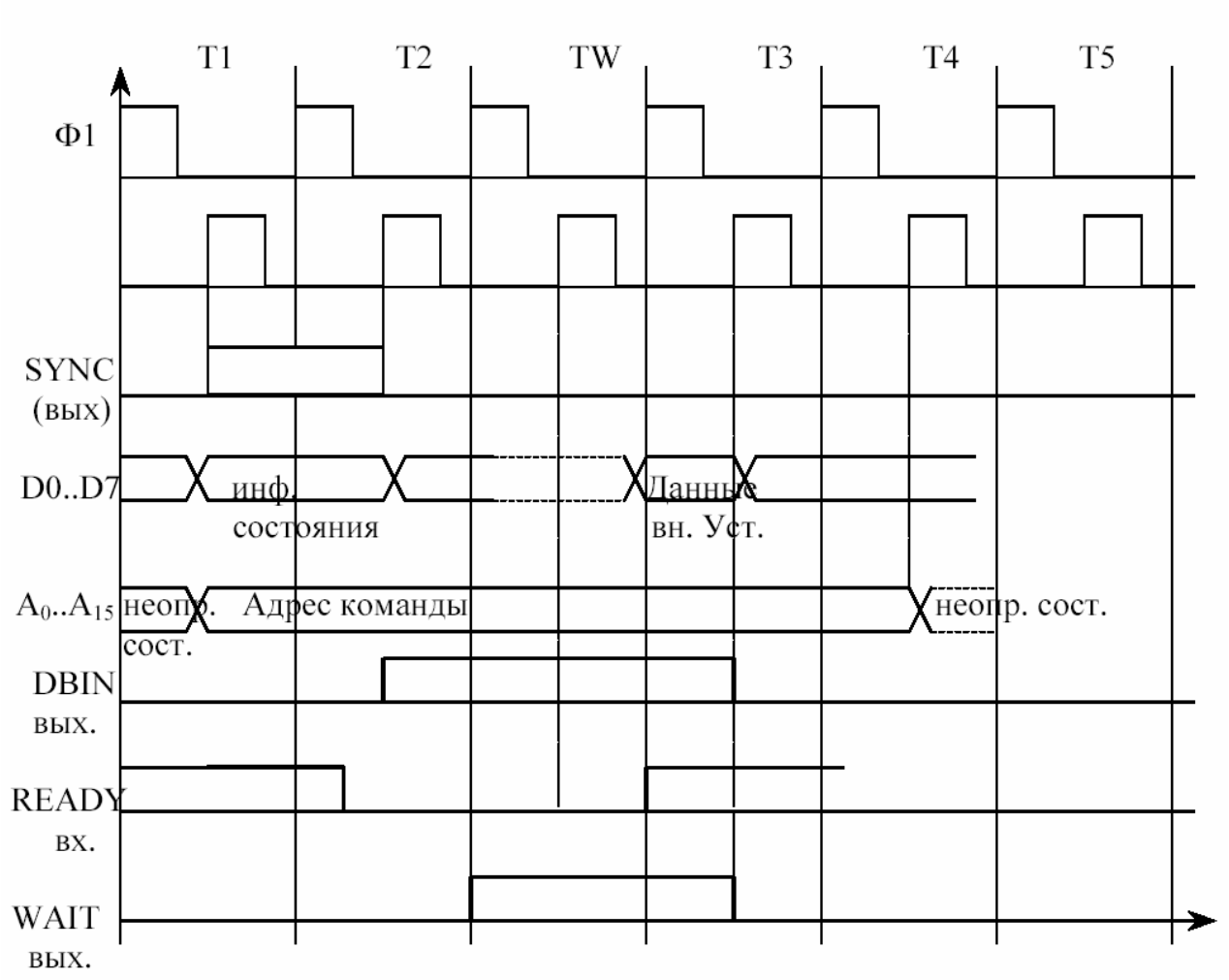
Эта информация может использоваться для выработки сигналов обращения к микросхемам памяти, устройством ввода/вывода и для организации различных режимов работы МП.

Таблица 1

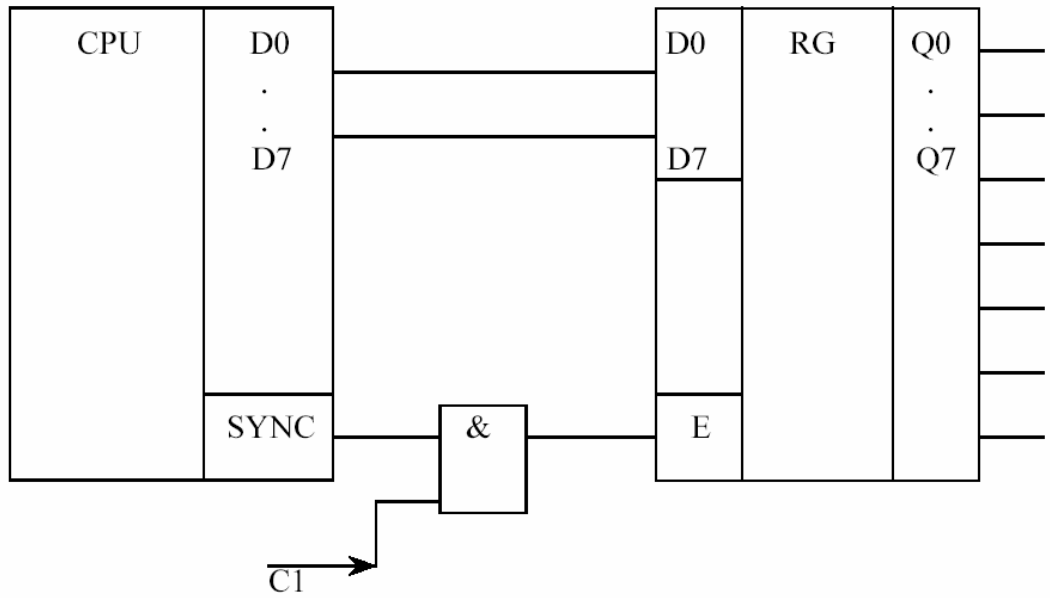
Разряд канала данных	Сигнал состояния	Цикл М1	Цикл чтения ЗУ	Цикл записи в ЗУ	Цикл чтения стека	Цикл записи в стек	Цикл ввода из порта	Цикл вывода в порт	Цикл прерывания	Цикл останова	Цикл прерывания при останове
D0	Подтверждение прерывания	0	0	0	0	0	0	0	1	0	1
D1	Запись/вывод	1	1	0	1	0	1	1	1	1	1
D2	Стек	0	0	0	1	1	0	0	0	0	0
D3	Подтверждение останова	0	0	0	0	0	0	0	0	1	1
D4	Вывод в порт	0	0	0	0	0	0	1	0	0	0
D5	M1	1	0	0	0	0	0	0	0	0	1
D6	Ввод из порта	0	0	0	0	0	1	0	0	0	0
D7	Чтение	1	1	0	1	0	0	0	0	1	0

В зависимости от сочетания сигналов состояния, выдаваемых в конкретном цикле, машинные циклы можно разделить на 10 типов:

1. Цикл M1 —прием первого байта команды в регистр команд.
2. Цикл чтения ЗУ (запоминающего устройства) по содержимому программного счетчика или содержимому одного из регистров BC, DE, HL.
3. Цикл записи в ЗУ — запись в ЗУ по содержимому одного из регистров BC, DE, HL
4. Цикл чтения стека — чтение ЗУ по содержимому указателя стека.
5. Цикл записи в стек - запись в ЗУ по содержимому указателя стека.
6. Цикл ввода - ввод информации в регистр результата (аккумулятор) из внешнего устройства.
7. Цикл вывода - вывод информации из регистра результата во внешнее устройство.
8. Цикл прерывания — прием кода команды RST или CALL из контроллера прерываний.
9. Цикл останова.
10. Цикл прерывания при останове — прием кода команды RST или CALL при выводе микропроцессора из режима «Останов» по прерыванию.



Слово внутреннего состояния сохраняется на магистрали данных только в течении одного такта. Поэтому для его хранения на время цикла в состав МПС вводится специальный регистр, информация в который записывается по сигналу «Ф2» и «SYNC»:



В такте Т1: МП выдает на адресный канал – адрес ячейки, в которой хранится команда программы, а на канал данных – информацию внутреннего состояния.

В такте Т2: МП переводит линии шины данных в режим приема информации и производит проверку готовности внешнего устройства к обмену.(контроль сигнала на вх. READY).

Если на входе READY появился сигнал неготовности, МП переходит к следующему такту TW в котором может находиться неограниченно долго, пока на входе READY не появится сигнал готовности.

В следующем такте Т3 происходит чтение данных из памяти или УВВ.

Такты Т4 и Т5 предназначены для дешифрации команды и её выполнения. Такты задержки TW могут отсутствовать.

Выполнение системных команд требует повторения нескольких машинных циклов. Например, по команде выборки данных используется три машинных цикла:

М1 – производится выборка команды, управляемый автомат по коду этой команды определяет, что она двухбайтная и запускает второй машинный цикл, изменяя содержимое ША на единицу.

М2 – чтение второго байта

М3 – чтение данных из УВВ, адрес которых содержит во втором такте команды.

При выполнении команд микропроцессор может переходить в одно из трех состояний: «ожидание», «захват» или «останов», длительность которых определяется внешними управляющими сигналами.

Сигнал высокого уровня на входе READY обеспечивает автоматическое выполнение команд программы микропроцессором с частотой тактовых сигналов. Если на выводе READY установлен сигнал низкого уровня, то микропроцессор переходит в режим «Ожидание» и формирует выходной сигнал WAIT высокого уровня.

Сигнал READY может быть использован для согласования работы микропроцессора с работой медленно действующих устройств, если длительность их цикла обращения составляет более одного периода тактовой частоты, а также для организации пошагового (по циклам) выполнения команды или покомандного выполнения программы.

При подаче на вход HOLD сигнала высокого уровня микропроцессор переходит в состояние «захват» и подтверждает переход в это состояние формированием сигнала высокого уровня на выходе HLDA.

Буферные схемы канала адреса и данных микропроцессора переключаются в высокоомное (высокоимпедансное) состояние, а выходные управляющие сигналы — в состояние низкого уровня (за исключением сигналов WR и HLDA). Микропроцессор переходит в состояние «захват» в такте ТЗ, если выполняется цикл чтения и на входе READY сигнал высокого уровня, и в такте, следующем за ТЗ, если выполняется цикл записи. Сигналы HOLD и HLDA позволяют организовать режим прямого доступа к памяти для любого внешнего устройства, формирующего сигнал HOLD.

При выполнении команды HLT микропроцессор переходит в состояние «останов» и переводит буферные схемы канала адреса и данных в высокоомное состояние. Из состояния «останов» микропроцессор выходит при наличии сигнала высокого уровня на одном из его входов:

на входе RESET - микропроцессор начинает работать с такта Т1 цикла M1;

на входе HOLD - микропроцессор переходит в состояние: «захват», а после перехода сигнала НОЫ) на низкий уровень возвращается в состояние «останов»;

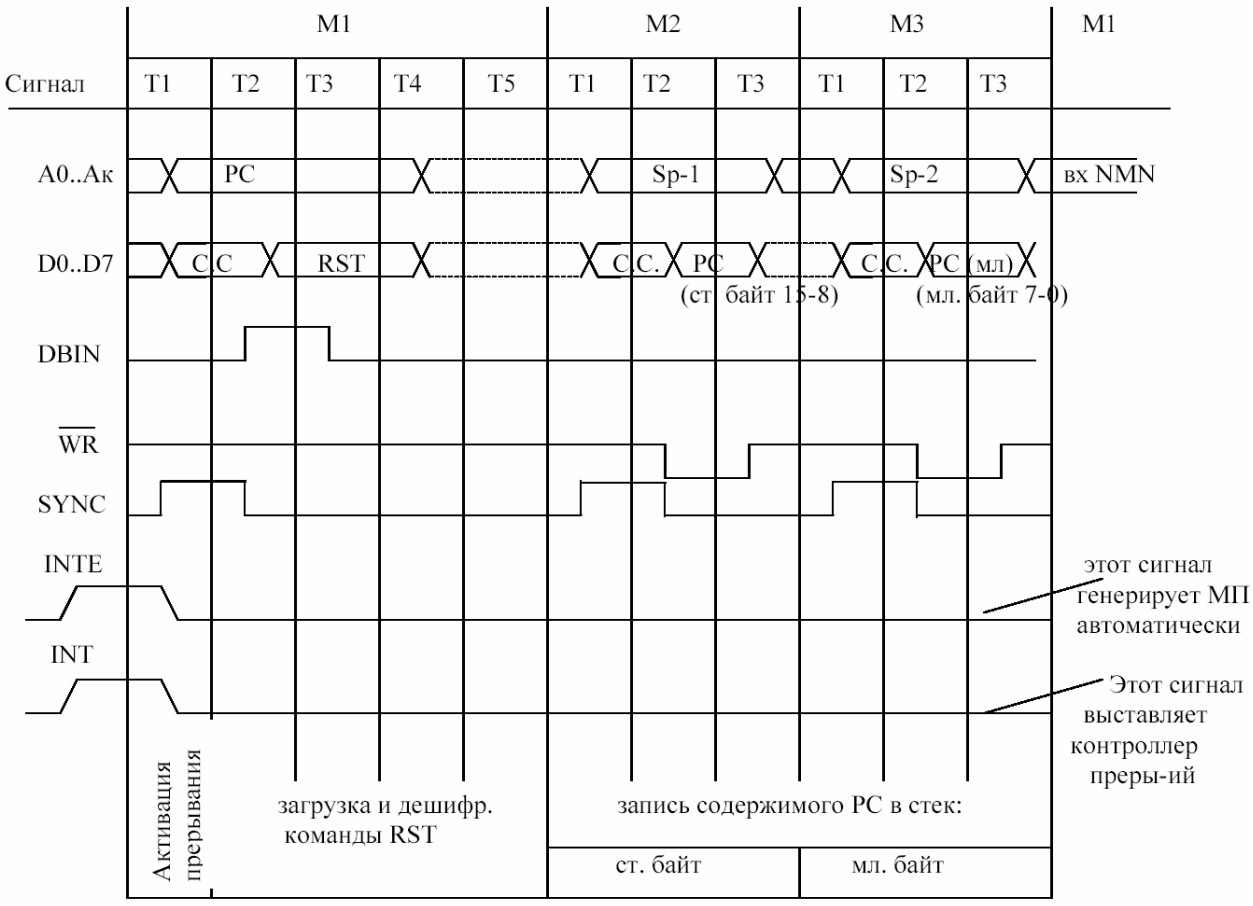
на входе INT — микропроцессор переходит к выполнению цикла прерывания при останове с такта Т1, если команде HLT предшествовала команда EI «разрешение прерывания», иначе он остается в состоянии «останов».

Сигнал высокого уровня на выводе INT позволяет прерывать выполнение текущей программы и переводить микропроцессор на выполнение подпрограммы обслуживания устройства, выдавшего запрос прерывания (например, часов). При поступлении сигнала INT микропроцессор (после окончания текущей команды) переходит с такта Т1 к выполнению машинного цикла «Прерывание» в том случае, если прерывание было разрешено ранее командой EI. При выполнении цикла «Прерывание» в такте Т1 микропроцессор выдает по шине данных сигнал состояния «Подтверждение прерывания». По окончании подпрограммы прерывания происходит возврат к прерванной программе.

Сигнал высокого уровня на входе RESET (длительность которого должна быть не менее трех периодов тактовой частоты) устанавливает микропроцессор в исходное состояние: триггеры разрешения прерывания и захвата, регистры команд, признаков и адреса команды устанавливаются в нулевое состояние. После

окончания действия сигнала RESET микропроцессор производит первое обращение за чтением команды к ячейке памяти по адресу 0000Н.

Сигналы INTE генерируется МП в конце последнего цикла каждой команды. Получив этот сигнал, контроллер прерываний выставляет сигнал «запрос прерывания», если есть устройство, требующее обследования по которому МП переходит в режим обработки прерывания в такте Т1 следующего машинного цикла:



В М1 выдается С.С., в котором присутствует информация, что сигнал INT принят и выполняется обработка прерывания. По этому сигналу контроллер обработки прерываний выставляет на шину данных код команды RST. RST записывается в рег. команд. В состав RST входит информация для задания начального адреса программы, обрабатывает прерывание. Исполнения команды RST включает в себя такие циклы занесения содержимого РС в стек.

Л. 6. ПРЕДСТАВЛЕНИЕ ИНФОРМАЦИИ В МИКРОПРОЦЕССОРНЫХ СИСТЕМАХ И МАШИННАЯ АРИФМЕТИКА

1. Краткие теоретические сведения

В ЭВМ арифметические и логические действия производятся над числами, представленными в виде специальных (машинных) кодов в принятой для данной машины системе счисления.

Под *системой счисления* понимается способ наименования и изображения чисел с помощью символов, имеющих определенные количественные значения.

Символы, применяемые для изображения чисел, называются *цифрами.*

В зависимости от способа изображения чисел с помощью цифр системы счисления делятся на позиционные и непозиционные.

Позиционной называется система счисления, в которой количественное значение каждой цифры зависит от ее места (позиции) в числе.

Примером такой системы может служить общепринятая в настоящее время арабская (десятичная) система счисления.

В *непозиционной* системе счисления цифры не меняют своего количественного значения при изменении положения в записи числа. К таким системам, например, относится римская система счисления, которая, однако, из-за сложности записи в ней многозначных чисел для вычислений не применяется.

В позиционной системе счисления числа записываются в виде последовательности цифр

$$A = a_{m-1}a_{m-2}\ldots a_k\ldots a_1a_0$$

Позиции, пронумерованные индексами k (в данном случае в пределах ($0 < k <, m$ -1), называются разрядами числа. Индекс m соответствует количеству разрядов.

Каждая цифра a_k в записанной последовательности может принимать одно из некоторого количества N возможных значений, т.е. $N\text{-}1 \geq a_k \geq 0$.

Количество *(N)* различных цифр, используемых для изображения чисел в позиционной системе счисления, называется *основанием системы счисления.*

Поскольку цифра a_k соответствует количеству единиц k-го разряда, содержащихся в числе, то основание *(N)* позиционной системы счисления указывает, во сколько раз единица *(k+1)* -го разряда больше единицы младшего k-го разряда. Следовательно, записанную выше последовательность цифр, соответствующих целому числу, можно представить в виде:

$A_m = a_{m-1}N^{m-1} + a_{m-2}N^{m-2} + \ldots + a_1N^{m-2} + a_0N^0$.

В общем случае выражение для любого числа, состоящего из целой и дробной частей (неправильная дробь), будет представлять собой ряд:

$$A_{(N)}=\pm[\, a_{m-1}N^{m-1}+a_{m-2}N^{m-2}+\ldots +a_1N^1+a_0N^0+ a_{-1}N^{-1}+\ldots +a_{-l}N^{-l}\,], \qquad (1)$$

где m и l — число разрядов соответственно целой и дробной частей числа, N^i- вес i-го разряда. Следует заметить, что в записи вида (1) основание N может быть разным для различных разрядов, например, запись угловых величин в градусах, минутах и секундах или запись величин, характеризующих время. Такие системы называются *неоднородными,* в отличии от *однородных* систем с равными основаниями для всех разрядов. В настоящее время в вычислительных машинах используются только однородные системы счисления.

Основание N позиционной системы счисления определяет и ее название. Так, например, общепринятая десятичная система счисления имеет основание N=10. Любое число в этой системе записывается с помощью различных цифр:

a_k=0,1,2,3,4,5,6,7,8,9

В качестве основания можно выбирать любое другое число. Такими числами, а следовательно и основаниями, могу быть: два, три, четыре, пять и т.д.

Исторически так сложилось, что именно десятичная система оказалась общепринятой и широко применяемой при ручном счете и в электромеханических вычислительных устройствах, Однако с точки зрения простоты конструктивного выполнения отдельных устройств для ЭВМ оказываются удобными также другие системы счисления с основаниями два -двоичная, восемь - восьмеричная и шестнадцать — шестнадцатиричная.

2. Двоичная система счисления

Двоичная система счисления проще десятичной. В ней используются только два символа, что хорошо согласуется с техническими характеристиками цифровых схем, имеющих лишь два устойчивых состояния. Как правило, в качестве символов в двоичной системе служат 0 и 1. Иногда используются другие обозначения: термины: «включено - выключено», «низкий - высокий уровень», «знак-пробел». Однако в любом случае речь идет о двух возможных состояниях.

При сравнении записи десятичных и двоичных чисел выясняется, что последние занимают значительно больше позиций, поскольку для их представления используются лишь два символа. В двоичной системе счисления, так же как и в десятичной, каждой позиции (разряду) присвоен определенный вес. Но если в десятичной системе вес равен числу 10 в некоторой степени, то в двоичной системе вместо числа 10 используется число 2. Веса первых 13 позиций (разрядов) цифр двоичного числа имеют следующие значения:

4096	2048	1024	512	256	128	64	32	16	8	4	2	1
2^{12}	2^{11}	2^{10}	2^9	2^8	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2°

В двоичной системе счисления даже сравнительно небольшие числа занимают много позиций. Например:

$101101_2 = 1x2^5+0x2^4+1x2^3+1x2^2+0x2^1+1\ x2^\circ=45_{10}$,

т. е. двоичное число 101101 имеет ту же величину, что и десятичное число 45. Для удобства идентификации записи двоичных и десятичных чисел в виде нижнего индекса числа записывают 2 и 10 соответственно. Так, например, 101_2 — двоичное число, а 101_{10}-десятичное, причем они выражают разные величины.

Как и в десятичной системе, в двоичной системе для отделения дробной части от целой используется точка *(двоичная точка).* Каждая позиция справа от этой точки имеет свой вес-вес разряда дробной части числа.

Значение веса в этом случае равно основанию двоичной системы, возведенному в отрицательную степень. Такие веса - это дроби 1/2, 1/4, 1/8, 1/16, 1/32 и т.д., которые могут быть записаны как 2^{-1} , 2^{-2} ,2^{-3}, 2^{-4} , 2^{-5} и т.п. Выразим эти веса через десятичные дроби:

$2^{-1} = 0.5$ $2^{-2} = 0.25$ $2^{-3} = 0.125$ $2^{-4}=0.0625$ $2^{-5}=0.03125$ $2^{-6} = 0.015625$ и т.д.

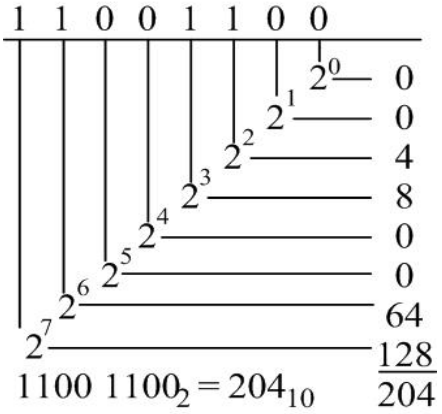
При записи числа в десятичной системе каждая позиция занята *десятичной цифрой,* Аналогично при записи двоичного числа каждая позиция занята *двоичной цифрой,* называемой *битом.* Говоря о двоичных числах, часто пользуются понятиями *наименьший значащий бит (самый младший двоичный разряд)* и *наибольший значащий бит (самый старший двоичный разряд)* по аналогии с наибольшей и наименьшей значащими цифрами десятичного числа. Наименьший значащий бит имеет наименьший вес, а наибольший значащий бит соответственно наибольший. Обычно двоичное число записывается так, что наибольший значащий бит является крайним слева.

3. Преобразование двоичных чисел в десятичные

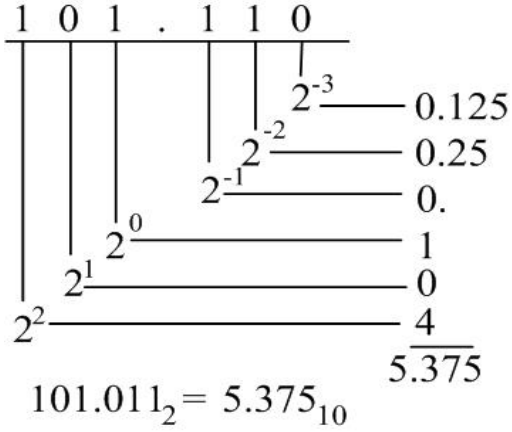
При работе с микро-ЭВМ часто бывает необходимо заменить двоичные числа их десятичными эквивалентами.

Процедура преобразования двоичного числа в десятичное проста: необходимо сложить десятичные веса всех разрядов двоичного числа, в которых содержатся единицы. Продемонстрируем это на следующих примерах.

Пример 1. Преобразование целого двоичного числа 11001100 в десятичное:



Пример 2. Преобразование вещественного двоичного числа 101.011 в десятичное:



4. Преобразование десятичных чисел в двоичные

Иногда бывает удобно для пользователя, чтобы микро-ЭВМ имела дело с десятичными числами. Для этого необходимо ввести такие числа в машину, преобразовать их в двоичные эквиваленты, побудить микропроцессор обработать двоичные числа, а затем полученные двоичные числа - результат преобразовать в десятичные числа. Следовательно, нужно владеть приемами преобразования десятичных чисел в двоичные. Во-первых, необходимо уметь выполнять требуемые операции вручную, пока осуществляется программирование работы микропроцессора. Это позволит нам ввести числовые константы в программу вычислений и провести арифметические расчеты. Во-вторых, мы должны быть способны использовать микропроцессор для преобразования десятичных чисел в двоичные. Только тогда пользователь может вводить десятичные числа, которые микропроцессор будет преобразовывать в двоичные. Освоив технику выполнения первой процедуры, легко разобраться и во второй, реализуемой программными средствами.

Процедура преобразования целых десятичных чисел в двоичные - это частный случай процедуры перевода чисел из одной системы счисления в другую. Предположим, что необходимо преобразовать десятичное число 10 в двоичное. Для этого проделаем следующее.

1. Разделим подлежащее преобразованию число на основание системы счисления, в которой число должно быть представлено. В данном случае 10 следует поделить на 2. При делении на 2 остаток может быть равен 1 или 0. Значение остатка присваивается младшему значащему разряду (МЗР) искомого числа. Для рассматриваемого примера частное равно 5, а остаток - нулю, т.е. 1 - й разряд равен нулю.

2. Результат деления па первом шаге необходимо разделить еще раз на 2. Остаток (0 или 1) используется в качестве значения следующего по значимости разряда. В данном случае частное от деления 5 на 2 равно 2, а остаток, т. е. значение 2-го разряда, равно 1.

3. Результат деления на предыдущем шаге необходимо разделить па 2, а значение остатка присвоить очередному разряду. В данном случае частное равно 1, а остаток равен нулю, т.е. 3-й разряд равен нулю.

4. Шаги описанной процедуры повторяются до тех пор, пока частное, полученное в результате очередной операции деления, не станет равным нулю. Тогда остаток от последнего деления используется в качестве значения старшего значащего разряда (СЗР). В данном случае частное от деления 1 па 2 составляет нуль, а остаток равен 1, поэтому значение 4-го разряда равно 1.

Итак, получено целое двоичное число 1010. Рассмотрим еще два примера преобразования десятичных чисел в двоичные.

Пример 1. Преобразование десятичного числа 57_{10} в двоичное число:

Шаг	Деление	Частное	Остаток
1	57/2	28	1 (МЗР)
2	28/2	14	0
3	14/2	7	0
4	7/2	3	1
5	3/2	1	1
6	1/2	0	1 (СЗР)

Результат: $57_{10} = 1000\ 0110_2$,

Пример 2. Преобразование десятичного числа 134_{10} в двоичное число:

Шаг	Деление	Частное	Остаток
1	134/2	67	0 (МЗР)
2	67/2	33	1
3	33/2	16	1
4	16/2	8	0
5	8/2	4	0
6	4/2	2	0
7	2/2	1	0
8	1/2	0	1 (СЭР)

Результат $-134_{10} = 1000\ 0110_2$

Изложенная процедура применима к преобразованию целых (или целой части) десятичных чисел в двоичные. Для дробных чисел (или дробных частей вещественных чисел) требуется отдельная, хотя и похожая, процедура. Если преобразовать выполнено отдельно для целой и дробной частей числа, то результат получают путем записи двоичных эквивалентов этих частей соответственно слева и справа от двоичной точки.

Процедуру преобразования десятичной дроби в двоичную рассмотрим на примере преобразования числа 0.375.

1. Преобразование осуществляется умножением дроби на основание системы счисления, в которой дробь должна быть представлена. В данном случае умножаем на 2: 2 x 0.375 = 0.75.

2. Если результат умножения меньше 1, то старшему значащему разряду присваивается значение 0; если больше 1, то присваивается 1. Поскольку 0.75 < 1, то **СЗР** = 0.

3. Результат предыдущей операции умножения опять умножается на 2. Заметим, что если бы результат предыдущей операции умножения был больше 1, то в данной операции умножения участвовала лишь его дробная часть. В данном случае 2 x 0.75 = 1.5.

4. Если полученный результат меньше 1, то следующему по значимости (ближайшему справа) разряду присваивается значение 0; если равен или больше 1, то присваивается 1. В рассматриваемом примере 1.5 > 1, поэтому значение разряда 2 равно 1.

5 Шаги описанной процедуры повторяются до тех пор, пока либо результат умножения не будет точно равен 1, либо не будет достигнута требуемая точность. В данном случае после выполнения очередного шага результат равен (2 x 0.5=1.0). Поэтому очередному разряду, являющемуся младшим значащим разрядом, присваивается значение 1.

Следовательно, получена двоичная дробь 0.011.

Следует отметить, что не всегда путем повторения операций умножения можно достичь результата умножения, точно равного 1. В таком случае процесс повторения останавливают по достижении необходимой точности, а целую часть результата последней операции умножения используют в качестве значения младшего значащего разряда.

Рассмотрим еще два примера преобразования десятичных дробей в двоичные.

Пример 1. Преобразование десятичного числа 0.34375_{10} в двоичное:

Умножение	Результат в целочисленной форме
2 x 0.34375 = 0.6875	0 (СЗР)
2x0.6875= 1.375	1
2 x 0.375 = 0.75	0
2 × 0,75 = 1,5	1
2 × 0,5 = 1,0	1
2 × 0 = 0	0 (МЗР)

Результат: $0{,}01011_2$.

Пример 2. Преобразование десятичного числа $0{,}3_{10}$ в двоичное:

Умножение	Результат в целочисленной форме
$2 \times 0{,}3 = 0{,}6$	0
$2 \times 0{,}6 = 1{,}2$	1
$2 \times 0{,}2 = 0{,}4$	0
$2 \times 0{,}4 = 0{,}8$	0
$2 \times 0{,}8 = 1{,}6$	1
$2 \times 0{,}6 = 1{,}2$	1
$2 \times 0{,}2 = 0{,}4$	0
$2 \times 0{,}4 = 0{,}8$	0
$2 \times 0{,}8 = 1{,}6$	1
$2 \times 0{,}6 = 1{,}2$	1
$2 \times 0{,}2 = 0{,}4$	0

Процедура преобразование в примере 2 носит характер бесконечного построения группы одинаковых операций и результатов. Поэтому ограничимся восемью разрядами. Тогда получим $0{,}3_{10} = 0{,}01001100_2$.

6. Шестнадцатеричная система счисления

В 16-и ричной системе счисления используются следующие 16 символов: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E и F. Практическое использование шестнадцатеричной системы объясняется тем, что число 16 есть число 2 в четвёртой степени. Иначе говоря, шестнадцатеричную цифру можно использовать как средство сокращенной записи 4-разрядного двоичного числа (тетрады). В табл.2 приводятся примеры шестнадцатеричных чисел и их двоичных и десятичных эквивалентов.

Таблица 2. Шестнадцатеричные числа и их двоичные и десятичные эквиваленты

Шестнадцатеричное число	Двоичное число	Десятичное число
0	0	0
1	1	1
2	10	2
3	11	3
4	100	4
5	101	5
6	110	6
7	111	7
8	1000	8

9	1001	9
A	1010	10
B	1011	11
C	1100	12
D	1101	13
E	1110	14
F	1111	15

Процедура преобразования двоичного числа в шестнадцатеричное довольно проста. Биты, начиная с младшего значащего бита (расположенного рядом с двоичной точкой), объединяются в группы по четыре. Каждой группе подбирается соответствующий шестнадцатеричный символ.

Рассмотрим два примера. Чтобы преобразовать двоичное число 10101011111101_2, необходимо добавить слева два незначащих нуля с целью формирования битов в группы по четыре: 0010 1010 1111 1101. Заменив каждую группу битов соответствующим шестнадцатеричным символом, получим число $2AFD_{16}$. В результате преобразования двоичного числа 11000 111_2 в шестнадцатеричное получим число $C7_{16}$. Сравним исходные данные и результаты обоих примеров, нетрудно заметить, что шестнадцатеричная форма записи числа много проще и легче воспринимается, чем двоичная. К аналогичному выводу проводит сопоставление восьмеричной записи числа с двоичной. Действительно, для рассматриваемых примеров имеем $10101011111101_2 = 25375_8 = 2AFD_{16}$ и $11000111_2 = 307_8 = C7_{16}$.

Следует помнить, что шестнадцатеричные и восьмеричные числа – это только способ представления двоичных чисел, которыми фактически оперирует микропроцессор. При этом шестнадцатеричная запись числа предпочтительнее восьмеричной, поскольку микропроцессоры манипулируют словами длиной 4, 8, 16 или 32 бит, т.е. словами, длина которых кратна 4. Ведь для записи двоичных чисел такой длины в восьмеричной системе счисления приходится добавлять незначащие нули, чтобы длина стала кратной трём.

В подобных добавлениях нет необходимости при использовании шестнадцатеричного представления слов микропроцессора. Например, 8-битовое число можно разделить на два 4-битовых, каждое из которых представляется одним шестнадцатеричным символом.

Простота соотношений между шестнадцатеричной и двоичной формами записи чисел – причина значительно большей распространённости шестнадцатеричной системы счисления по сравнению с восьмеричной. Последняя нашла широкое применение в семействе ЭВМ PDP-8 фирмы DEC (Digital Equipment Corporation), поскольку эти машины оперируют 12-битовыми словами, которые легко представляются в восьмеричной системе счисления.

Преобразование двоичных дробей в шестнадцатеричные осуществляется по правилам, аналогичным для преобразования этих дробей в восьмеричные. Для этой цели биты дробной части, начиная со старшего

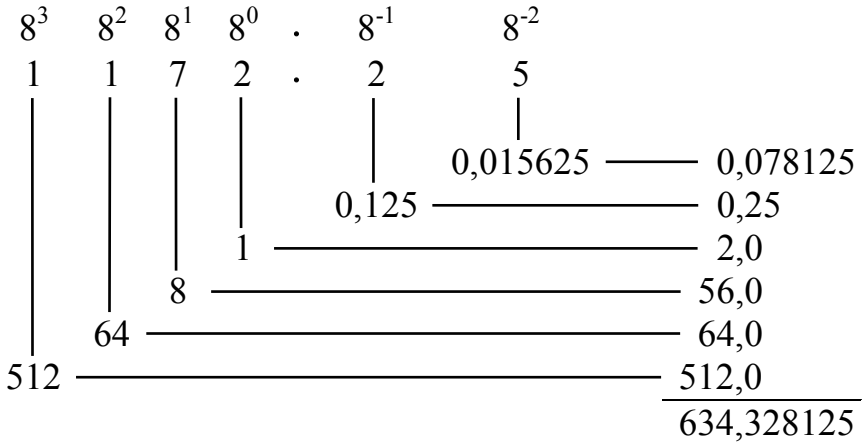
значащего бита (расположенного справа от двоичной точки), группируются по четыре. Добавление незначащих нулей осуществляется по мере необходимости. Например, для преобразование двоичной дроби $0{,}0101101_2$ в шестнадцатеричную биты группируют по четыре: 0,0101 1010. Затем, каждую группу заменяют шестнадцатеричным символом, получая в результате $0{,}5A_{16}$. Подобным образом, преобразуя число 1101.0111_2, составляют группы 1101.0111, и после их замены шестнадцатеричными символами формируется число $D.7_1$.

7. Восьмеричная, десятичная и шестеричная системы счисления. Преобразования из одной системы в другую.

При работе с микропроцессорами возникает необходимость преобразования десятичных чисел в двоичные и обратного преобразования. Часто оказывается желательным представление двоичных чисел в восьмеричной или шестнадцатеричной форме. Как следствие этого, возникает и обратная задача: преобразование восьмеричных и шестнадцатеричных чисел в двоичные. Может также потребоваться преобразовать восьмеричные и шестнадцатеричные числа в десятичные.

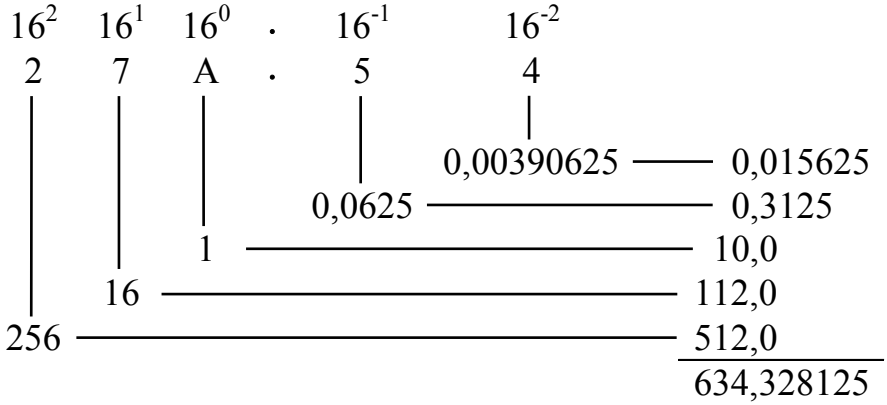
Преобразование восьмеричных или шестнадцатеричных чисел в десятичные схоже с преобразованием этих чисел в их двоичные эквиваленты. Каждой позиции числа присваивается определённый вес. Затем значение веса позиции умножается на цифру, занимающую эту позицию. Результаты операции умножения, выполненных для всех позиций числа, суммируются. Следующие ниже примеры демонстрируют преобразование чисел из восьмеричной и шестнадцатеричной систем в десятичную.

Пример. Представить в десятичной системе восьмеричное число 1172.25_8:



Результат: $1172.25_8 = 634.328125_{10}$.

Пример. Представить в десятичной системе шестнадцатеричное число $27A.54_{16}$:



Результат: $27A.54_{16} = 634.328125_{10}$.

Описанная процедура в действительности весьма проста. Что же касается дробной части, то на примере двоичных дробей можно было убедиться, что число может оказаться весьма длинным. Поэтому часто приходиться прибегать к округлению.

Рассмотрим теперь преобразование десятичных чисел в восьмеричные и шестнадцатеричные. Можно воспользоваться процедурой, подобной той, с помощью которой осуществлялось преобразование десятичных чисел в двоичные, поскольку она в определённом смысле универсальна. Для преобразования десятичного числа в восьмеричное или шестнадцатеричное вместо деления (умножения для дробей) исходного числа на 2 необходимо выполнить деление (умножение) на 8 или 16. Остатки (целые части произведений) используются для формирования результата.

Ниже приводятся два примера преобразования десятичного числа в восьмеричное и шестнадцатеричное. Процедуры обработки целой и дробной частей десятичного числа отличаются друг от друга и выполняются порознь. Результаты объединятся при формировании восьмеричного или шестнадцатеричного эквивалентов.

Пример 1. Преобразовать десятичное число 634.328125_{10} в восьмеричное:

Преобразование целой части				Преобразование дробной части	
Шаг	Деление	Частное	Остаток	Умножение	Произведение (целая часть)
1	634/8	79	2 (МЗР)	8 × 0,328125 = 2,625	2 (СЭР)
2	79/8	9	7	8 × 0,625 = 5,0	5 (МЗР)
3	9/8	1	1		
4	1/8	0	1 (СЭР)		

Результат: $634_{10} = 1172_8$ $0{,}328125_{10} = 0{,}25_8$

Общий результат: $634{,}328125_{10} = 1172{,}25_8$

Пример 2. . Преобразовать десятичное число 634.328125_{10} в шестнадцатеричное:

Преобразование целой части				Преобразование дробной части	
Шаг	Деление	Частное	Остаток	Умножение	Произведение

					(целая часть)
1	634/16	39	$10_{10} = A_{16}$ (МЗР)	$16 \times 0{,}328125 = 5{,}25$	5 (СЗР)
2	39/16	2	7	$16 \times 0{,}25 = 4{,}0$	4 (МЗР)
3	2/16	0	2 (СЗР)		

Результат: $634_{10} = 27A_{16}$ $0{,}328125_{10} = 0{,}54_{16}$

Общий результат: $634{,}328125_{10} = 27A,\ 54_{16}$

Отметим, что преобразование десятичных чисел в их восьмеричные или шестнадцатеричные эквиваленты выполняется с целью получения более компактной формы представления исходных десятичных данных. Преобразуя числа $1172{,}25_8$ и $27A,\ 54_{16}$ в их двоичные эквиваленты, можно убедиться, что одно и то же число:

1	1	7	2,	2	5_8
001	001	111	010,	010	101
2	7	A,	5	4_{16}	
0010	0111	1010,	0101	0100	

Безусловно, существуют также процедуры преобразования восьмеричных чисел в шестнадцатеричные, и наоборот. Однако необходимость в их использовании возникает крайне редко. Но если такая проблема появилась, одним из способов её решения является представления исходного числа в двоичной форме, а затем преобразование полученного двоичного числа в требуемую систему счисления.

8. Двоичное сложение и вычитание

Выполняются как и в десятичной системе счисления.
(привести примеры)

9. Двоичные числа в дополнительном коде

Возникает вопрос: можно ли записать отрицательные двоичные числа, и если можно, то каким образом?

Существует несколько способов представления отрицательных двоичных чисел. Большинство из них не соответствует возможностям «двоичной электроники», т.е. аппаратной основе АЛУ. Однако анализ этих способов и требований, связанных с работой электронных схем, привёл к разработке приемлемых решений. Один из способов – *представление числа посредством величины и знака*, причём бит знака занимает самый старший разряд поля представления двоичного числа. Если число положительное, бит знака равен 0, если оно отрицательное, то этот бит равен 1. Рассмотрим, например, десятичное число 28. Его 7-разрядный двоичный эквивалент имеет вид 0011100. Если десятичное число положительное (+28), то к указанному

двоичному эквиваленту следует приписать слева 0 (бит положительного знака), а именно 0001 1100. Если же десятичное число отрицательное (-28), то требуется добавить 1 (бит отрицательного знака): 1001 1100.

Бит знака, равный нулю для положительных чисел и единице для отрицательных, используется и при записи двоичных чисел в *обратном коде* (в виде так называемого дополнения до 1). Обратный код двоичного отрицательного числа формируется заменой всех нулей числа на единицы, а всех единиц – на нули. Это же выполняется и для самого старшего разряда поля представления числа. Поскольку этот разряд «не занят» битами величины числа, то в исходном состоянии там нуль, а по завершении формирования обратного кода отрицательного числа - единица, выполняющая роль кода знака.

Правило формирования обратного кода простое, однако работа с обратными кодами вызывает ряд затруднений. Так, нулевой результат может быть представлен комбинацией или двоичных нулей, или двоичных единиц. Приведём пример записи отрицательного числа в обратном коде: десятичному числу -28 соответствует двоичная запись 1110 0011. (Сравните с двоичным числом, являющимся эквивалентом десятичного числа +28.)

В микро-ЭВМ широко используется представление отрицательных чисел в *дополнительном коде* (в виде так называемого дополнения до 2). При таком представлении исчезает двусмысленность представления нулевого результата, присущая записи в обратном коде. Формирование дополнительного кода, или сокращенно дополнения, состоит из двух операций: получения обратного кода и добавления единицы. Как следует из табл.3, это позволяет, например, посредством 8 бит представить в двоичной форме десятичные числа от -128 до + 127, включая 0. В таблице показаны два типичных для микропроцессоров способа использования двоичных кодов: как двоичных чисел со знаком, так и без знака. Левый столбец содержит двоичные числа (двоичные коды) от 0000 0000 до 1111 1111, правый столбец - их десятичные эквиваленты от 0 до 255, полученные в предположении, что рассматриваются числа без знака.

Таблица 3. **Десятичные эквиваленты двоичных чисел**

	Десятичный Эквивалент	
8- разрядное двоичное число	двоичного числа со знаком (отрицательное число в дополнительном коде)	двоичного числа без знака
00000000	+0	0
00000001	+1	1
00000010	+2	2
00000011	+3	3
…	…	…
01111100	+124	124
01111101	+125	125

01111110	+126	126
01111111	+127	127
10000000	-128	128
10000001	-127	129
10000010	-126	130
10000011	-125	131
…	…	…
11111100	-4	252
11111101	-3	253
11111110	-2	254
11111111	-1	255

В центральном столбце находятся десятичные эквиваленты двоичных чисел левого столбца, полученные в предположении, что отрицательные числа записывались в дополнительном коде. Здесь положительным двоичным числам (от 0000 0000 до 0111 1111) соответствуют десятичные числа от 0 до + 127, а содержащим 1 в восьмом разряде отрицательным двоичным числам (от 1000 0000 до 1111 1111)- десятичные от -128 до -1.

Используя восемь двоичных разрядов и представляя отрицательные числа в дополнительном коде, можно записать 256 различных чисел: 127 положительных, пуль и 128 отрицательных. Упомянутую выше процедуру формирования дополнения - представление отрицательного числа в обратном коде и добавление единицы — продемонстрируем на следующем примере:

Число 4_{10} в двоичной форме	0000100
Обратный код числа 4_{10}	1111011
Добавляемая к обратному коду 1_2	1
Число 4_{10} в дополнительном коде	1111100

Сопоставьте полученный результат с соответствующим кодом в табл. 3.

Для представления двоичного числа в дополнительном коде можно пользоваться другим способом, отличным от описанного выше и более коротким по числу операций. В поисках первого бита, равного единице, просматривают справа налево разряды числа, начиная с наименьшего по значимости. До тех пор пока встречаются нули, их копируют в одноименные разряды результата. Первая встретившаяся единица также копируется в соответствующий разряд результата, но каждый последующий бит исходного числа заменяется на обратный.

Арифметические операции над двоичными числами без знака ничем не отличаются от подобных операций над двоичными числами со знаком, отрицательные из которых представлены своими дополнениями. Это существенно упрощает аппаратную реализацию подобных операций в микропроцессоре. Однако следует не упускать из виду, с какими числами вы имеете дело в данный момент; без знака или со знаком. Например, при сложении двух чисел без знака результат — число без знака в виде некоторой

последовательности битов, которую можно интерпретировать и как отрицательное число в дополнительном коде.

В общем случае при сложении или вычитании чисел со знаком результат есть число со знаком; если при этом бит старшего разряда равен единице, то результат - отрицательное число в дополнительном коде. Если требуется определить абсолютное значение (величину) результата, последний необходимо представить в обратном коде, а затем прибавить единицу.

Т.о., вычитания выполняется следующим образом: определяется дополнительный код вычитаемого и производится сложение этого кода с уменьшаемым. Если разность — число положительное (бит старшего разряда равен 0), то бит переноса необходимо отбросить; полученная последовательность битов и есть двоичный код результата. Если разность - число отрицательное (бит старшего разряда равен 1), то она представлена в дополнительном коде. Выше указывалось, что для определения абсолютной величины отрицательного числа, представленного в таком виде, необходимо применить к нему операцию вычисления дополнительного кода.

Проиллюстрируем операции вычитания следующими примерами.

Пример 1. Вычислить разность чисел 58 - 23:

а) Определение дополнительного кода числа 23

00010111	Число 23_{10}
1110 1000	Обратный код 23_{10}
0000 0001	Единица, добавляемая к обратному коду
1110 1001	Дополнительный код числа 23_{10}

б) Вычисление разности

Десятичная арифметика	Двоичная арифметика	
58	0011 1010	Число 58_{10}
—	+	
23	1110 1001	Дополнительный код числа 23_{10}
35	**1** 0010 0011	Разность 35_{10}

Единица переноса, отбрасываемая в случае положительного результата.

Пример 2. Вычислить разность чисел 26-34:

а) Определение дополнительного кода числа 34

0010 0010	Число 34_{10}
1101 1101	Обратный код 34_{10}
0000 0001	Единица, добавляемая к обратному коду
1101 1110	Дополнительный код числа 34_{10}

Десятичная арифметика	Двоичная арифметика	
26	0001 1010	Число 26_{10}

—	+	
34	1101 1110	Дополнительный код числа 34_{10}
-08	1111 1000	Разность в форме дополнения (поскольку в старшем разряде 1

в) Определение абсолютного значения разности

1111 1000	Дополнительный код разности
0000 0111	Обратный код
0000 0001	Единица, добавляемая к обратному коду
0000 1000	Абсолютное значение разности (8_{10})

10. Двоично-десятичные числа

С целью удобства преобразования чистые двоичные числа представляются десятичными либо шестнадцатеричными. Однако двоично-десятичное преобразование — операция не простая. В калькуляторах, магистралях и числовых приборах, когда на доступных пользователю выходах и входах широко распространены десятичные числа, для их представления используют специальный двоично-десятичный код (ДДК). В табл. 4 приведено несколько десятичных чисел и соответствующих им двоично-десятичных эквивалентов (система 8421). Этим определяются веса позиций каждого из четырех бит ДДК (используют другие ДДК, например 5421 и плюс 3).

Таблица 4. **Двоично-десятичный код 8421**

Десятичные числа	Двоично-десятичные числа 8421
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001

Запишем десятичное число 3691 в ДДК 8421. Каждая десятичная цифра преобразуется прямо в свой двоично-десятичный эквивалент из 4 бит, и преобразования дают 3691_{10} = 0011 0110 1001 $0001_{ДДК}$:

Десятичное число	3	6	9	1
Двоично-десятичное число	0011	0110	1001	0001

Преобразуем теперь двоично-десятичное число 1000 0000 0111 0010 в его десятичный эквивалент. Каждая группа из 4 бит прямо преобразуется в ее десятичный эквивалент, и тогда получаем 1000 0000 0111 0010ддк = 8072_{10}:

Двоично-десятичное число	1000	0000	0111	0010
Десятичное число	8	0	7	2

Микропроцессоры складывают чистые двоичные числа, но они обладают, однако, командами для преобразования результата своих сложений в двоично-десятичную запись. Полученные двоично-десятичные числа легко затем представить в десятичной записи, используя выше описанные простые процедуры.

11. Двоичное умножение

Двоичное и десятичное умножение, так же, как и двоичное и десятичное сложение или вычитание, во многом похожи. Умножение – это быстрый способ сложения нескольких одинаковых чисел. Например, умножение 7 на 5 сводится к сложению пяти одинаковых чисел, каждое из которых равно 7.

При умножении одного числа на другое одно из чисел называется множимым, другое – множителем. Умножение выполняется поразрядно. Часто возникает необходимость переноса в следующий по старшинству разряд. Перемножая десятичные числа, мы обычно «решаем в уме» возникающие при этом проблемы переноса. По завершении умножения множимого на значение младшего разряда множителя получается первое частичное произведение. В результате умножения множимого на значение следующего по старшинству разряда множителя формируется второе частичное произведение. Подобная процедура повторяется с целью получения всех необходимых частичных произведений. Поскольку каждое очередное частичное произведение – результат перемножения множимого и разряда множителя, значимость которого в 10 раз больше значимости разряда, использованного в предыдущей операции умножения, то все цифры полученного произведения сдвигаются влево на одну десятичную позицию (разряд). Для получения результирующего произведения смещенные относительно друг друга частичные произведения складываются. Возникающие при сложении переносы должны быть учтены при формировании окончательного результата.

Теперь перейдем к рассмотрению двоичного умножения. Таблица двоичного умножения чрезвычайно проста: в умножении участвуют цифры, принимающие только два значения – 0 или 1, в результате умножения перенос не возникает никогда.

Вычислим произведения 8-разрядных двоичных эквивалентов десятичных чисел 17 и 12:

```
00010001  Множимое 17(10)
00001100  Множитель 12(10)
--------
00000000  Первое частичное произведение
 00000000  Второе частичное произведение
  00010001  Третье частичное произведение
   00010001  Четвертое частичное произведение
    00000000  Пятое частичное произведение
     00000000  Шестое частичное произведение
      00000000  Седьмое частичное произведение
       00000000  Восьмое частичное произведение
000000000000000  Перенос
----------------
0000000011001100  Результат (204(10))
```

Прежде всего отметим, что получено восемь частичных произведений, поскольку множитель состоит из 8 разрядов. Первые два частичных произведения включают только нули, так как множители - значения первого и второго разрядов двоичного эквивалента числа 12_{10}-равны 0. Третье частичное произведение - копия множимого. Разница между ними заключается лишь в том, что копия сдвинута относительно множимого на два двоичных разряда влево, поскольку для получения этого частичного произведения в качестве множителя используется значение третьего разряда. Четвертое частичное произведение также является копией множимого, смещенной относительно последнего на три двоичных разряда влево. Частичное произведение с пятого по восьмое состоит только из нулей, так как соответствующие множители, участвующие в формировании произведения, – двоичные нули. Сложение всех частичных произведений в данном примере не сопровождается переносом, однако возникновение последнего, вообще говоря, не исключено. Результат занимает 16 позиций. Значение восьми старших разрядов равно нулю, поэтому запись полученного произведения можно ограничить восемью младшими значащими разрядами, которых достаточно для представления числа меньшего по величине, чем 255.

Создан простой способ выполнения двоичного умножения, получивший название умножения путем сдвига и сложения. Перечислим основные правила этого способа.

1. Формирование первого частичного произведения. Если значение младшего значащего разряда множителя равно 0, то и результат равен 0, если значение этого разряда равно 1, то результат является копией множимого.

2. Правило сдвига. При использовании очередного разряда множителя для формирования частичного произведения производится сдвиг множимого на один разряд (позицию) влево.

3. Правило сложения. Каждый раз, когда значение разряда множителя равно 1, к результату необходимо прибавить множимое, расположенное в позиции, определенной правилом сдвига.

4. Определение результирующего произведения. Искомое произведение есть результат выполнения всех операций сдвига и сложения.

Рассмотренный выше пример подтверждает сформулированные правила умножения путем сдвига и сложения. При использовании первого (младшего значащего) разряда множителя множимое не смещается, а поскольку значение этого разряда равно 0, то первое частичное произведение равно 0. Следовательно, равно 0 и текущее значение результата. При использовании второго разряда со значением, равным 0, множимое сдвигается на один разряд влево, но сложение, как и в предыдущем случае, не выполняется. Значение третьего разряда множителя равно 1, поэтому множимое сначала сдвигается еще на один разряд влево, а затем добавляется в качестве слагаемого к текущему значению результата. При использовании четвертого разряда, значение которого равно 1, осуществляется сдвиг множимого на один разряд влево по сравнению с его позицией после операции с третьим разрядом. Затем множимое добавляется к текущему значению результата. Поскольку остальные разряды множителя (с пятого по восьмой) содержат нули, никаких добавлений к текущему значению результата больше не происходит.

Применительно к рассмотренному примеру умножения 17_{10} на 12_{10} правила сдвига и сложения можно продемонстрировать в более компактной форме, а именно:

```
00010001
00001100
--------
 10001        Множимое, сдвинутое влево на
              2 разряда
+
10001         Множимое, сдвинутое влево на
              3 разряда
--------
11001100      Сумма сдвинутых множимых
```

Умножение путем сдвига и сложения существенно упрощает двоичное умножение. Однако это возможно лишь благодаря тому, что при умножении двоичного числа на 0 получается 0, а результат умножения двоичного числа на 1 есть само это число.

12. Двоичное деление

Деление-это операция, обратная умножению. Иначе говоря, при делении операцию вычитания повторяют до тех пор, пока уменьшаемое не станет меньше вычитаемого. Число этих повторений показывает, сколько раз вычитаемое укладывается в уменьшаемом.

Процедура деления несколько сложнее процедуры умножения. Рассмотрим, например, деление числа 204_{10} на число 12_{10}, пользуясь правилами десятичной и двоичной арифметик:

Десятичное деление

```
204   12
      --
12    17
--
 84
 84
--
  0
```

Двоичное деление

```
1100 1100   1100
            ----
1100        10001
----
   01
    0
   ---
   011
    0
   110
    0
   ---
   1100
   1100
   ----
      0
```

Двоичное деление начинается с анализа делимого (1100 1100) и делителя (1100). Сразу же обнаруживаем, что делитель (1100) точно укладывается в 1100, а поэтому записываем цифру 1 в поле, предоставленное для формирования частного. Умножаем делитель на 1 и вычитаем результат из 1100. Разность равна нулю, т. е. меньше делителя, а потому процесс деления можно продолжить. Объединяем нуль остатка со значением следующего разряда делимого, равным 1. Поскольку делитель (1100) укладывается 0 раз в числе 1, записываем 0 в поле представления частного, а число 1 объединяем со следующей цифрой делимого и т.д. Описываемая процедура продолжается до тех пор, пока делимое не оказывается исчерпанным.

Из рассмотренного выше следует, что реализовать операцию деления в вычислительной машине не столь просто, как операцию умножения. Слишком много усилий требуется для выяснения того, сколько раз делитель укладывается в определенном числе. Несмотря на указанные затруднения, был разработан несложный способ двоичного деления, используемый в микропроцессорах.

Процедура двоичного деления в действительности проста, потому что каждый бит частного принимает одно из двух возможных значений: 1 или 0. Как и в случае двоичного умножения, удобным оказывается использование операции сдвига. Продемонстрируем упомянутый способ двоичного деления на рассмотренном выше примере, предварительно представив делитель (12) в дополнительном коде. Это позволит ограничиться двоичным сложением во всех случаях, когда нужно выполнять сложение или вычитание. Дополнительный код двоичного представления числа 12_{10} определяется следующим образом:

```
0 1100   Двоичное представление числа 12
1 0011   Обратный код двоичного представления числа 12
0 0001   Единица, добавляемая к обратному коду
-------
1 0100   Дополнительный код числа 12 (т.е. число – 12)
↑   ↑
Бит знака Биты абсолютной величины
```

Теперь можно приступить к собственно делению. Как и в случае выше рассмотренного так называемого длинного двоичного деления, необходимо определить, сколько раз делитель укладывается в числе, образованном соответствующим количеством старших значащих битов делимого.

Конечно, микропроцессор не может строить догадок относительно того, сколько раз делитель укладывается в указанном числе. В действительности микропроцессор начинает вычитать делитель из этого числа. Если делитель не будет укладываться в упомянутой части делимого, всегда .можно вернуть вычтенные биты обратно делимому. О том, что делитель не укладывается, свидетельствует появление отрицательного результата вычитания (бит знака разности равен 1).

Предпримем попытку выполнить вычитание первый раз:

```
0 11001100          Делимое
1 01000000          Вычитаемое число 12₁₀
-----------
0 00001100          Первый результат
↑
Наличие здесь нуля
означает, что 1-й бит
искомого частного ра-
вен 1               Частное: 1XXXX
```

Если делитель укладывается в соответствующую часть делимого, бит знака равен 0. Это означает, что результат деления – положительное число. В данном примере это оказалось именно так. А поэтому первый бит искомого результата (частного) равен 1.

Выполним второй шаг процедуры деления. Следует еще раз попытаться выполнить вычитание делителя. Но предварительно необходимо сдвинуть первый результат. Сдвиг должен быть таким, чтобы при последующей операции вычитания был сформирован второй бит частного. В результате сдвига получим

```
0 00001100   Первый результат до сдвига
0 0001100    Первый результат после сдвига
```

Теперь можно выполнить вторую операцию вычитания:

```
0 0001100          Сдвинутый первый результат
1 0100000          Вычитаемое число 12(10)
---------
1 0101100          Второй результат
↑
Наличие здесь единицы означает, что
2-й бит искомого
частного равен 0   Частное: 10XXX
```

Результат этого вычитания содержит 1 в позиции знака, т. е. получено отрицательное число. Следовательно, делитель не укладывается в соответствующем числе. Поэтому прежде всего во вторую (по старшинству) позицию поля представления частного следует записать 0. Кроме того, поскольку вычитание не состоялось, необходимо вернуть биты делителя обратно первому результату:

```
1 0101100          Второй результат
0 1100000          Возвращаемое число 12(10)
---------
0 0001100          Сдвинутый первый результат (полученный вторично)
```

Теперь наступает очередь следующего сдвига:

```
0 000100           Сдвинутый первый результат
0 001100           Первый результат после второго сдвига
```

И вновь все готово для попытки уложить делитель в соответствующее число (дважды сдвинутый первый результат). Выполняем третью операцию вычитания:

```
0 001100           Дважды сдвинутый первый результат
1 010000           Вычитаемое число 12(10)
---------
1 011100           Третий результат
↑
Наличие здесь единицы
означает, что 3-й бит ис-
комого частного равен 0 Частное: 100XX
```

Третий результат – отрицательное число, т. е. подвергнутый двойному сдвигу первый результат оказался меньше делителя, а следовательно, третий бит искомого частного также равен 0. Поэтому число 12_{10} (делитель) следует вернуть первому результату, сдвинутому дважды:

```
1 011100           Третий результат
0 110000           Возвращаемое число 12(10)
---------
0 001100           Дважды сдвинутый первый результат
```

После исправления допущенной «ошибки» необходимо выполнить очередной сдвиг:

```
0 001100  Дважды сдвинутый первый результат
0 01100   Первый результат после третьего
          сдвига
```

Теперь можно предпринять попытку вычесть делитель из первого результата, подвергнутого троекратному сдвигу:

```
0 01100          Первый результат после третье-
                 го сдвига
1 01000          Вычитаемое число 12(10)
---------
1 10100          Четвертый результат
↑
Наличие здесь еди-
ницы означает, что
4-й бит частного ра-
вен 0            Частное: 1000X
```

Поскольку результат этого вычитания есть число отрицательное, бит частного равен 0, и число 12_{10} необходимо прибавить к четвертому результату:

```
1 10100     Четвертый результат
0 11000     Возвращаемое число 12
---------
0 01100     Первый результат после третьего
            сдвига
```

Выполняем четвертый сдвиг:

```
0 01100     Первый результат после третьего
            сдвига
0 1100      Первый результат после четверто-
            го сдвига
```

Опять предпринимается попытка вычитания делителя из последнего результата. Обратите внимание, что на этот раз число 12_{10} вычитается из числа 12_{10} :

```
0 1100           Первый результат после четвер-
                 того сдвига
1 0100           Вычитаемое число 12(10)
-------
0 0000           Пятый результат
↑
Наличие здесь нуля
означает, что 5-й бит
частного равен 1    Частное: 10001
```

На этом выполнение процедуры заканчивается. Таким образом, при делении числа 1100 1100 на число 1100 результат равен 10001.

Рассмотренная процедура двоичного деления несколько сложнее процедуры

двоичного умножения. Однако ее нетрудно осуществить на практике, если последовательно выполнять предписываемые операции. Поскольку

правила деления четко сформулированы, они могут быть реализованы в микропроцессоре. Следует, однако, удостовериться, что эти правила справедливы для всех возможных ситуаций.

Рассмотрим еще один пример, чтобы быть уверенным в правильности понимания описанной выше процедуры. Требуется разделить число 35 на число 5. Процедура так называемого длинного деления двоичных эквивалентов этих чисел имеет следующий вид:

```
100011   101
000      |---
---      0111
1000
 101
----
00111
  101
  ---
  0101
   101
   ---
   000
```

Представим делитель в дополнительном коде:

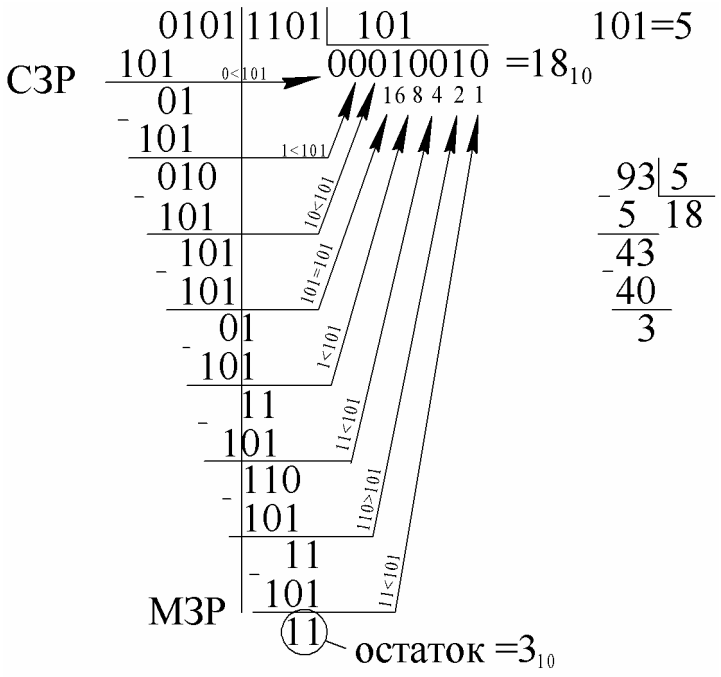
0101	Число 5_{10}
1010	Обратный код числа
0001	Единица, добавляемая к обратному коду
1101	Дополнительный код числа

Теперь опишем процедуру деления методом вычитания и сдвига влево:

Двоичная арифметика	Бит частного	Наименование операндов
0100011		Делимое 35_{10}
1011000		Дополнительный код числа 5_{10}
1111011	0	Первый результат
1111011		Первый результат
0101000		Возвращаемое число 5_{10}
0100011		Делимое
100011		Делимое после первого сдвига
101100		Дополнительный код числа 5_{10}
001111	1	Второй результат
01111		Второй результат после сдвига на 1 позицию
10110		Дополнительный код числа 5_{10}
00101	1	Третий результат
0101		Третий результат после сдвига на 1 позицию
1011		Дополнительный код числа 5_{10}
0000	1	Четвертый результат

Если процедуру деления продолжить и далее, то все последующие биты частного окажутся равными 0. Обычно известно число ожидаемых

битов частного и местоположение двоичной точки. Например, при делении 16-битового делимого на 8-битовый делитель частное состоит из 8 бит. Когда известно положение двоичной точки делимого и делителя, то автоматически определяется положение двоичной точки частного.



12' Еще двоичное деление (скомпоновать с предыдущим)

35 : 5

32	16	8	4	2	1	
1	0	0	0	1	1	- 35_2
			1	0	1	- 5_2
		X	X	X	X	- результат

(0)101 – код числа 5_{10}
1010 - обратный код
1011 – доп. код числа 5_{10}

			↓бит переноса						
	+	(0)	1	0	0	0	1	1	число 35
		1	0	1	1	0	0	0	-5
		1	1	1	1	0	1	1	результат первого вычитания <0 =>
	+								частное 0XXX
		0	1	0	1	0	0	0	возвращаем 5
отбр. <	(1)	0	1	0	0	0	1	1	восстановленное делимое

	+	1	0	0	0	1	1	0	1 сдвиг
		1	0	1	1	0	0	0	-5
отбр. <	(1)	0	0	1	1	1	1		результат второго вычитания >0 => частное 01XX

	+	0	1	1	1	1	2 сдвиг
		1	0	1	1		-5
отбр. <	(1)	0	0	1	0	1	результат третьего вычитания >0 => частное 011X

	+	0	1	0	1		3 сдвиг
		1	0	1	1		-5
отбр. <	(1)	0	0	0	0		результат четвертого вычитания >0 => частное: 0111 - 7_{10}

Пример выполнения двоичного «длинного» деления:

```
_ 1 1 0 0 1 1 0 0 | 1 1 0 0
  1 1 0 0         | 1 0 0 0 1
  ---------
        _ 0 1 1 0 0
          1 1 0 0
          -------
                0
```

Как видим, двоичное деление сводится к последовательному выполнению операций вычитания и сдвига. Важной особенностью является то, что вычитание начинается со старших разрядов. Это требует определенной подготовки операндов (делимого и делителя) перед началом вычислений на машине заданной разрядности. Этот процесс условно назовем нормализацией операндов. Его суть заключается в последовательном сдвиге операндов влево до заполнения старшего бита (не включая знаковый восьмой разряд). Операцию вычитания будем производить в дополнительном коде.

Пример: разделить 37 на 5

$37_{10} = 00100101$; $5_{10} = 00000101$

Нормализация операндов:

$37_H = 01001010$ (1 сдвиг влево) $S_1=1$

$5_H = 01010000$ (потреб. 4 сдвига влево) $S_2=4$

Дополнительный код числа 5_H:

$-5_H = 10110000$

Теперь приступим к выполнению деления.

Двоичная арифметика		Комментарии	Результат (частное)
	01001010	37н	????
+	10110000	(-5)н – пробуем выполнить первое вычитание	
	11111010	результат <0, следовательно, первая цифра частного – 0	0???
+	01010000	+5н – восстанавливаем делимое, поскольку 1е вычитание неудачно	
	01001010	восстановленное делимое	
	10010100	сдвиг делимого влево	
+	10110000	(-5)н – пробуем выполнить вычитание	
	01000100	результат >0, следовательно, 2-я цифра частного – 1	01??
	10001000	сдвиг влево	
+	10110000	(-5)н – третье вычитание	
	00111000	результат >0, следовательно, 3-я цифра частного – 1	011?
	01110000	сдвиг остатка влево	
+	10110000	(-5)н – четвертое вычитание	
	00100000	результат >0, следовательно, 4-я цифра частного – 1	0111

Положение двоичной точки в результате определяется разрядностью исходных операндов. В общем случае, если делимое занимает n_1 разрядов, а делитель – n_2, то частное будет содержать $n_3=n_1-n_2+1$ целых разрядов. Разрядность частного можно узнать на стадии нормализации операндов. Если S_1 – число сдвигов влево делимого, S_2 – число сдвигов делителя, то $n_3=S_2-S_1+1$ – число разрядов целой части частного.

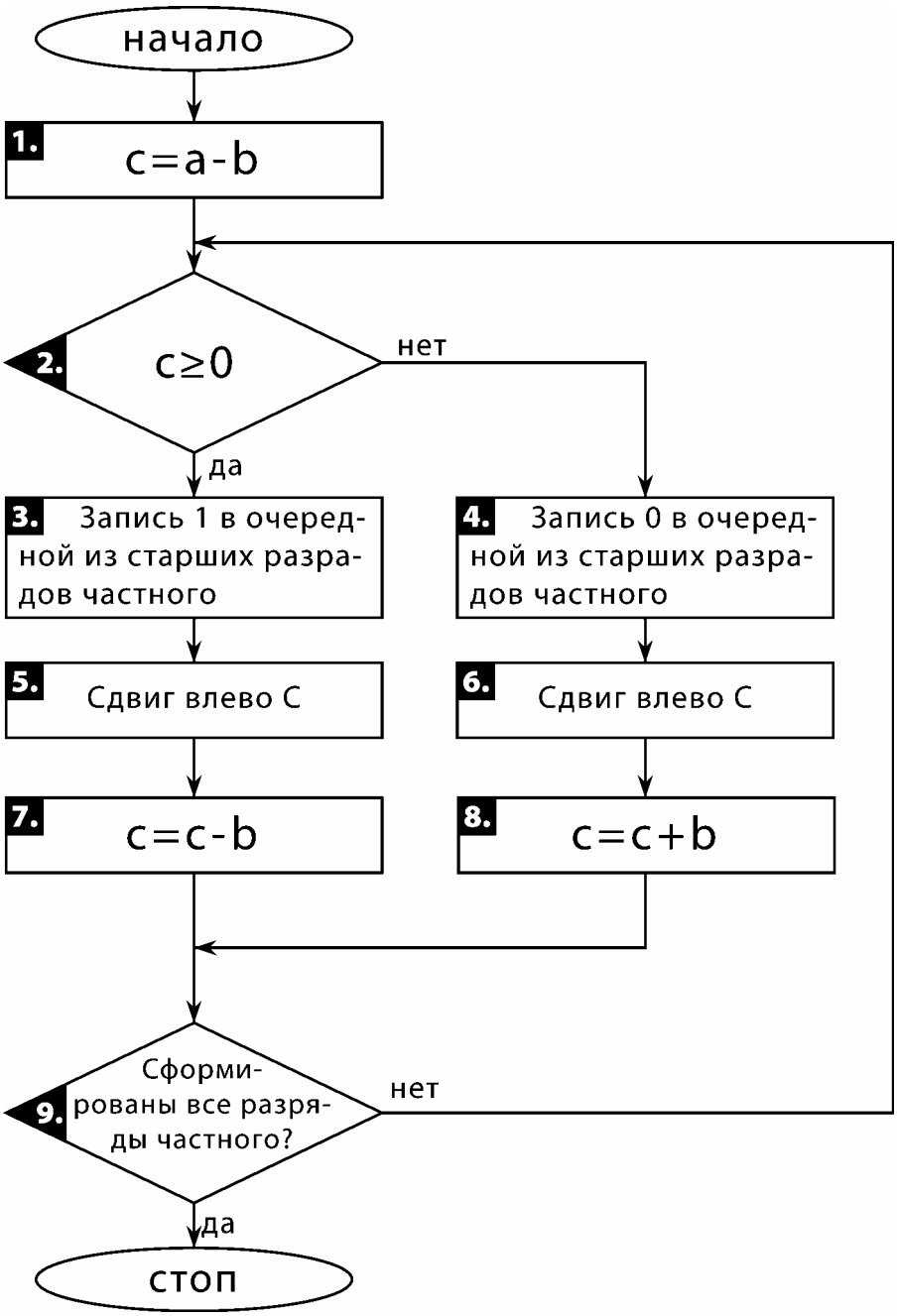


Схема алгоритма деления положительных чисел a и b.

13. Арифметика повышенной точности

При работе с микропроцессором часто выясняется, что длина слов, которыми он оперирует, недостаточна для достижения определенной точности вычислений. Это означает, что необходимо представление чисел больших размеров и в такой форме, чтобы микропроцессор мог работать с ними. Например, широко распространенные 8-разрядные микропроцессоры позволяют использовать числа в диапазоне от — 128 до + 127. Очевидно, что для большинства задач такой диапазон неприемлем. Используя два 8-битовых слова с представлением отрицательных чисел в дополнительном коде, получим диапазон от – 32 768 до 32767. В этом случае ошибка представления чисел оценивается одной шестидесятитысячной, или +

0,0015%. Для решения многих задач указанной двойной точности вполне достаточно. Однако иногда требуется еще более высокая точность вычислений, достигаемая, например, тройной точностью представления чисел: 1 бит для знака и 23 бит для абсолютной величины числа.

При использовании в 8-разрядном микропроцессоре тройной точности представления чисел диапазон последних простирается от - 8 388 608 до + 8 388 607, включая 0. Это обеспечивает значительно меньшую ошибку представления чисел по сравнению с использованием для этих целей только 8 бит. Ошибка оценивается величиной меньшей, чем одна миллионная. Однако за все приходится расплачиваться: при работе с арифметикой повышенной точности требуется больший объем памяти для хранения данных и более интенсивная работа микропроцессора. Пусть, например, вы хотите использовать арифметику тройной точности в 8-разрядной вычислительной системе. В этом случае для выполнения сложения недостаточно извлечь из памяти два слова, сформировать их сумму в аккумуляторе и записать результат в 1-байтовую область памяти. Сначала необходимо произвести обращение к младшему значащему байту каждого числа. После сложения двух байтов результат записывают в память, а возможные при этом переносы подлежат временному хранению. Затем из памяти извлекают средние по значимости байты и складывают с битами переноса, полученными в результате предыдущей операции сложения. Результат записывают в память на место, специально зарезервированное для среднего байта суммы. Наконец из памяти извлекают старшие значащие байты, складывают их, к сумме добавляют биты переноса, полученные при предыдущей операции сложения, и результат записывают в область памяти, зарезервированную для старшего значащего байта суммы.

Из рассмотренного примера следует, что сложение тройной точности по сравнению со сложением одинарной точности, т.е. сложением 8-разрядных двоичных чисел, занимает времени в три раза больше; кроме того, при этом требуется в три раза больший объем памяти. И это не единственный недостаток арифметики повышенной точности. Если, например, в процессе сложения чисел тройной точности произойдет прерывание, инициированное другими шагами программы, то необходимо временно сохранить содержимое регистра состояния. В противном случае будут утеряны промежуточные биты переноса, что приведет к неверному результату сложения.

Арифметику повышенной точности можно использовать применительно ко всем четырем основным операциям: сложению, вычитанию, умножению и делению.

14. Арифметика чисел с плавающей точкой

Не все проблемы могут быть разрешены при использовании арифметики повышенной точности. Так, проводимое до настоящего момента рассмотрение было ограничено целыми числами. Мы не знаем, как обращаться с дробной частью числа. Нам неизвестно, как представлять очень большие и очень малые числа.

Перечисленные проблемы разрешимы с помощью арифметики чисел с плавающей точкой (запятой), позволяющей микропроцессору отслеживать положение десятичной точки. Это достигается благодаря использованию представления десятичных дробей в нормализованном виде, т.е. в виде мантиссы, диапазон значений которой простирается or 0,1 до 1, и порядка-показателя степени числа 10. Например, число 50 представляется как 0,5 x 10^2, а число — 750 как — 0,75 x 10^3. Очень малое число 0,00105 записывается в виде 0,105 x 10^{-2}.

Числа с плавающей точкой хранятся в микропроцессоре так же, как и целые числа. Записывается мантисса со знаком и порядок со знаком. Не следует забывать, что мантисса – число, принадлежащее диапазону 0,1-1, а порядок - показатель степени числа 10.

Представление числа в форме с плавающей точкой в 8-разрядном микропроцессоре можно изобразить схематически следующим образом:

Адрес байта	Содержимое байта
M + 3 (4-й байт)	+ 7-битовый порядок или –
M + 2 (3-й байт)	+ 7 старших битов мантиссы или –
M + 1 (2-й байт) -	8 средних битов мантиссы
M (1-й байт)	8 младших битов мантиссы

Число в форме с плавающей точкой занимает 4 байт. В первом байте расположены 8 младших битов мантиссы, во втором-8 средних битов, в третьем-7 старших битов и бит знака. Мантисса представлена как число тройной точности. Четвертый байт занят порядком: 7 бит величины и 1 бит знака. Согласно такому представлению, число с плавающей точкой имеет следующий формат:

± XXXXXXXXXXXXXXXXXXXXXXX 2* $^{xxxxxxx}>$

где X – условное обозначение двоичной цифры (бита). Арифметика чисел с плавающей точкой позволяет оперировать числами от $-2^{23}xN^{127}$ до $+(2^{23}-l)xN^{127}$. Значение N обычно равно 2, но иногда и 10. Следовательно, диапазон представления чисел очень большой.

Если микропроцессор «настроен» на работу с числами, представленными в форме с плавающей точкой, то, как правило, все арифметические операции выполняются под управлением пакета подпрограмм арифметики с плавающей точкой. Эти подпрограммы обеспечивают размещение числа в 4 байт памяти в соответствии с форматом числа с плавающей точкой. Однако, если число находится в таком «аккумуляторе с плавающей точкой», можно обращаться к подпрограммам выполнения арифметических операций над содержимым этого «аккумулятора» и числами, расположенными в других областях памяти. Большинство пакетов подпрограмм арифметики чисел с плавающей точкой

содержат не только подпрограммы выполнения сложения, вычитания, умножения и деления, но и возведения в квадрат, извлечения квадратного корня, вычисления тригонометрических функций (синуса, косинуса, тангенса) и логарифмов. Будучи один раз написанным, пакет таких подпрограмм используется многократно и часто воспринимается пользователями как расширение аппаратных средств микро-ЭВМ. Однако не следует забывать, что, говоря об «аккумуляторе с плавающей точкой», мы имеем в виду четыре смежные области памяти, к которым обращаются подпрограммы арифметики чисел с плавающей точкой.

Пакет подпрограмм арифметики чисел с плавающей точкой выполняется значительно медленнее, чем традиционные команды микропроцессора. Так, время сложения чисел с плавающей точкой в 10-20 раз больше, чем время реализации команды сложения аппаратными средствами микропроцессора. Это объясняется тем, что в сложении чисел с плавающей точкой участвуют от 20 до 30 команд микропроцессора. Подпрограмме приходится не только манипулировать числами тройной точности, но и постоянно следить за значением порядка.

Пакет подпрограмм арифметики чисел с плавающей точкой используется почти всегда, когда микропроцессор управляет работой системы, оперирующей самыми разнообразными числовыми данными. Одним из типичных примеров применения этого пакета является программирование работы микро-ЭВМ на языке БЕЙСИК. Программное обеспечение всех языков высокого уровня, подобных языку БЕЙСИК или ПАСКАЛЬ, выключает свой собственный пакет подпрограмм арифметики чисел с плавающей точкой. Структура таких пакетов весьма сложна по сравнению с их аналогами, используемыми для выполнения ограниченного набора функций системы управления микропроцессора.

Л.7 Составление блок-схем алгоритмов

Графическое изображение алгоритма решения задачи в виде блок-схемы - важный этап подготовки задачи к решению на ЭВМ. Блок-схема алгоритма помогает пользователю адекватно представить работу программы. Анализируя блок-схему, можно выяснить, как различные входные данные влияют на окончательный результат.

Блок-схема алгоритма (БСА) - одна из важных частей документации, подготавливаемой для решения задачи на ЭВМ. Однако алгоритмизация и графические средства изображения алгоритмов не являются прерогативой вычислительной техники; они могут использоваться для описания решения любых задач. Например, вы можете составить блок-схему алгоритма вашего поведения утром - от момента подъема с постели до ухода на работу или в учебное заведение.

Блок-схема алгоритма составляется из отдельных блоков. Различают четыре типа блоков, каждый из которых имеет один или несколько входов и один или несколько выходов (рис. 1). Стрелками обозначают направление хода вычислений. Блок в форме прямоугольника символизирует выполнение каких-либо операций по обработке данных; текст внутри блока является кратким описанием этого процесса обработки (рис.1,а).

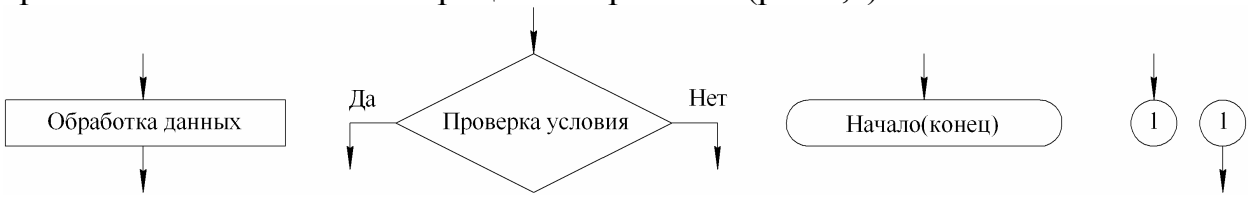


Рис. 1. Типы блоков, используемых для составления блок-схем алгоритмов: а) блок обработки данных; 6) блок проверки условия; в) блок начала или конца алгоритма; г) соединители.

Пример (поддержине температуры)

1. Составьте алгоритм решения с помощью ЭВМ формулируемой ниже задачи..

В лаборатории стандартов необходимо поддерживать температуру равной 20°С независимо от температуры снаружи помещения. В вашем распоряжении имеются кондиционер, отопительная система и два термостата: один - для измерения температуры, равной или меньше 20°С, другой -для измерения температуры больше или равной 20°С.

Обозначения для развернутой (подробной) БСА

t_1 – показания 1 термометра : (0: $t>20$; 1: $t\leq 20$C,)

t_2 – показания 2 термометра : (0: $t<20$; 1: $t\geq 20$C,)

x = 1 – включить отопление; x = 0 – выключить отопление

y = 1 – включить кондиционер; y = 0 – выключить кондиционер

z – сигнализация:

z = 0 – остановить систему; z = 1 – запустить систему

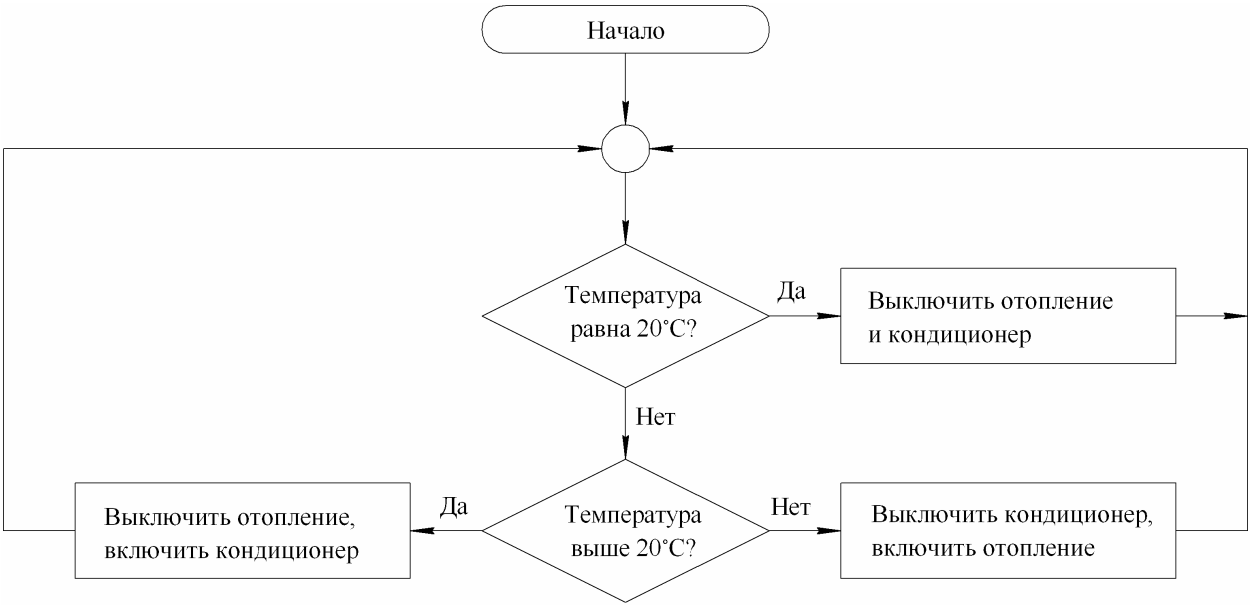


Рис. 2. БСА функционирования системы.

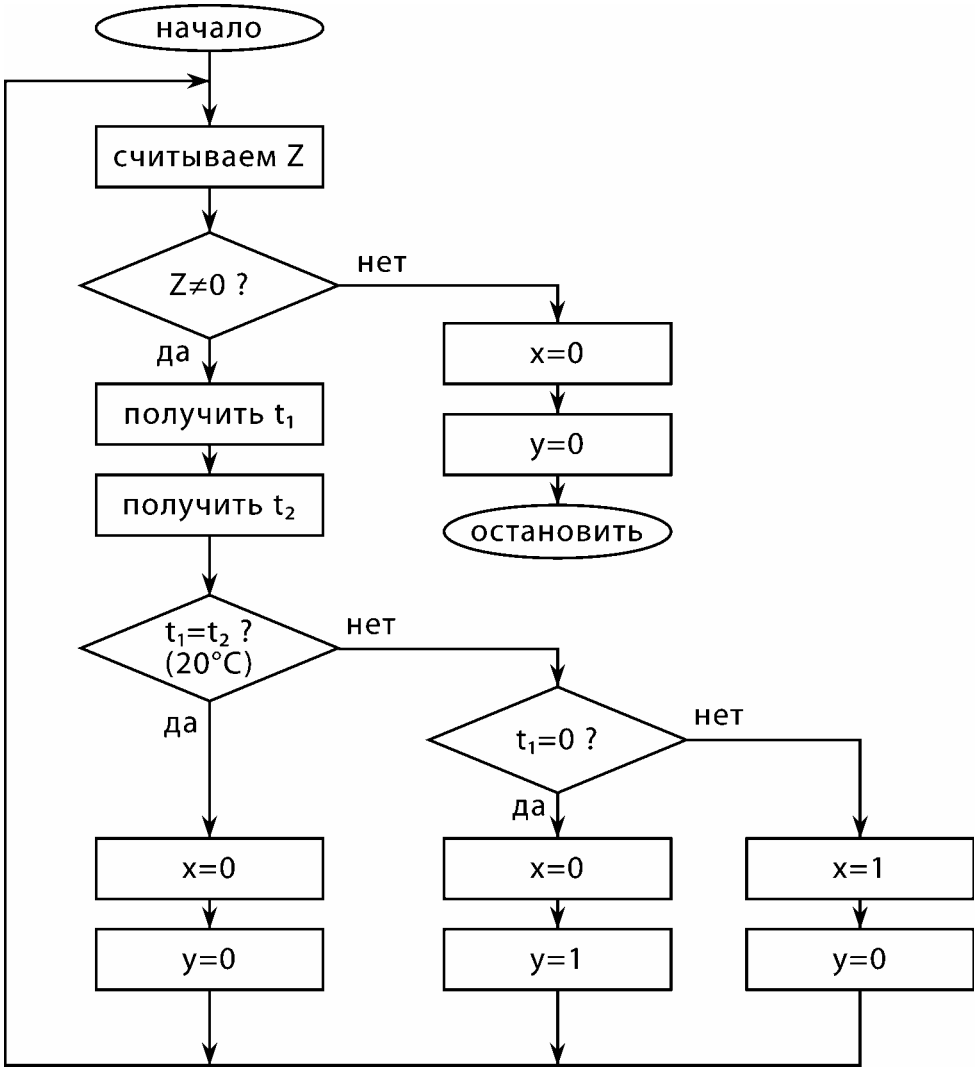


Рис. 3 - Развернутая БСА

Подпрограммы

Подпрограмма-это часть программы, используемая обычно несколько раз в процессе выполнения программы. Однако текст подпрограммы записывается программистом только один раз. Когда же программисту необходимо воспользоваться подпрограммой, достаточно указать в программе соответствующую ***команду вызова*** (обращения к подпрограмме), адресующуюся к области памяти, в которой расположена подпрограмма. Большинство языков программирования располагает для этих целей специальными командами вызова подпрограмм CALL, которые не только инициируют выполнение подпрограмм, но и побуждают вычислительную машину запоминать состояние программы в момент обращения к подпрограмме. В результате вызова подпрограммы ей передается управление с целью пошагового выполнения ее операций. Последней выполняемой командой подпрограммы, как правило, является так называемая ***команда возврата*** RETURN, передающая управление обратно вызвавшей программе, заставляя последнюю продолжить выполнение с той команды, которая непосредственно следует за командой вызова подпрограммы.

Основное достоинство подпрограмм заключается в том, что благодаря возможности их многократного использования сокращается текст программы в целом. Вместо того чтобы по мере необходимости повторять запись одного и того же фрагмента программы, достаточно оформить запись фрагмента как подпрограмму и обращаться к ней столько раз, сколько требуется в соответствии с алгоритмом решения задачи.

Л.8.1. INTERFACE Организация интерфейса в МПС.

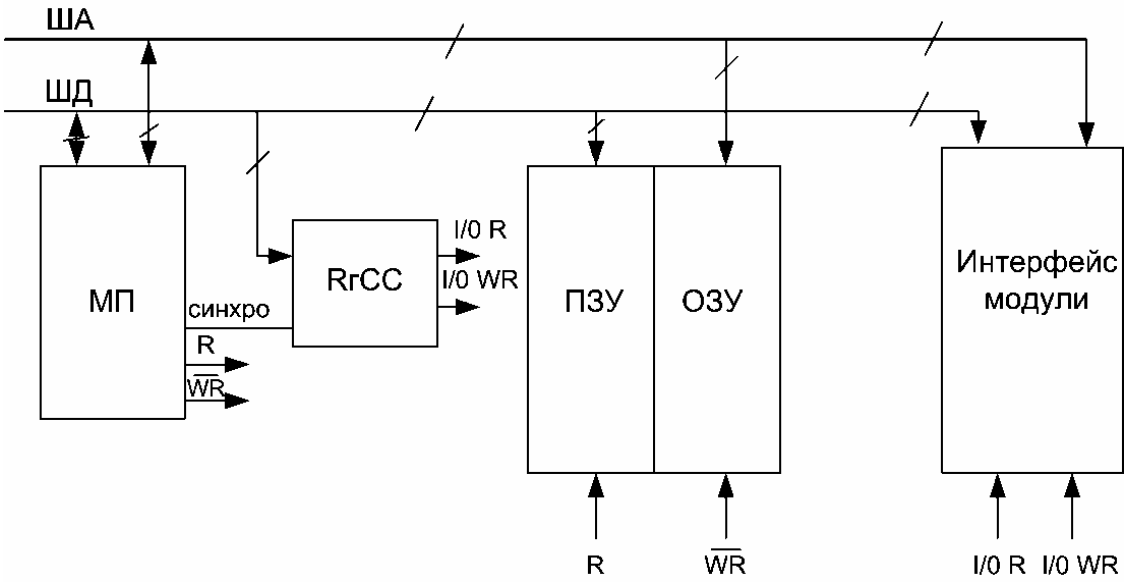
МПС осуществляет обмен информацией с внешним миром посредством периферийных устройств. Все ПУ для функционирования требуют определенного набора управляющих сигналов, которые непосредственно не формируются МП. Поэтому, для подключения ПУ используют специальные электронные схемы, называемые интерфейсными модулями. Кроме аппаратной части для организации интерфейса необходимо разрабатывать программное обеспечении, опр-ее режимы (протоколы) обмена, направление обмена, необходимость использования прерываний и т.д. Т.о., интерфейс – это совокупность аппаратных и программных средств, с помощью которых компоненты МПС могут обмениваться информацией.

2 основных принципа организации интерфейса между процессором, памятью и ПУ:

1. Двухшинную организацию, или интерфейс с изолированной шиной. При этой организации существует два тракта передачи информации: МП-ЯП, и МП-ПУ. Обмен по этим трактам осуществляется разными группами команд.

2. Одношинную организацию, или интерфейс с общей шиной. При этом часть общего адресного пространства отводится для ПУ, регистры которых адресуются так же, как и ЯП.

Интерфейс с изолированной шиной



Раздельная адресация ЯП и ПУ. Это осуществляется путем и си-ия отдельных групп команд для обмена с памятью и периферией. Для обмена с ПУ используются команды IN B_2 и OUT B_2 , где B_2 – второй байт, номер ПУ. Команды позволяют адресоваться к 256 портам ввода и 256-ти портам вывода.

Управление обменом осуществляется под действием сигналов I/O R и I/O WR шины управления (6 и 4 биты слова состояния). Адрес из МП передается по восьми младшим и восьми старшим линиям ША. Обмен осуществляется между аккумулятором и портом ПУ.

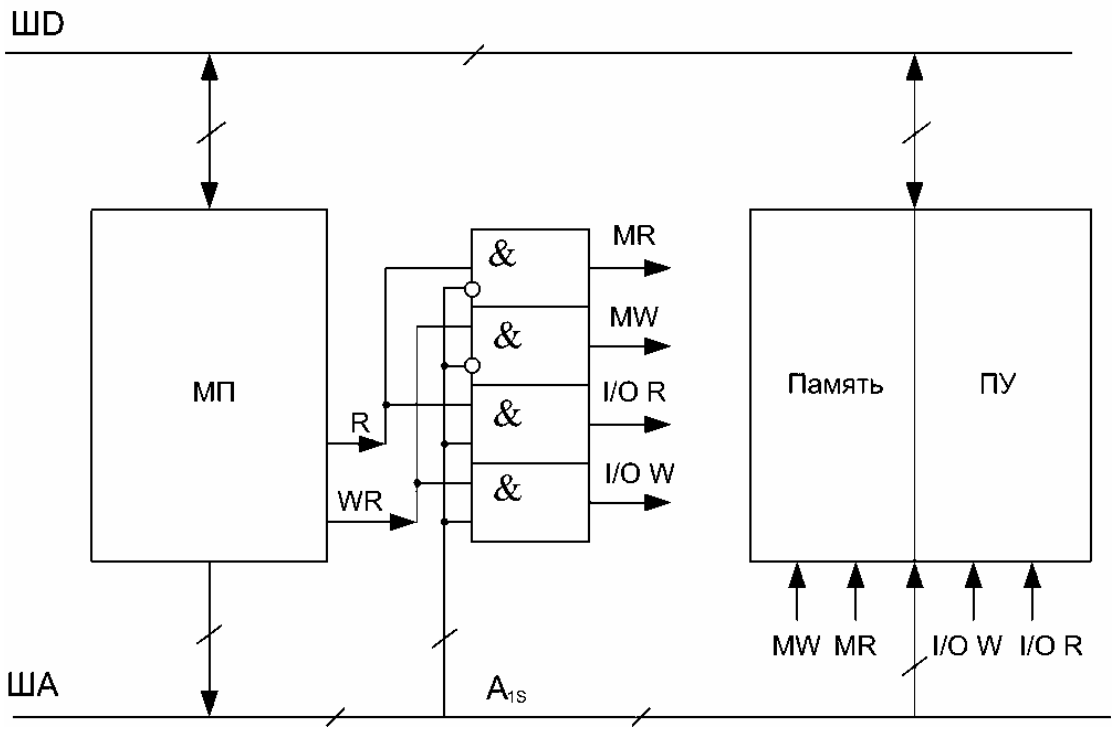
Обмен информацией с памятью выполняется путем использования команд пересылки данных.

Недостатки интерфейса с изолированной шиной:

1. Процедуры обмена с ПУ очень медленны, поскольку выполняются через аккумулятор.
2. Число ПУ не более 256.

Достоинство: в рассмотренных прикладных программ все адресное пространство.

Интерфейс с общей шиной



Часть адресного пр-ва отводится для ПУ, порты которых адресуются так же, как и ЯП. Обращение к ПУ осуществляется посредством набора команд, используемых для обмена данными с памятью. Команды IN и OUT не используются. Такой принцип построения интерфейса реализуется путем выделения старшего (или любого другого) разряда ША в качестве признака обращения к памяти. В качестве управления сигналов для обращения к памяти и ПУ используются сигналы R, WR МП и этот разряд адреса.

Достоинства:

1. Расширение набора команд для обращения к ПУ.
2. Значит увеличение числа ПУ.
3. возможность внепроцессорного обмена данными ПУ – ЯП.

Недостаток: Уменьшение объема адресуемой памяти.

Л. 8.2 Системный контроллер КР580ВК28 и КР580ВК38

Предназначен для обмена данными между МП и периферийными устройствами, а также для увеличения нагрузочной способности шины данных. Условное обозначение микросхемы приведено на рис.1, назначение выводов – в табл. 1.

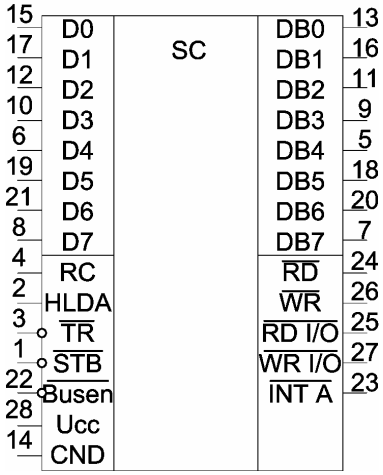


Рис.1

Таблица 1.

Вывод	Обозначение	Тип выв	Функц. назначение
1	STB	вх	Стробирующий сигнал состояния
2	HLDA	вх	Подтверждение захвата
3	TR	вх	Выдача информации
4	RC	вх	Прием информации
5,7,9,11,13,16,18,20	DB[4,7,3,2,0,1,5,6]	вх/вых	Канал даннах системы
6,8,10,12,15,17,19,21	D[4,7,3,2,0,1,5,6]	вх/вых	Канал данных МП
14	GND	-	Общий
22	BUSEN	вх	Управление передачей данных и выдачей сигналов
23	INT A	вых	Подтвержд-е запроса прерывания
24	RD	вых	Чтение из памяти
25	RD I/O	вых	Чтение из УВ/В
26	WR	вых	Запись в память
27	WR I/O	вых	Запись в УВ/В
28	Ucc	-	Напряжение питания +5В

МС ВК28 и ВК38 отличаются способом формирования двух управляющих сигналов WR и WR I/O (см. ниже).

Системный контроллер формирует управляющие сигналы в соответствии со словом внутреннего состояния процессора:

- при обращении к ЗУ: RD и WR
- при обращении к устройствам ввода/вывода: RD I/O и WR I/O;

сигнала INT A, а также обеспечивает прием и передачу 8-ми разрядной информации между шиной данных МП D7..D0 и системной шиной данных DB7..DB0.

Системный контроллер содержит двунаправленную буферную схему данных, регистр состояния и дешифратор управляющих сигналов. 8-ми

разрядная двустабильная схема данных принимает информацию с ШД МП по выводам D7..D0 и передает информацию о внутреннем состоянии МП (слово состояния) по сигналу “SYNC”=”STB” в регистр состояния, а на системную ШД по выводам DB7..DB0 выдает данные в цикле записи по сигналу TR. В цикле чтения по сигналу RC буферная схема принимает данные с системной ШД по выводам DB7..DB0 и передает по выводам D7..D0 на ШД МП.

Регистр состояния по входному сигналу STB фиксирует слово внутреннего состояния МП в такте Т1 каждого машинного цикла МП. Дешифратор управляющих сигналов формирует один из управляющих сигналов в каждом машинном цикле:

- при чтении ЗУ – RD
- при записи в ЗУ – WR
- при чтении из УВВ – RD I/O
- при записи в УВВ – WR I/O
- при подтверждении запроса прерывания – INT A.

Асинхронный сигнал BUSEN управляет выдачей данных и управляющих сигналов с дешифратора: при напряжении низкого уровня данные и сигналы выдаются, при высоком все выходы МС переводятся в высокоомное состояние.

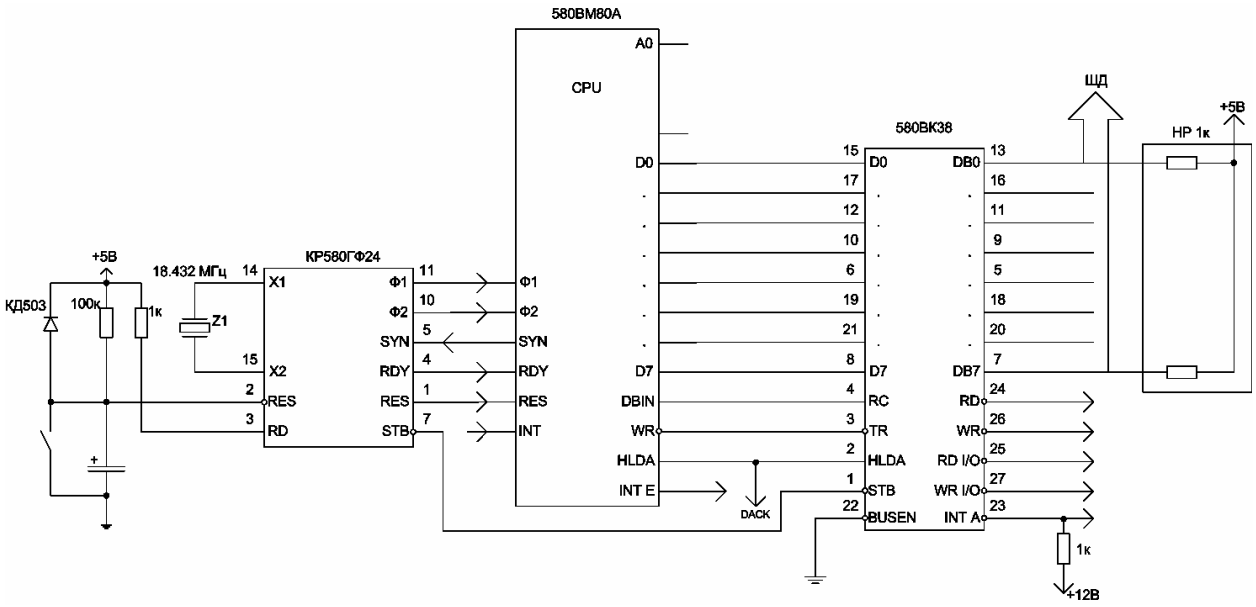
Напряжение высокого уровня на входе HLDA переводит выходы RD, RD I/O, INT A в пассивное состояние (высокий лог. уровень) и блокирует передачу информации через буферную схему данных.

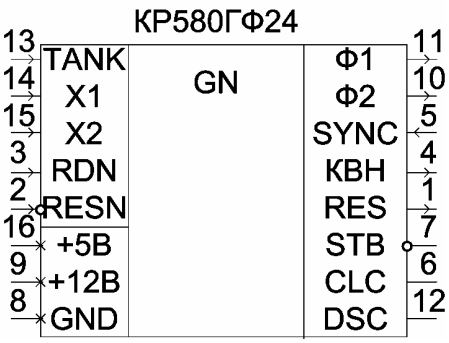
Управляющие сигналы WR и WR I/O формируются в цикле записи в МС ВК28 по сигналу TR; в МС ВК38 – по сигналу STB.

При работе с МП КР580ВМ80А системный контроллер в цикле подтверждения запроса прерывания формирует три сигнала INT A для приема трёх байт команды CALL от контроллера прерываний. В небольших МПС вывод INT A МС ВК28/ВК38 можно подсоединить к напряжению +12В через сопротивление 1 кОм. Тогда во время действия сигнала RC буферная схема данных формирует код команды RST 7 и передаёт его на ШД МП. Т.о., МС обеспечивает единственный вектор прерывания с номером 7 без использования дополнительных компонентов.

МС **КР580ГФ24** служит для управления БИС микропроцессорной системы. Формирует положительные импульсы Ф1 и Ф2, а также положительный импульс стандартного ТТЛ уровня и отрицательный импульс строба состояния. Формирование всех этих импульсов происходит с периодом повторения, равным 9 периодам колебаний задающего генератора. Для работы БИС предусмотрено подключение внешней времязадающей цепи, состоящей из индуктивно-емкостного колебательного контура, подключаемого к выводу 13 (контур обычно настраивается на высшие гармонические составляющие кварцевого резонатора, чем обеспечивается более высокая тактовая частота).

Максимальная образцовая частота генерации - 27 МГц. Типовая – 18 МГц (18 : 9 = 2 МГц - для КР580ВМ80А).





11	Ф1	Сигнал высокого уровня длительностью 2 периода колебаний задающего генератора для управления МОП-входами
10	Ф2	Вторая фаза – выходной сигнал высокого уровня длительностью в 5 периода колебаний задающего ген-ра для управления МОП-входами
5	SYNC	Синхронизация – вход тактовой последовательности
4	RDY	Выход готовности – выходной сигнал, формируемый микросхемой для управления МПС
1	RES	Выход сброса
7	STB	Выходной импульс строба состояния низкого уровня длительностью 1 период колебаний задающего генератора
6	CLC	То же, что Ф2, но ТТЛ-уровня
12	OSC	Выход генератора гармонических сигналов (задающего)
13	TANK	Вход для подключения внешней времязадающей цепи
14,15	X1,X2	Вход для подключения кварцевого резонатора
3	RDN	Вход готовности – входной сигнал, инициирующий формирование сигнала готовности системы
2	RESN	Вход сброса – сигнал, инициирующий формирование сигнала «сброса»

Л. 9.1. Интерфейс микропроцессора с подсистемами памяти.

Микросхемы памяти. Представляют собой совокупность:

1. Матрицы запоминающих устройств;
2. Устройств управления вводом и выводом;
3. Устройств выбора запоминающих устройств.

Делятся на микросхемы ОЗУ и ПЗУ (RAM и ROM)
RAM – Random Access Memory;
ROM – Read Only Memory.
ПЗУ подразделяются на:

- масочные, программируемые при изготовлении (ROM)
- однократно программируемые пользователем (PROM)
- репрограммируемые ПЗУ:
- – с УФ стиранием ранее записанной информации (EPROM)
- – с электрическим стиранием (EEPROM)

Микросхемы ОЗУ по типу ячеек памяти делятся на:

1. статические (SRAM – Static RAM)
2. динамические (DRAM – Dynamic RAM)

Статические ОЗУ используют электронные схемы с двумя устойчивыми состояниями (триггеры) в качестве запоминающих элементов. В динамических ОЗУ каждый бит представляется в виде наличия (или отсутствия) заряда на конденсаторе, образованном в структуре полупроводникового кристалла. Динамические ОЗУ более просты и ёмки, но требуют постоянной регенерации содержащейся в них информации.

§ Запоминающие элементы ОЗУ

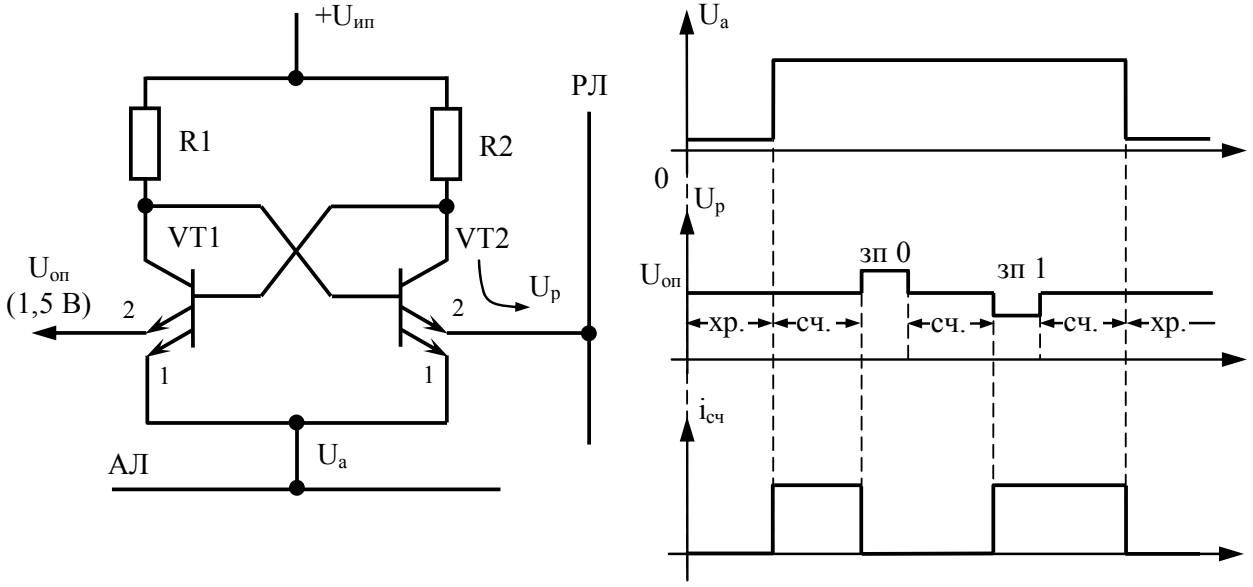
На кристалле каждой микросхемы ЗУ формируются накопитель и схемы обрамления. Накопитель представляет собой регулярную структуру из запоминающих элементов. Схемы обрамления – это дешифратор адреса, схемы управления режимами работы ЗУ, усилители и формирователи сигналов.

В накопитель запоминающие элементы объединяются системой линий, в общем виде содержащей адресную и разрядную линии.

В зависимости от комбинации напряжений на этих линиях запоминающие элементы могут работать в одном из трёх режимов:

1. Хранитель информации
2. Запись
3. Считывание.

На рис.1 приведена схема статического ЗЭ на биполярных транзисторах.



ЯП представляет собой RS-триггер. Элементы 1 транзисторов VT1, VT2 соединены с адресной линией АЛ, потенциал которой в режиме хранения должен быть самый низкий (ближе к нулю) $U_a < (U_{оп} = U_р)$. В этом случае схема находится в одном из устойчивых состояний: открыт VT2 или VT1. Ток утекает по Э1 открытого транзистора, Э2 обоих транзисторов обесточены.

Режим считывания обеспечивается подачей положительного напряжения $U_a > (U_р = U_{оп})$. Пусть в триггер была записана 1, что соответствует отрытому состоянию VT2. В этом случае ток открытого транзистора VT2 перейдёт в цепь Э2 в разрядную линию. Наличие тока в РЛ означает считываемую «1», отсутствие тока – «0».

Режим записи

Запись нуля в ячейку (закрывание транзистора VT2) обеспечивается напряжением $U_р > U_{оп}$ при высоком $U_a > U_р$, запись «1» - $U_р < U_{оп}$ при высоком $U_a > U_{оп}$.

$$P = 0{,}5 \div 1{,}5 \text{ мВт / бит}$$

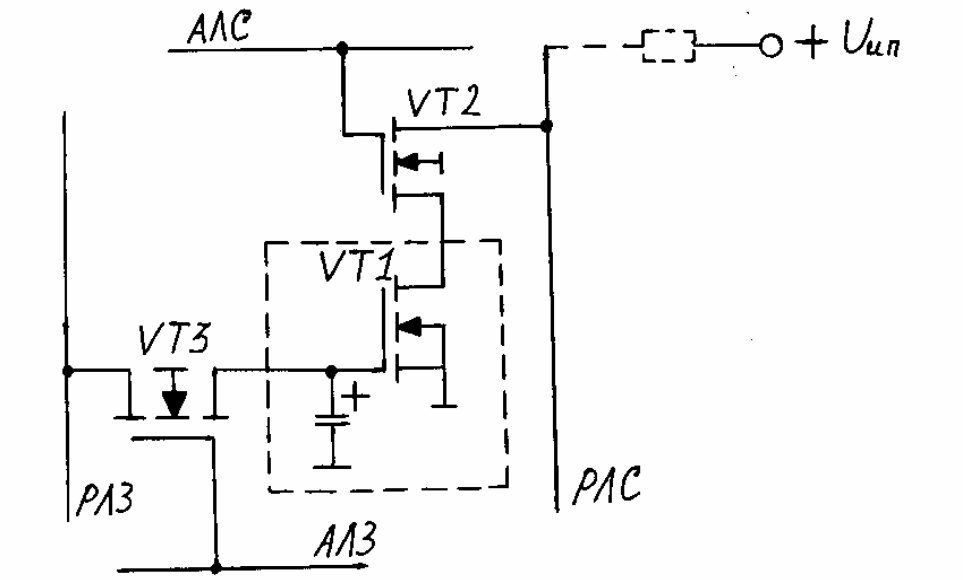
Л.9.1. стр 3-10

На таких ячейках собраны все микросхемы 155 серии (К 155 РУ…)

Триггеры могут выполняться на n - МОП или p – МОП транзисторах (например, 537 серия), в этом случае ток потребления в режиме хранения может быть снижен (0,001 ÷ 5 мА).

Динамические ОЗУ.

На рисунке 2 приведена схема динамической ЯП на n - МОП транзисторах. В ней используется недостаток полевого транзистора – емкость между истоком и затвором.



Информация хранится в виде заряда на конденсаторе, образованным затвором VT1 и его подложкой. VT2 – транзистор считывания, VT3 – записи. При указанной полярности напряжения на конденсаторе VT1 открыт.

В режиме хранения потенциалы адресных линий близки к нулю, транзисторы VT2 и VT3 закрыты и разрядные линии отключены от ЯП. При считывании на АЛС подается высокий уровень и транзистор VT2подключает РЛС к стоку VT1.Если конденсатор заряжен – на РЛС низкий потенциал, и наоборот.

Запись информации происходит при подаче высокого потенциала на АЛЗ, в результате чего VT3 открывается и подключается конденсатор к РЛЗ.

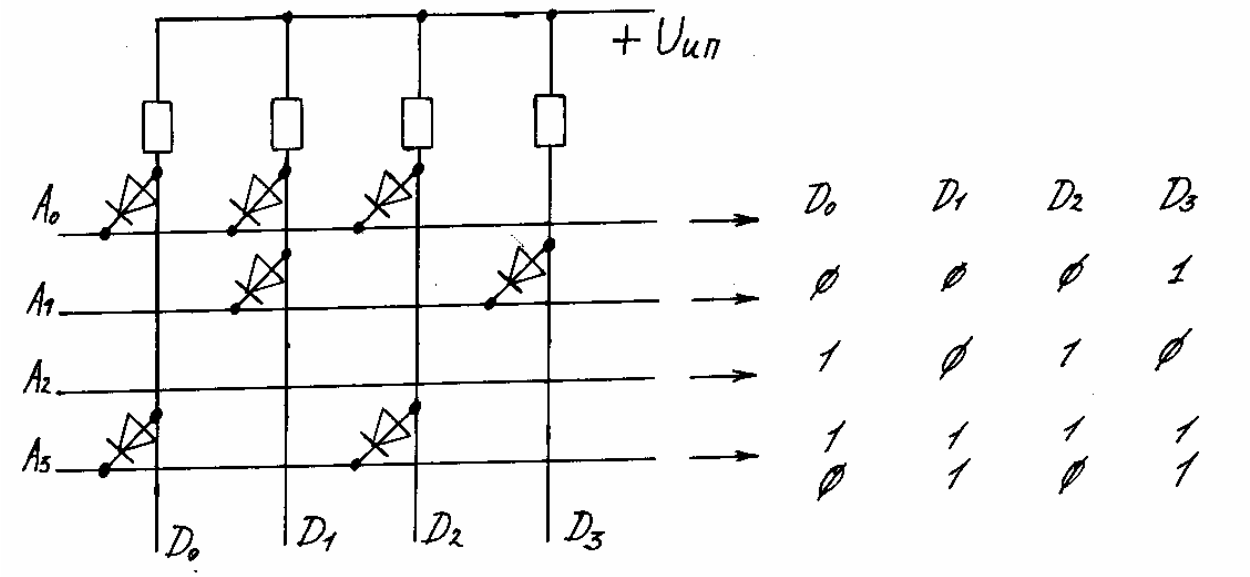
Отечественные динамические ОЗУ представлены в основном серией КР565.

Недостатком динамических ячеек является то, что заряд конденсатора со временем уменьшается. Время хранения заряда зависит от типа ячейки, технологии изготовления, внешних условий и обычно составляет от единиц миллисекунд до нескольких секунд. Для восстановления (регенерации) исчезающей информации, хранимой в ячейке, информационный код шины считывания инвертируется и вновь записывается в ту же ячейку.

Типы ПЗУ.

1. Масочные ПЗУ.

Информация в масочные ПЗУ записывается при изготовлении ПЗУ на заводе в соответствии с фотошаблоном коммутации. Этот фотошаблон выполняется в соответствии с пожеланиями заказчика по картам заказа. В накопителях масочного ПЗУ используются, как правило, диоды или транзисторы, подключенные соответствующим образом к строкам и столбцам накопителя (см. рис.)



(На рисунке D0-D3 сместить вверх)

Наличие или отсутствие элемента в узле пересечения строки и столбца соответствует хранению нуля или единицы в элементе памяти.

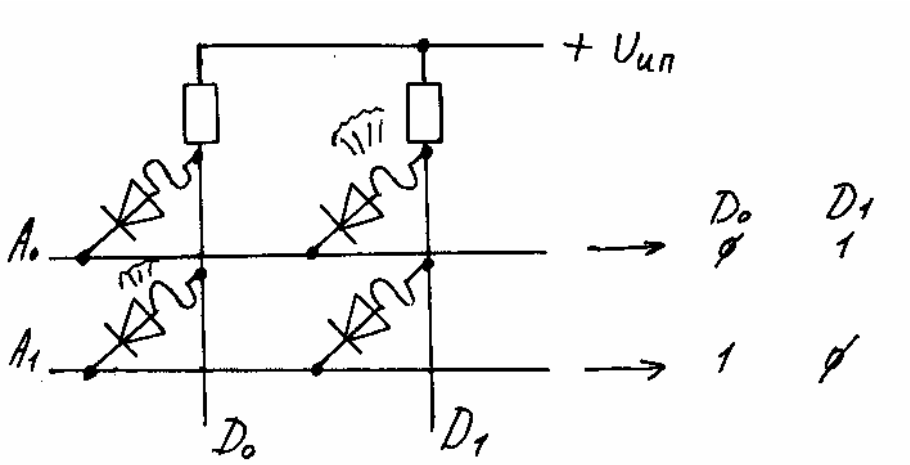
Для заказа МС предприятие – потребитель направляет предприятию - изготовителю гарантийное письмо на выполнение заказа, карту – заказ, форма которой приводится в ТУ на микросхему, перфоленту (дискету), содержащую программу с контрольной суммой; заявку с указанием требуемого числа микросхем (не менее 200).

Примером масочных ПЗУ служат К596, КР1801.

2. Электрические программируемые ПЗУ (PROM).

PROM дают возможность только однократной записи у потребителя путем разрушения элементов структуры ПЗУ с помощью приложенного U или I.

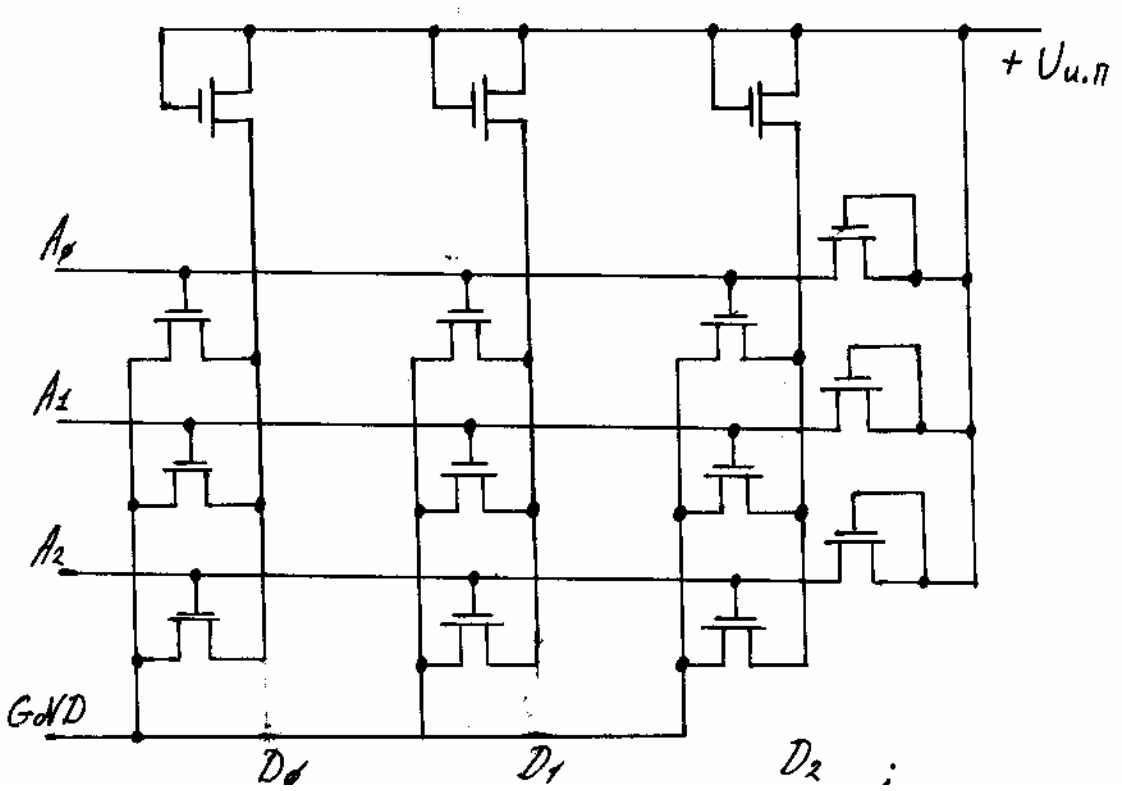
Разрушаемыми элементами могут быть проводящие перемычки из металлической или поликремневой пленки, или даже p-n переходы самих диодов матрицы. Программирование нарисованного ПЗУ заключается в поочередном выборе адреса снова (низким логическим уровнем) и подачей мощных импульсов тока в линии данных, где должны быть единицы (высокий логический уровень).

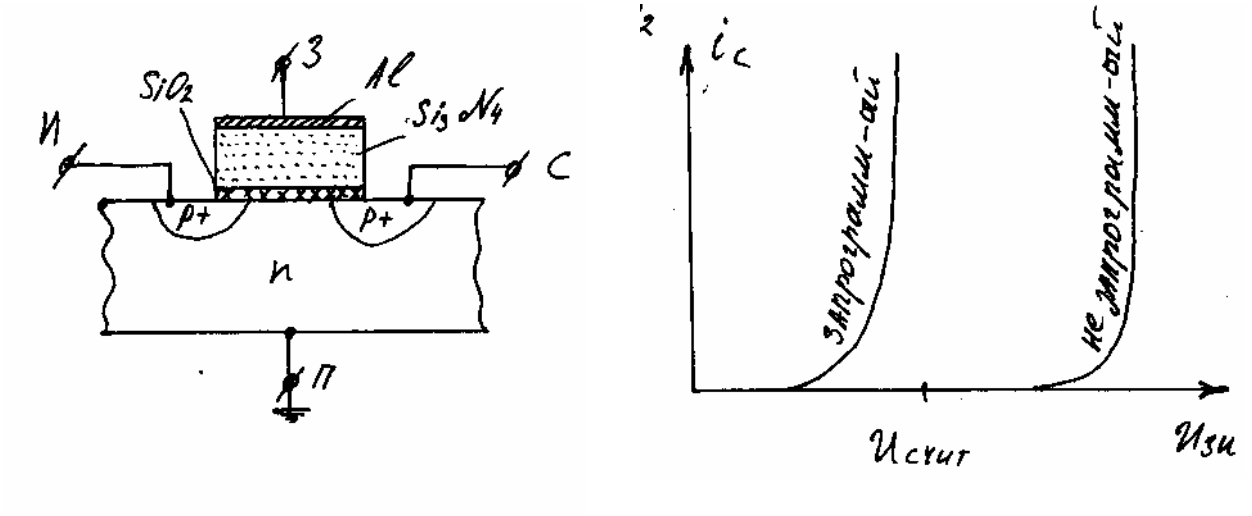


Особенностью таких м/схем является то, что испарившийся металл оседает туда же, где он был испарен, поэтому имеют место случаи восстановления перемычек. Для фиксирования этого предусмотрена электротермотрепировка, которая проводится в течение 168 часов при повышенной температуре в определенном режиме (обычно – чтении всех ячеек). После этого осуществляется контроль записанной информации. Если в процессе контроля обнаружены восстановившиеся перемычки, допускается повторное программирование. Если ошибка обнаруживается повторно – бракуется. Типичный представитель – КР 556 РТ…

3. Перепрограммируемые ПЗУ – EPROM.

Основу таких ПЗУ составляет матрица МОП – транзисторов.

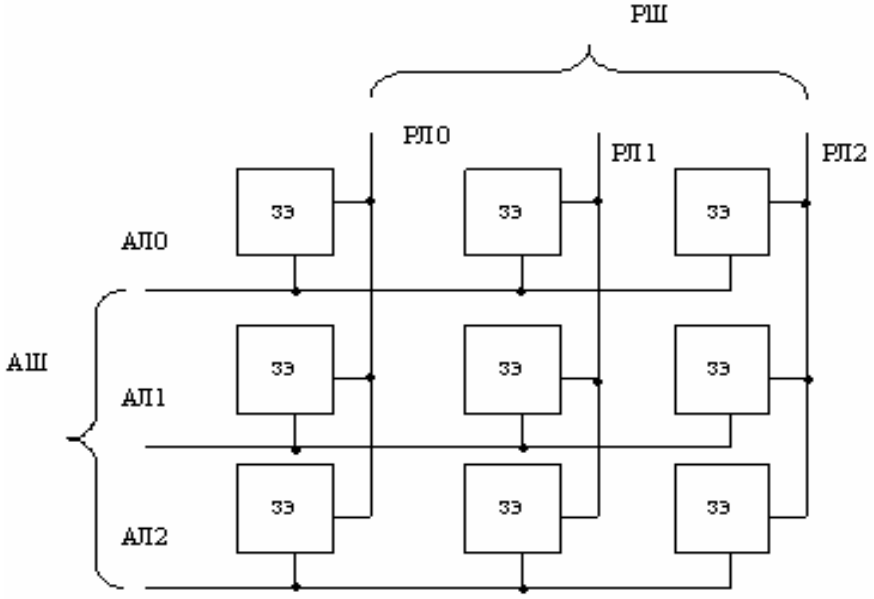




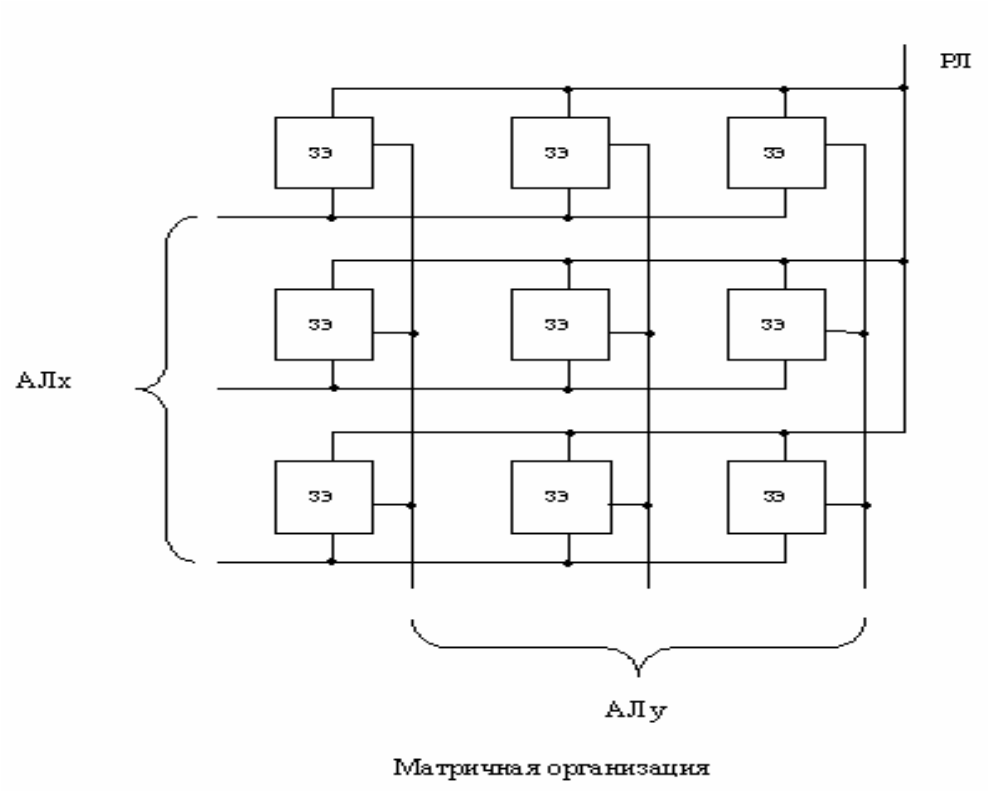
Конструкция транзистора предусматривает возможность подачи на его затвор высокого напряжения. Если такое напряжение подано, транзистор сохраняет низкое сопротивление. Транзистор может быть выведен из этого состояния путем воздействия на него интенсивного УФ излучения порядка 1000 Ангестемм. Время хранения информации составляет сотни тысяч часов (К 537 РФ).

Структуры микросхем памяти.

Отдельные ЗЭ объединяются в накопители двумя способами: матричным и словарным. Словарная организация предусматривает одновременное обращение к нескольким находящимся в строке ЗЭ – одному слову. В накопителе матричного типа обеспечивается обращения к каждому ЗЭ. Выбор нужного ЗЭ обеспечивается выбором двух адресных линий.

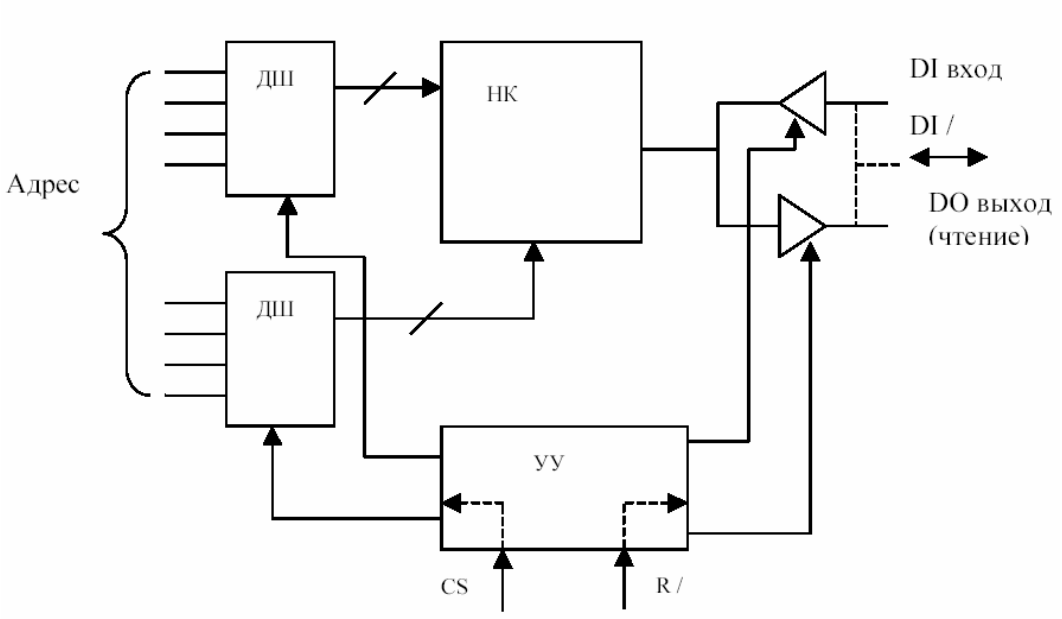


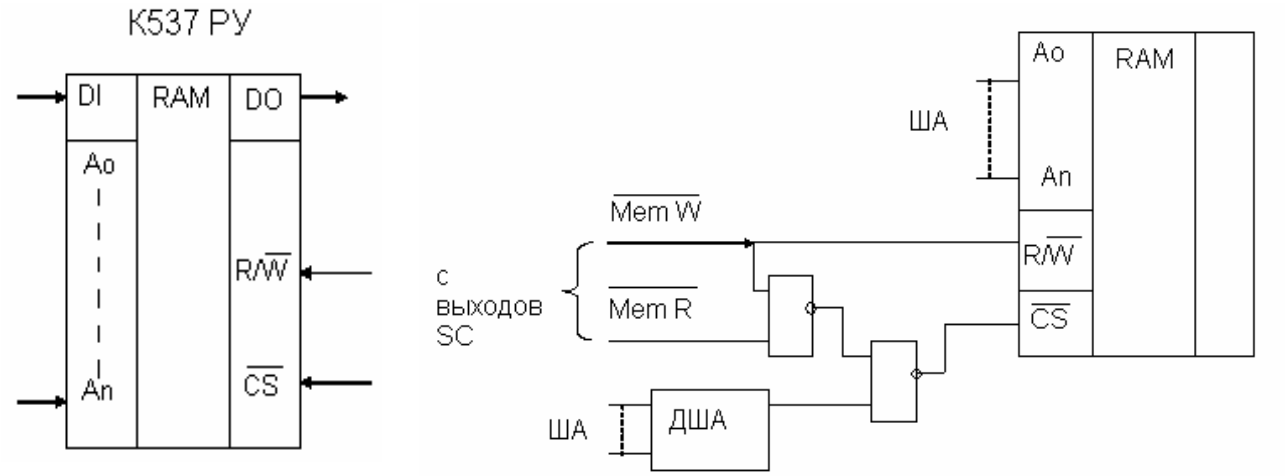
_ словарная организация



Матричная организация

Структура микросхемы статической памяти.





Микросхемы памяти статического ОЗУ содержит:

1. Накопитель НК запомин. элементов;
2. Дешифратор строки ДШх;
3. Дешифратор столбца ДШу;
4. устройство управления УУ;
5. Усилителями записи и считывания.

УУ задает режим работу ЗУ в соответствии с комбинацией сигналов CS и R/WR.

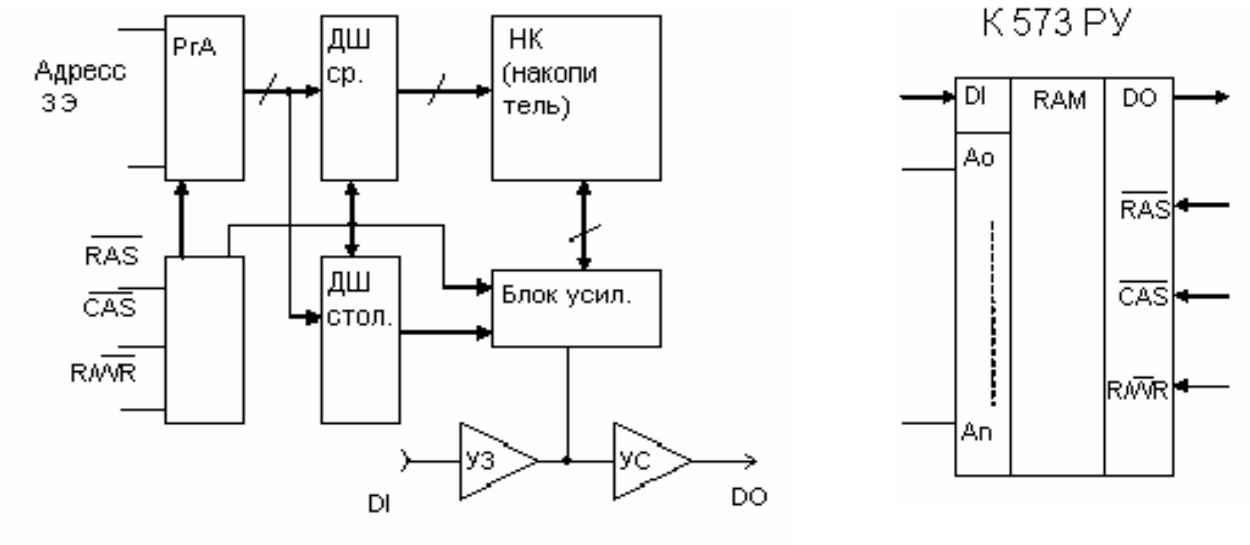
CS=1 соответствует не выбранному устройству. При этом блокируется прием информации со входа DI, а выход D0 переводится в состояние с высоким сопротивлением. Подача сигнала CS = 0 определяет выбор микросхемы для записи или считывания.

Информация со входа DI записывается в адресованную ячейку при сигнале R/W = 0, считывается и направляется на выходе DO по сигналу R/W = 1.

Сигнал CS играет роль синхросигнала, определяющего начало записи или считывания информации. К моменту разрешающего значения этого сигнала должны быть сформированы требуемые значения остальных сигналов (A, DI, DO, R/W).

Динамические ОЗУ.

При построении микросхем ОЗУ с большой емкостью используют динамические запоминающие элементы. Из-за ограничения по числу выводов применяется передача адресной информации по частям: сначала - адрес строки, потом – адрес столбца ЭЗ. Адреса строк и столбцов подаются по одним и тем же выводам микросхемы в два приема.

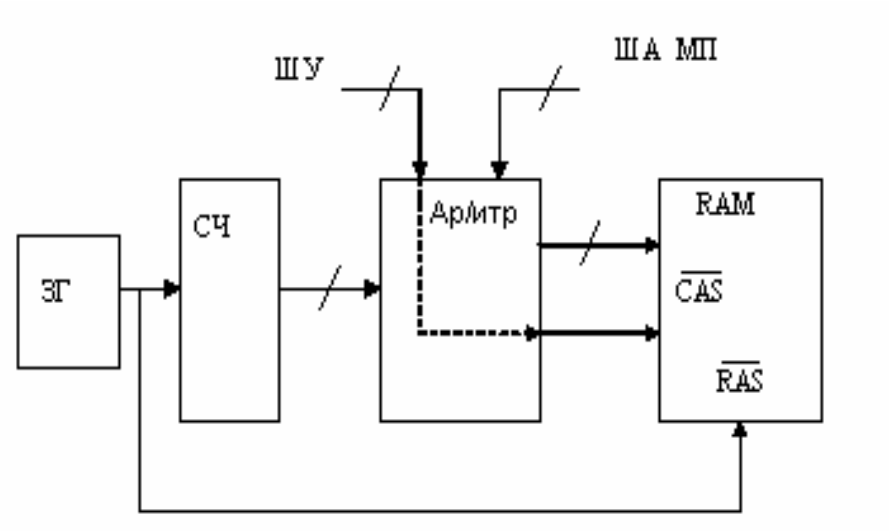


Режимы работы микросхемы задаются комбинацией сигналов RAS, CAS, R/WR. Поступление по ША кода строки фиксируется в регистре адреса РгА по разрешающему значению сигнала RAS = 0 (Row Adress Strobe). При этом дешифратором строки ДШ стр. обеспечивается выбор строки накопителя. При отсутствии сигнала CAS (Column Adress Strobe) производится регенерация информации всей строки. Она представляет передачу информации из всех ЭЗ строк в блок усилителей и последующую запись её обратно в строку. Для обращения к определенному ЗЭ нужно на адресной шине сформировать адрес столбца. Этот код по сигналу CAS = 0 обеспечит выбор одного из усилителей.

Режим записи или считывания будет определяться сигналом R/WR, который присутствует к моменту формирования сигнала CAS = 0. Если R/WR = 1, будет считываться информация из адресного ЗЭ на выход DO. При R/WR = 0 будет произведена запись информации в ЗЭ.

Для регенерации системы динамической памяти используется один из двух способов:

1. Асинхронный. Память имеет собственную аппаратную часть (контроллер), которая поочередно адресуется к строкам памяти. Контролер регенерации предназначен для разрешения конфликтов регенерации и обращения к памяти.



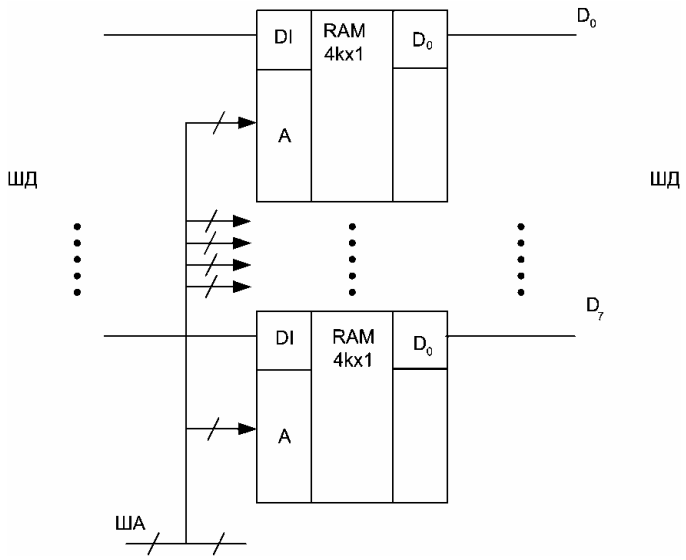
2. Синхронный. Для регенерации выбраны те циклы, в которых процессор не обращается к памяти. Контролер содержит комбинационную схему планировщика регенерации, анализирующего состояние счетчика регенерации и слова состояния МП.

Л9.2 стр 11 -15

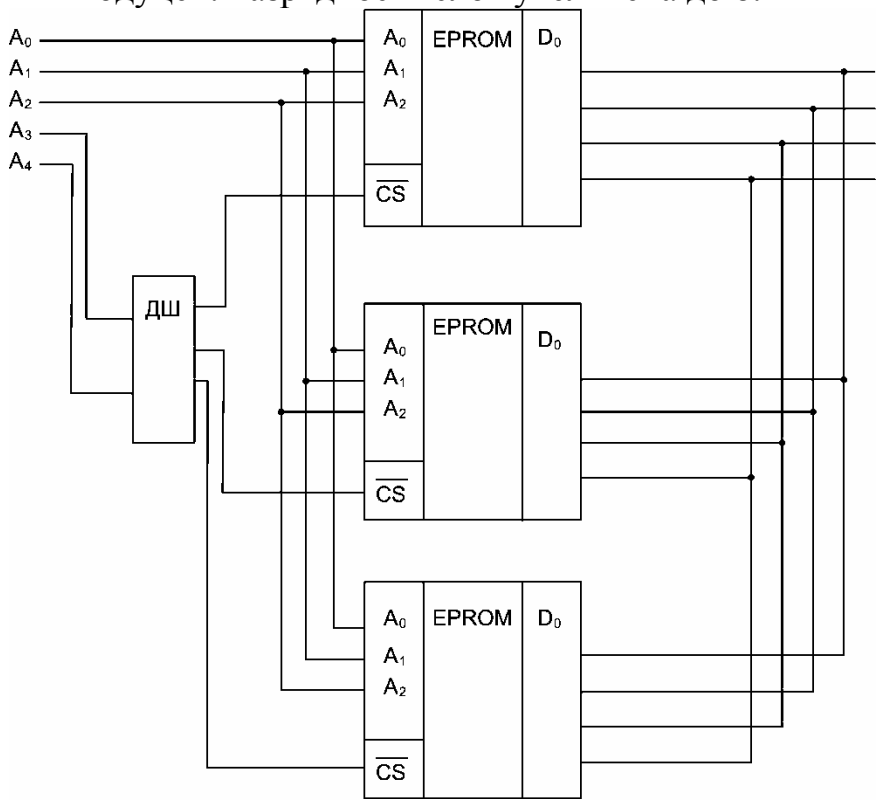
Группирование микросхем памяти.

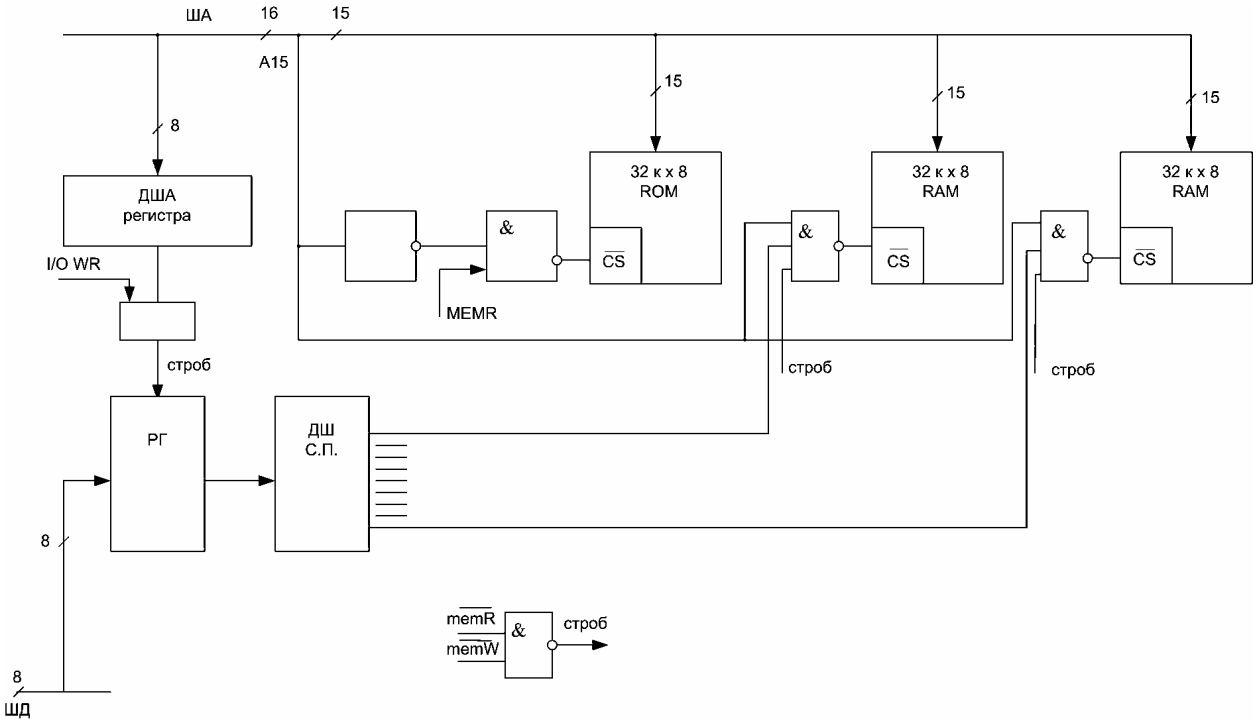
Отдельные микросхемы запоминающих устройств объединяются между собой для 1) увеличения емкости памяти;

2) для увеличения разрядности слов.

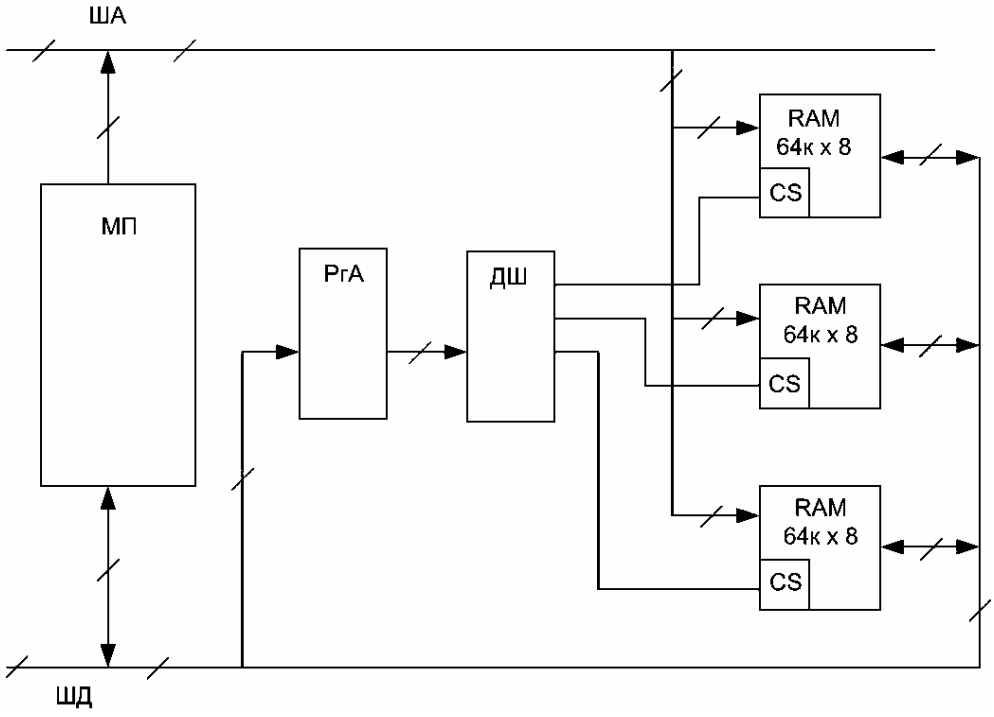


Модуль памяти на 4 килобайта, составленный из 4- килобитных одноразрядных модуцей. Разрядность слов увеличена до 8.





Для увеличения объема памяти свыше объема непосредственно адресуемой, можно использовать принцип страничной организации памяти. В этом случае прямо адресуемая память рассматривается как одна страница.



Для обращения к научной странице используется задание адреса в два этапа:

1. в РгА записывается номер страницы (с ШД)

2. Выбирается ячейка памяти в пределах адресного пространства.

Примечание. Регистр адреса можно рассматривать как один из портов ввода/вывода. При этом выбор нужной страницы будет осуществляться командой OUT РгА.

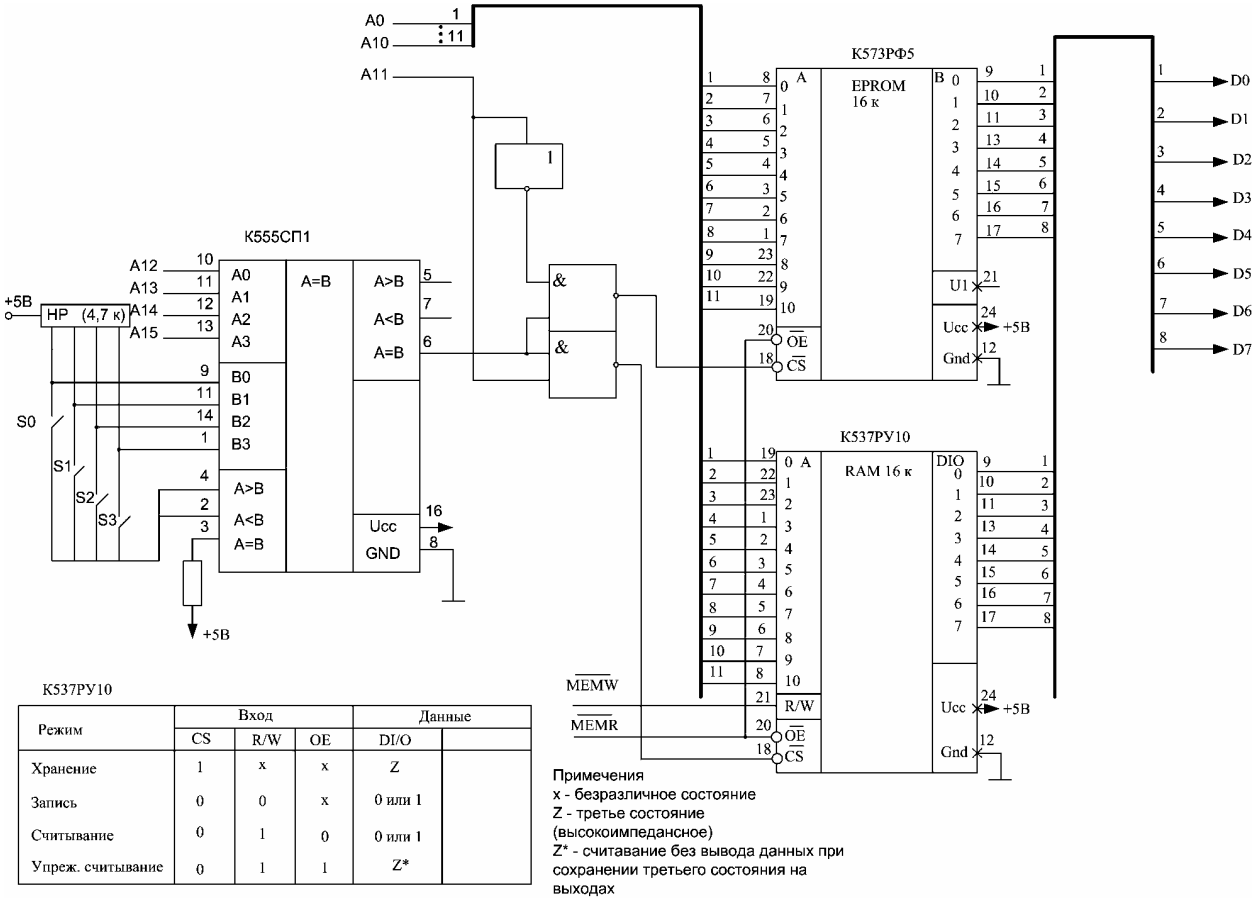
Пример построения модуля памяти.

Для построения модуля памяти с объемом ПЗУ и ОЗУ по 2 килобайта применим ве микросхемы:

К573РФ5 – ПЗУ 2к х 8

К537РУ10 – ОЗУ 2к х 8

Причем младшие адреса закрепим за ПЗУ, а старшие – за ОЗУ. Это объясняется тем, что после включения питания или системного сброса процессор общается за командой к ячейке памяти с нулевым адресом. Поэтому эта память должна быть энергонезависимимой.



К537РУ10

Режим	Вход			Данные	
	CS	R/W	OE	DI/O	
Хранение	1	x	x	Z	
Запись	0	0	x	0 или 1	
Считывание	0	1	0	0 или 1	
Упреж. считывание	0	1	1	Z*	

Примечения
x - безразличное состояние
Z - третье состояние (высокоимпедансное)
Z* - считавание без вывода данных при сохранении третьего состояния на выходах

Рассмотрим работу модуля.

Адресные сигналы A0-A10 подаются непосредственно на адресные входы ИС ПЗУ и ОЗУ. Адресная линия A11 служит для выбора либо первой, либо второй микросхемы. С помощью 11-ти адресных линий A0-A10 можно адресовать любую ячейку в 2к-байтовой области (2^{11}=2к). Сигнал, поступающий по линии A11 позволяет выбрать первую или вторую из этих 2к байтовых областей.

Сигнал A11 проходит через инвертор и схему И-НЕ, после чего подается на вывод « выбор кристалла» (CS) микросхема ПЗУ. Аналогичный сигнал на м/сх ОЗУ поступает без инвертирования через схему И-НЕ. Следовательно, когда один из кристаллов памяти выбран, второй оказывается

не выбран. Сигнал, поступающий на вторые входы упомянутых схем И-НЕ, позволяет блокировать выбор обеих микросхем памяти независимо от значения сигнала A11.

На компараторе К555СП1 собран дешифратор адреса блока памяти. Сигнал высокого уровня на его выходе «A=B» поступает на вторые входы элементов И-НЕ и означает выбор данного 4 КБ блока. Подобных блоков может быть несколько, поскольку объем непосредственно адресуемой памяти МП к580 ВМ80А 64КБ>4КБ. На вход дешифратора подключаются оставшиеся незадействованные 4 адресные линии A12…A15. При совпадении кода на этих линиях с кодом, набранным ключами S0…S3, выбирается данный блок памяти. Очевидно, если блок единственный, ключи S0…S3 должны быть замкнуты все, т.е. задавать код 0000_2.

Чтение из памяти

1) МП помещает на адресные линии информацию о требуемом адресе.
2) МП вырабатывает сигнал MEMR(чтении памяти)
3) Содержимое адресованное ячейки памяти поступает на внешнее выводы кристалла памяти.
4) МП в очередном такте принимает эти данные и приступает к их обработке.

Запись в память

1) МП помещает на адресные линии информацию о требуемом адресе, а на линии данных – информацию, передаваемую на хранение.
2) МП вырабатывает сигнал MEMW(запись в память)
3) Данные записывается в адресованную ячейку памяти.

Л.8.3 Интерфейс микропроцессора с внешними устройствами.

Различают 2 основных типа интерфейса с внешними устройствами:

1. Интерфейс параллельной передачи данных (параллельный интерфейс)
2. Интерфейс последовательный (побитных) передачи данных (последовательный интерфейс)

§1. Параллельный интерфейс.

Вообще говоря, магистраль МП можно считать параллельны интерфейсом. (И это действительно так, если идет речь об обмене данными с ОЗУ и ПЗУ). Однако внешнее устройства (такие как принтер) редко подключаются к магистрали непосредственно. Для этого разработаны специальные схемы (микросхемы) сопряжения и стандарты (протоколы) обмена данными.

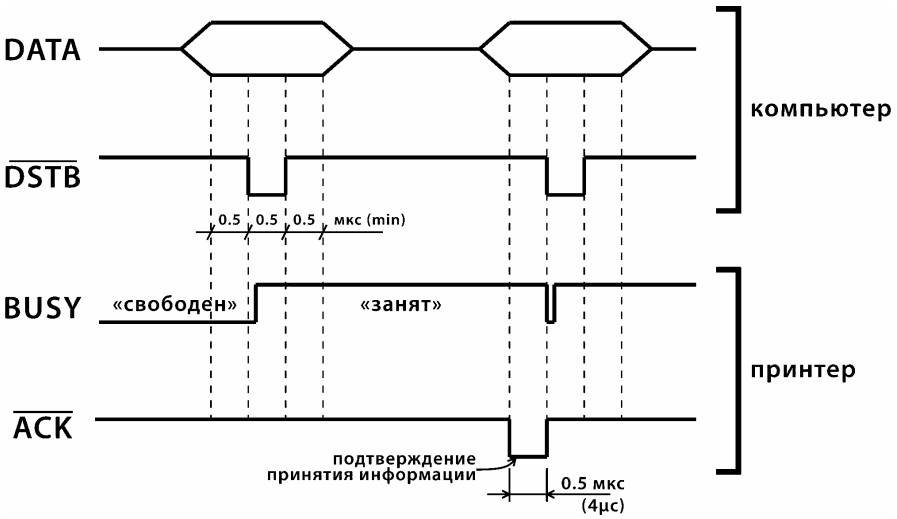
Типичным представителем аппаратной части параллельного интерфейса является программируемый периферийный адаптер (ППА) К580ВВ55А, а также другие устройства (шинные драйверы, буферные регистры).

Наиболее известным стандартом является интерфейс Centronixs, используемый для порта принтера в PC. Есть и другие стандарты (IEE-488, GPIB, HPIB). Все новые устройства разрабатываются для подключения к какому-либо стандартному интерфейсу, а не к магистрали непосредственно.

Интерфейс Centronixs

Основные характеристики:
Скорость передачи данных: 1000 симв/сек
Синхронизация: внешним импульсов строба
Ответ: сигналом BUSY (занято) или ACK (подтверждение)
Логические уровни: ТТЛ-совместимые.

Протокол обмена весьма просто. Передатчик (PC) выставляет данные и выдает стробирущий сигнал, по которому выставленные данные захлопываются буфером приемника (принтера). Одновременно приемник выставляет сигнал BUSY (занят). По окончании обработки выдается короткий сигнал подтверждения ACK и сбрасывается сигнал BUSY.



Назначение выводов на разъеме PC

контакт (25-pin)	Сигнал	Направление (по отн. к PC)	Описание
1	DSTB	выход	Сигнал строба данных
2…9	D0…D7	выход	Данные
10	ACK	вход	Подтверждение (отклик на DSTB)
11	BUSY	вход	Занят
12	PE	вход	Отсутствие бумаги
13	SLCT	вход	Select – выбор устройства (сообщение о готовности “Select” mode)
14	AUTOLF	выход	Автоматический перевод строки и возврат каретки
15	ERROR	вход	Сообщение об ошибке принтера
16	INIT	выход	Инициализация принтера
17	SLCT-IN	выход	Переводит принтер в выбранное состояние (Select mode)
18…25	GND	–	Общий

Интерфейс IEEE-488

IEEE – международное сообщество инженеров электронной техники.

Стандарт IEEE-488 появился в фирме «HP», на базе него разработана интерфейсная шина многоцелевого назначения GPIB. Она позволяет одному устройству управлять 14-ю различными приборами через один интерфейс. В шине используется асинхронная система сигналов квитирования (подтверждения) связей. Поэтому данные можно передавать с той скоростью, которая возможна при работе с тем или иным приемником или передатчиком. Аппаратная реализация шины ограничивает скорость обмена величиной 250 кбайт/с, т.е. 2 Мбита/с. В состав шины входит 16 линий. Через

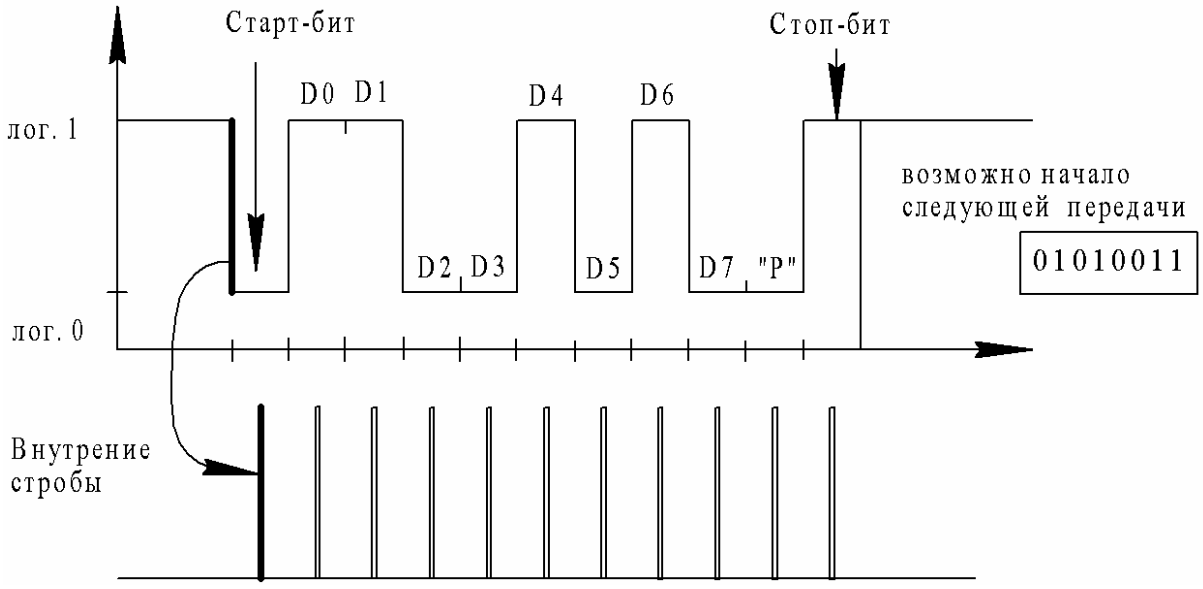
8 линий передаются байты данных или команд, остальные 8 обеспечивают передачу управляющих сигналов. Максимальная длина шины не должна превышать 20 м.

В МП комплекте КР580 для реализации интерфейса IEEE-48 служит м/с КР580ВК91А.

§2 Последовательный интерфейс

В отличие от параллельного, последовательный интерфейс для передачи данных в одну сторону использует только одну сигнальную линию, по которой информационные биты передаются друг за другом последовательно.

При асинхронной передаче каждому байту предшествует “старт-бит”, сигнализирующий приемнику о начале очередной посылки, за которым следует биты данных и, возможно, бит наритета (контроля четкости). Завершает посылку “Стоп-бит”, гарантирующий определенную выдержку между соседними посылками. Старт-бит следующего посланного байта может посылаться в любой момент после окончания стоп-бита, т.е. между передачами возможны паузы произвольной длительности. Старт-бит, имеющий всегда строго определенное значение (лог. 0) обеспечивает простой механизм синхронизации приемника по сигналу от передатчика. Внутренний генератор синхронизации приемника использует счетчик-делитель опорной частоты, обнуляемый в момент приема старт-бита. Этот счетчик генерирует внутренние стробы, по которым приемник фиксирует последующие принимаемый биты.



Очевидно, что для успешного обмена данными, приемник и передатчик должен работать на одной скорости обмена, измеряемой количеством передаваемых бит в секунду.

Для асинхронного режима принят ряд стандартных скоростей обмена: 50, 75, 110, 150, 300, 600, 1200, 2400, 4800, 9600, 19200, 38400, 57600 и 115200 $\frac{бит}{с}$.

Количество бит данных в посылке может составлять 5. 6, 7 или 8 (5 и 6 – редко). Количество стоп-битов может быть 1, 1.5, 2 (1.5 бита подразумевает только длительность стопового интервала).

Асинхронный режим в PC реализуется с помощью COM-порта с использованием протокола RS-232C.

Синхронный режим передачи предполагает постоянную активность канала связи. Посылка начинается с синхробайта, за которым вплотную следует поток информационных бит. Если у передатчика нет данных для передачи, он заполняет паузу непрерывной посылкой байтов синхронизации. Очевидно, что при передаче больших массивов данных накладные расходы на синхронизацию в данном режиме обмена будут ниже, чем в асинхронном. Однако в синхронном режиме необходима внешняя синхронизация приемника с передатчиком, поскольку даже малое отклонение частот приведет к быстро накапливающейся ошибке и искажению принимаемых данных. Внешняя синхронизация возможна либо с помощью отдельной линии для передачи сигнала синхронизации, либо с использованием самосинхронизирующего кодирования данных (например, манчестерский код или NRZ), при котором на приемной стороне из принятого сигнала могут быть выделены и импульсы синхронизации. В любом случае синхронный режим требует либо дорогих линий связи, либо дорогого оконечного оборудования (а может, и того и другого).

Последовательный интерфейс на *физическом уровне* может иметь различные реализации, различающиеся способами передачи электрических сигналов. Существует ряд родственных международных стандартов: RS-232C, RS-423A, RS-422A и RS-485. На рис. 9.11 приведены схемы соединения приемников и передатчиков и показаны их ограничения на длину линии (L) и максимальную скорость передачи данных (V).

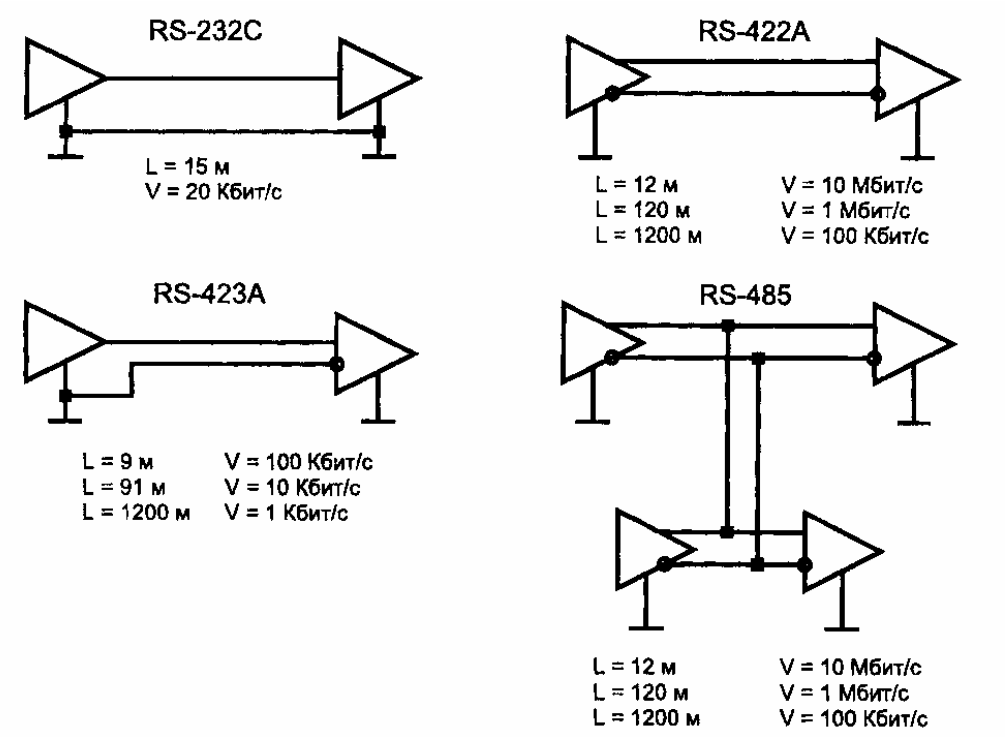


Рис. 9.11. Стандарты последовательного интерфейса

Несимметричные линии интерфейсов RS-232C и RS-423A имеют самую низкую защищенность от синфазной помехи, хотя дифференциальный вход приемника RS-423A несколько смягчает ситуацию. Лучшие параметры имеет двухточечный интерфейс RS-422A и его магистральный (шинный) родственник RS-485, работающие на симметричных линиях связи. В них для передачи каждого сигнала используются дифференциальные сигналы с отдельной (витой) парой проводов.

Наибольшее распространение в PC получил простейший из этих — стандарт RS-232C. В промышленной автоматике широко применяется RS-485, а также RS-422A, встречающийся и в некоторых принтерах. Существуют относительно несложные преобразователи сигналов для согласования всех этих родственных интерфейсов.

Интерфейс RS-232C.

Используемые сокращения:

Русский вариант		Английский вариант	
аббр.	Расшифровка	аббр.	Расшифровка
ООД	Оконечное оборудование данных(аппаратура, принимающая данные)	DTE	Data Terminal Equipment (компьютер, принтер, плоттер, вольтметр, котроллер)
АПД	Аппаратура передачи данных (Аппаратура, передающая данные)		

АКД	Аппаратура канала данных	DCE	Data Communication Equipment (модем)

Интерфейс RS-232C предназначен для подключения аппаратуры, передающей или принимающей данные (*DTE*) к оконечной аппаратуре каналов данных (*DCE)*. Конечной целью подключения является соединение двух устройств DTE, полная схема соединения приведена на рис. 9.12. Интерфейс позволяет исключить канал удаленной связи вместе с парой устройств DCE (модемов), соединив устройства непосредственно с помощью нуль-модемного кабеля (рис. 9.13).

Стандарт описывает управляющие сигналы интерфейса, пересылку данных, электрический интерфейс и типы разъемов. Стандарт описывает асинхронный и синхронный режимы обмена, но *COM-порты поддерживают только асинхронный режим.* Функционально RS-232C эквивалентен стандарту МККТТ V.24/ V.28 и стыку С2, но они имеют различные названия одних и тех же используемых сигналов.

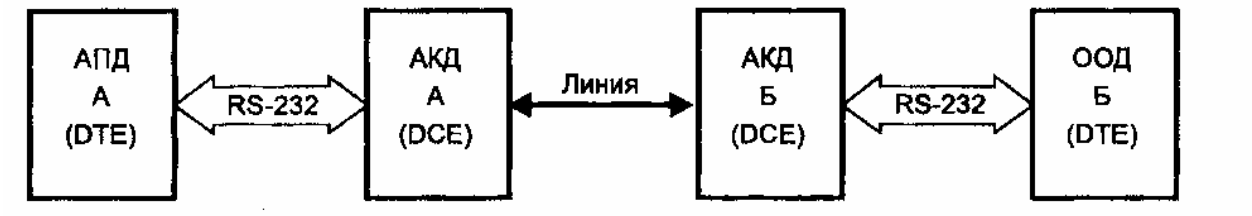


Рис. 9.12. Полная схема соединения по RS-232C

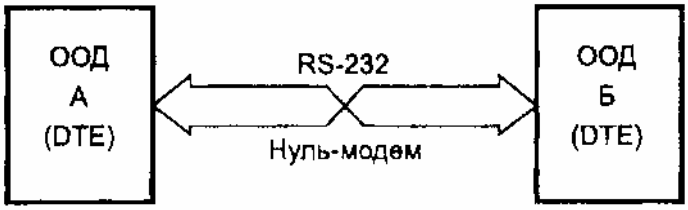


Рис. 9.13. Соединение по RS-232C нуль-модемным кабелем

Электрический интерфейс

Стандарт RS-232C использует несимметричные передатчики и приемники — сигнал передается относительно общего провода — схемной земли . Интерфейс *НЕ ОБЕСПЕЧИВАЕТ ГАЛЬВАНИЧЕСКОЙ РАЗВЯЗКИ* устройств. Логической единице соответствует уровень напряжения на *входе приемника* в диапазоне —12...—3 В. Для линий управляющих сигналов это состояние называется ON («включено»), для линий последовательных данных называется MARK. Логическому нулю соответствует напряжение в диапазоне +3... + 12 В. Для линий управляющих сигналов это состояние называется OFF («выключено»), для линий последовательных данных называется SPACE. Между уровнями -3...+3 В имеется зона

нечувствительности, обусловливающая гистерезис приемника: состояние линии будет считаться измененным только после пересечения соответствующего порога. Уровни сигналов на выходах передатчиков должны быть в диапазонах -12...-5 В и +5... +12 В для представления единицы и нуля соответственно. Разность потенциалов между схемными землями (SG) соединяемых устройств должна быть менее 2 В, при более высокой разности потенциалов возможно неверное восприятие сигналов.

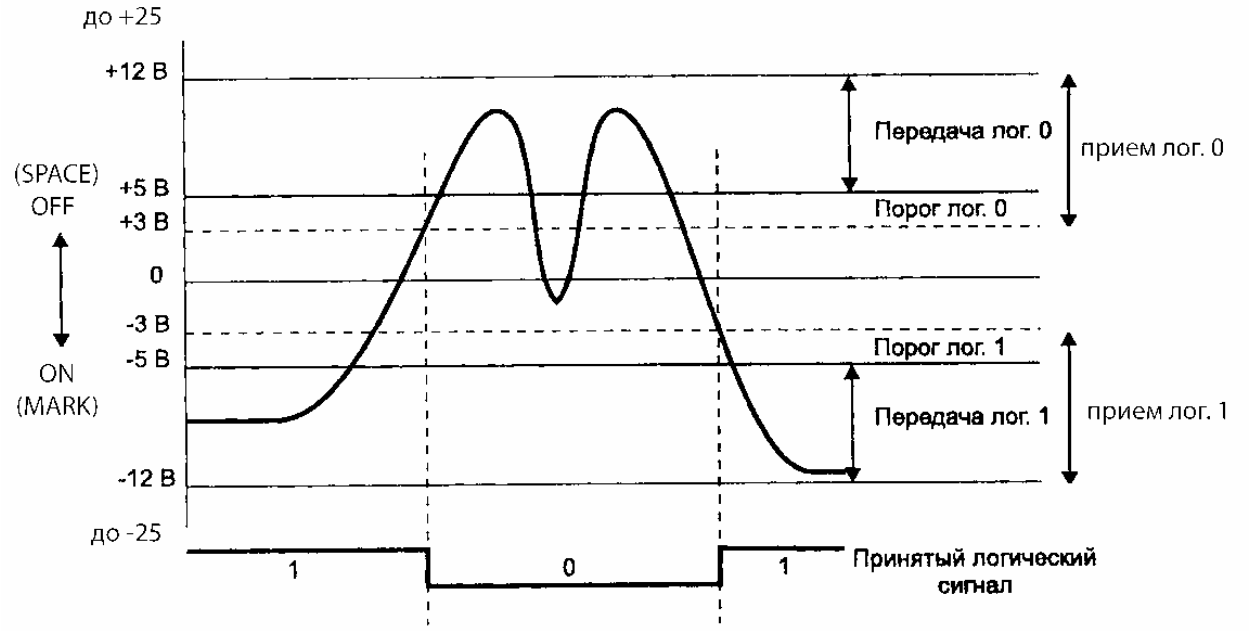


Рис. 9.14. Прием сигналов RS-232C

Интерфейс предполагает наличие *ЗАЩИТНОГО ЗАЗЕМЛЕНИЯ* для соединяемых устройств, если они оба питаются от сети переменного тока и имеют сетевые фильтры (см. главу 11).

Подключение и отключение интерфейсных кабелей устройств с автономным питанием (не питающихся от интерфейса, таких как, например, мышь) должно производиться *при отключении питания.* В противном случае разность не выровненных потенциалов устройств в момент коммутации (присоединения или отсоединения разъема) может оказаться приложенной к выходным или входным (что опаснее) цепям интерфейса и вывести из строя микросхемы.

Стандарт RS-232C регламентирует *типы применяемых разъемов,* что обеспечивает высокий уровень совместимости аппаратуры различных производителей.

На аппаратуре *DTE* (в том числе, и на COM-портах PC) принято устанавливать ***вилки*** (male — «папа») *DB25-P* или более компактный вариант — *DB9-P.*

Девятиштырьковые разъемы не имеют контактов для дополнительных сигналов, необходимых для синхронного режима (в большинстве 25-штырьковых разъемов эти контакты не используются).

На аппаратуре *DCE* (модемах) устанавливают *розетки* (female — «мама») *DB25-S* или *DB-9S.*

Это правило предполагает, что разъемы DCE могут подключаться к разъемам DTE непосредственно (если позволяет геометрия конструктива) или через переходные «прямые» кабели с розеткой и вилкой, у которых контакты соединены «один в один». Переходные кабели могут являться и переходниками с 9 на 25-штырьковые разъемы.

Если на каком-либо устройстве DTE (принтер, плоттер, дигитайзер) установлена розетка — это почти стопроцентный признак того, что к другому устройству (компьютеру) оно должно подключаться прямым кабелем, аналогичным кабелю подключения модема. Розетка устанавливается обычно на тех устройствах, у которых удаленное подключение через модем не предусмотрено (или бессмысленно как, например, у дигитайзера).

Разъемы и сигналы интерфейса RS-232C

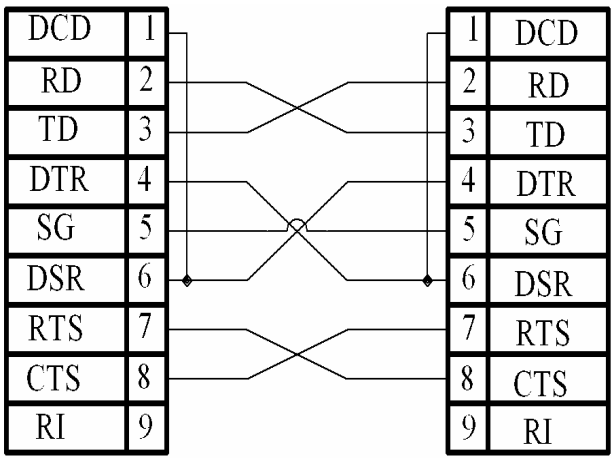
Обозначение цепи		Контакт разъема		Провод шлейфа dыносного разъема PC				Направ-ление	Название цепи
RS232	Стык2	DB25S	DB9S	1*	2*	3*	4*	1/0	
PG	101	1	-	(10)	(10)	(10)	1	-	Protect Ground-Защитная земля
TD	103	2	3	3	5	3	3	0	Transmit Data-Передаваемые данные
RD	104	3	2	2	3	4	5	1	Receive Data-Принимаемые данные
RTS	105	4	7	7	4	8	7	0	Request To Send-Запрос на передачу
CTS	106	5	8	8	6	7	9	1	Clear To Send-Готовность модема к приему данных для передачи
DSR	107	6	6	6	2	9	11	1	Data Set Ready-Готовность модема к работе
SG	102	7	5	5	9	1	13	-	Signal Ground-Схемная земля
DCD	109	8	1	1	1	5	15	1	Data Carrier Detected-Несущая обнаружена
DTR	108/2	20	4	4	7	2	14	0	Data Terminal Ready-Готовность терминала (PC) к работе
RI	125	22	9	9	8	6	18	1	Ring Indicator-Индикатор вызова

1* — шлейф 8-битных мультикарт.

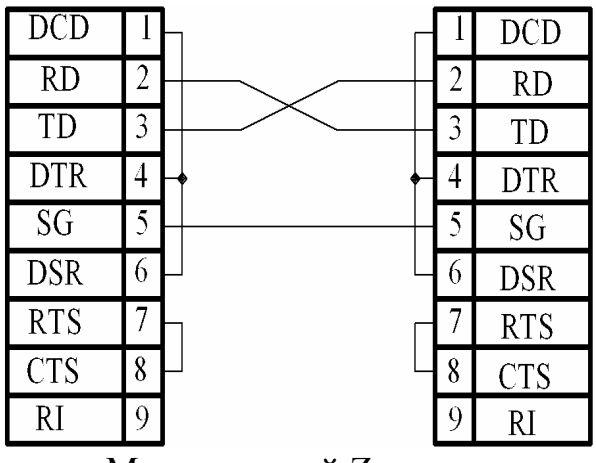
2* — шлейф 16-битных мультикарт и портов на системных платах,

3* — вариант шлейфа портов на системных платах.

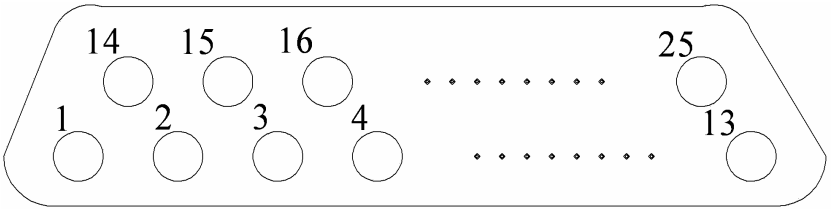
4* — широкий шлейф к 25-контактному разъемую.



Полный нуль – модемный кабель



Минимальный Z-модем



розетка со стороны гнезд
(вилка со тороны пайки)

Назначение сигналов интерфейса RS-232C

Сигнал	Назначение
PG	Защитная земля, соединяется с корпусом устройства и экраном кабеля
SG	Сигнальная (схемная) земля, относительно которой действуют уровни сигналов
TD	Последовательные данные — выход передатчика
RD	Последовательные данные — вход приемника
RTS	Выход запроса передачи данных: состояние «включено» уведомляет модем о наличии у терминала данных для передачи. В полудуплексном режиме используется для управления направлением — состояние «включено» является сигналом модему на переключение в режим передачи
CTS	Вход разрешения терминалу передавать данные. Состояние «выключено» аппаратно запрещает передачу данных. Сигнал используется для аппаратного управления потоками данных
DTR	Выход сигнала готовности терминала к обмену данными. Состояние “включено” поддерживает коммутируемый канал в состоянии соединения
DSR	Вход сигнала готовности от аппаратуры передачи данных (модем в рабочем режиме подключен к каналу и закончил действия по согласованию с аппаратурой на противоположном конце канала)
DCD	Вход сигнала обнаружения несущей удаленного модема
RI	Вход индикатора вызова (звонка). В коммутируемом канале этим сигналом модем сигнализирует о принятии вызова

Если аппаратура DTE соединяется без модемов, то разъемы устройств (вилки) соединяются между собой *нуль-модемным кабелем* (Zero-modem или Z-modem), имеющим на обоих концах розетки, контакты которых соединяются перекрестно по одной из схем, приведенных на рис. 9.17.

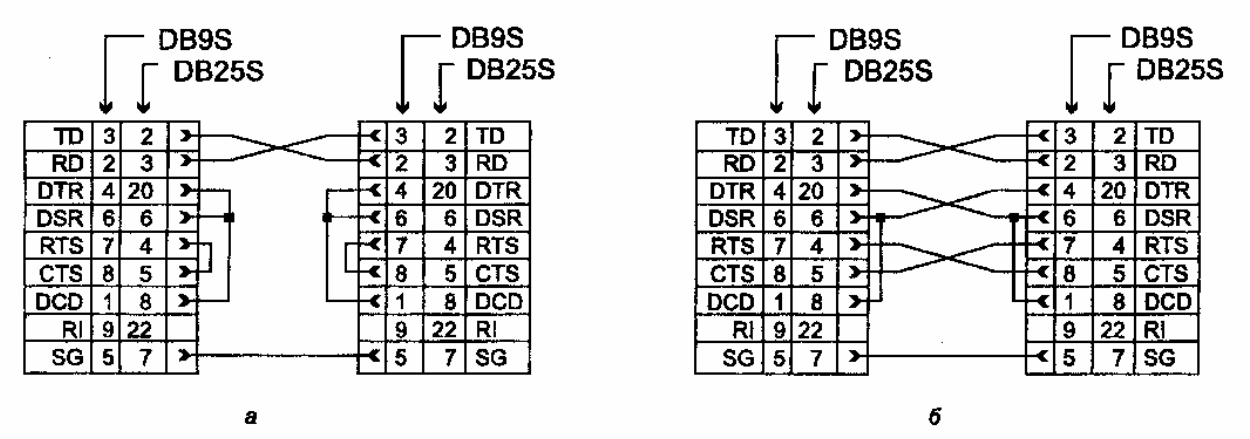


Рис. 9.17. Нуль-модемный кабель: *а* — минимальный, *б* — полный кабель

Управление потоком передачи

Для управления потоком данных (Flow Control) могут использоваться два варианта протокола — аппаратный и программный. Иногда управление потоком путают с квитированием, но это разные методы достижения одной цели — согласования темпа передачи и приема. *Квитирование* (Handshaking) подразумевает посылку уведомления о получении (квитанции) элемента, в то время как *управление потоком* предполагает посылку уведомления о невозможности последующего приема данных.

Аппаратный протокол управления потоком RTS/CTS (Hardware Flow Control) использует сигнал CTS, который позволяет остановить передачу данных, если приемник не готов к их приему. Работу этого протокола иллюстрирует рис.

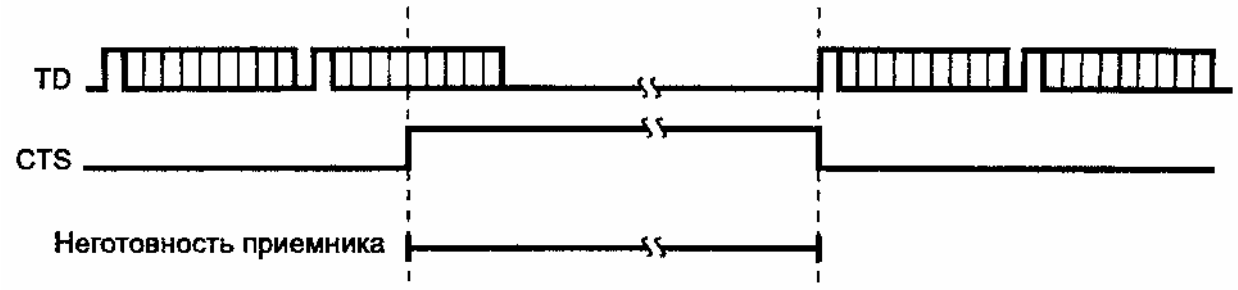


Рис. Аппаратное управление потоком

Передатчик «выпускает» очередной байт только при включенном состоянии линии CTS. Байт, который уже начал передаваться, задержать сигналом CTS невозможно (это гарантирует целостность посылки).

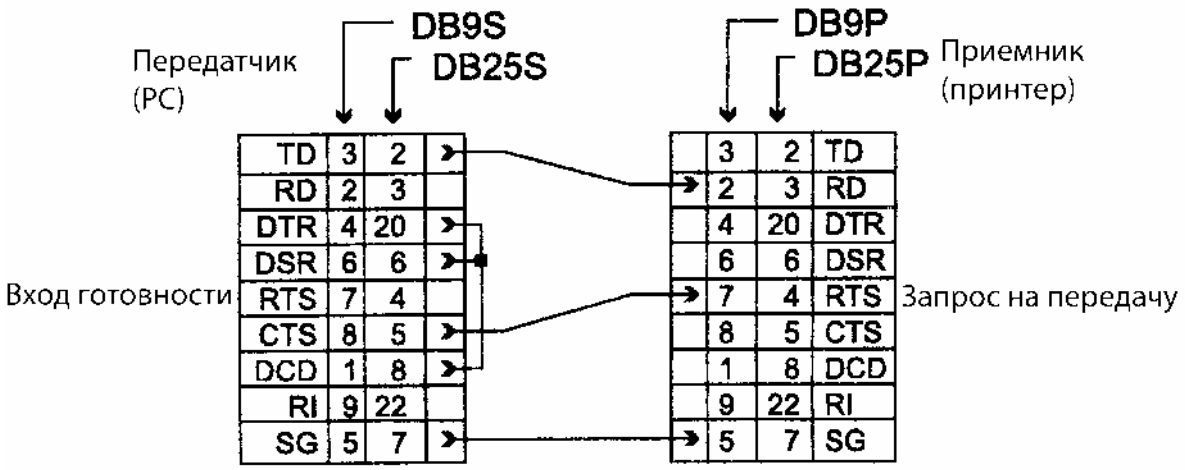


Рис. 9.19. Кабель подключения принтера с протоколом RTS-CTS

Если аппаратный протокол не используется, то при непосредственном соединении у передающего терминала должно быть обеспечено состояние «включено» на линии CTS (обычно соединением собственных линий RTS - CTS). В противном случае передатчик будет упорно молчать (хотя есть программный способ его «разговорить», но им пользуются редко).

Программный протокол управления потоком XON/XOFF предполагает наличие двунаправленного канала передачи данных. Работает он следующим образом: если устройство, принимающее данные, обнаруживает причины, по которым оно не может их дальше принимать, оно по обратному последовательному каналу посылает байт-символ XOFF (13h). Противоположное устройство, приняв этот символ, приостанавливает передачу. Далее, когда принимающее устройство снова становится готовым к приему данных, оно посылает символ XON (11h), приняв который противоположное устройство возобновляет передачу. Время реакции передатчика на изменение состояния приемника по сравнению с аппаратным протоколом увеличивается по крайней мере на время передачи символа (XON или XOFF) плюс время реакции программы передатчика на прием символа (рис. 9.20). Из этого следует, что данные без потерь могут приниматься только приемником, имеющим дополнительный буфер принимаемых данных и сигнализирующим о неготовности заблаговременно (имея в буфере свободное место).

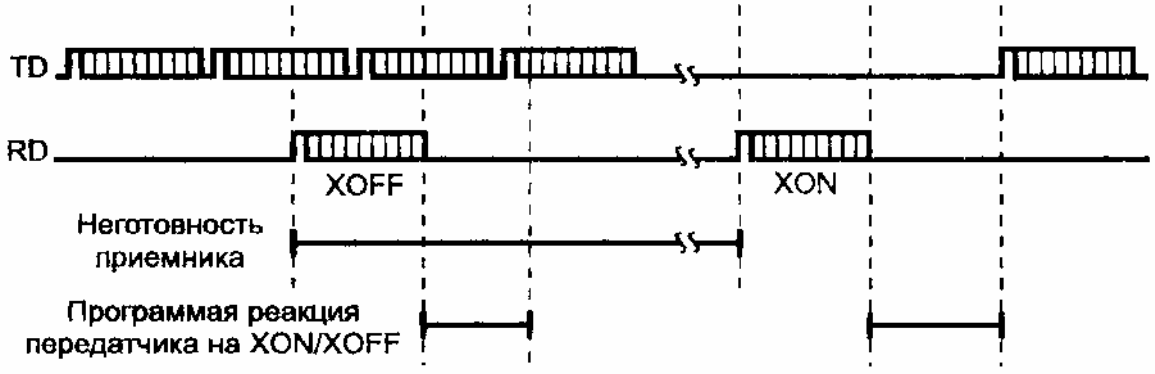


Рис. 9.20. Программное управление потоком XON/XOFF

Преимущество программного протокола при непосредственном соединении устройств заключается в отсутствии необходимости передачи управляющих сигналов интерфейса — минимальный кабель для двустороннего обмена может иметь только 3 провода (см. рис. 9.17, *а)*. Недостатком, кроме требования наличия буфера и большего времени реакции (снижающего и общую производительность канала из-за ожидания прохождения сигнала XON), является сложность реализации полнодуплексного режима обмена. В этом случае из потока принимаемых данных должны выделяться (и обрабатываться) символы управления потоком, что ограничивает набор передаваемых символов. Минимальный вариант кабеля для подключения принтера (плоттера) с протоколом XON/XOFF приведен на рис. 9.21.

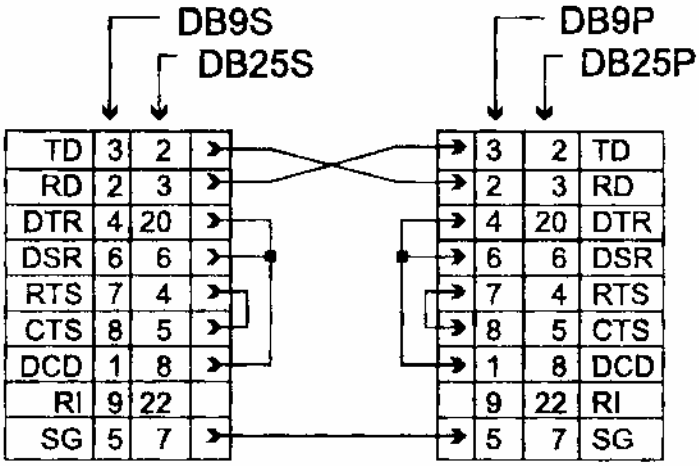
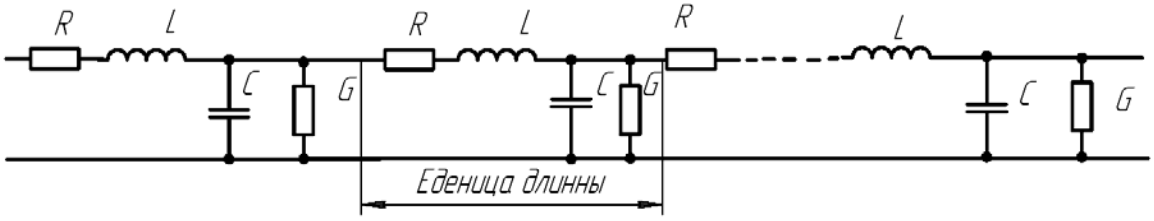


Рис. 9.21. Кабель подключения принтера с протоколом XON/XOFF

Кроме этих двух распространенных стандартных протоколов, поддерживаемых и устройствами, и операционными системами, существуют и некоторые другие. Например, некоторые плоттеры с последовательным интерфейсом используют программное управление, но посылают не стандартные символы XON/XOFF, а *слова* (ASCII-строки). Такой обмен на уровне системной поддержки протокола практически не поддерживается (эти плоттеры рассчитаны на прямой диалог с прикладной программой). Конечно, можно написать и драйвер COM-порта (перехватчик INT14h), но необходимость обработки в нем текстовых сообщений от устройства вывода обычно не вызывает восторга системного программиста. Кабель для подключения такого устройства совпадает с приведенным на рис. 9.21.

Л.8.4. Учёт особенностей линий передачи. Интерфейс «Токовая петля».

По мере того, как растёт длина линии, соединяющий передатчик и приёмник, а также с ростом частоты передачи, кабель линии связи можно рассматривать только как длинную линию и представить моделью показанной на рисунке 1/



Здесь R, L, C и G — удельные сопротивления, индуктивность, ёмкость, и проводимость (на единицу длины)

Перечислим ряд проблем, возникающих в системах обмена данными и связанных с особенностями линии передачи.

1) Отражение. Если нагрузка не согласована с линией, то в линии возникают отражения, которые могут вызвать ошибки при детектировании. Эта проблема решается выбором такой нагрузки, сопротивления которой будет равно характеристическому импедансу линии (волновому сопротивлению).

2) Ослабление. При очень большой длине линии может оказаться весьма значительным ослаблением. Данная проблема устраняется за счёт установки ретрансляторов (или повторителей).

3) Заряд ёмкости. Эквивалентная электрическая ёмкость линии передачи и скорость нарастание напряжения сигнала связано с током, который источник способен выдавать в линию:

$i = C_л \cdot \frac{dU}{dt}$, где Сл полная ёмкость линии:

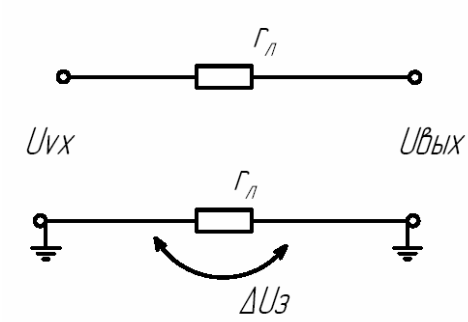
$C_л = C_{уд} \cdot l$;

l-длина линии

Эта характеристика линии имеет очень важное значение, если время нарастания сигнала превышает задержку на распространение сигнала по линии.

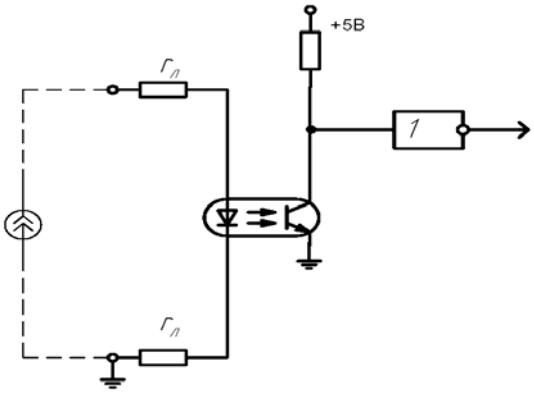
4) Помехи. Повышение частоты следования передаваемых сигналов приводит к появлению перекрёстных помех (за счёт ёмкостной связи различных линий в общем телефонном кабеле). Решение этой проблемы достигается конструкциями линями связи (введение витых пар, экранирование), специальными протоколами связи (избыточное кодирование), а также ограничение скорости передачи.

5) Разный уровень земли



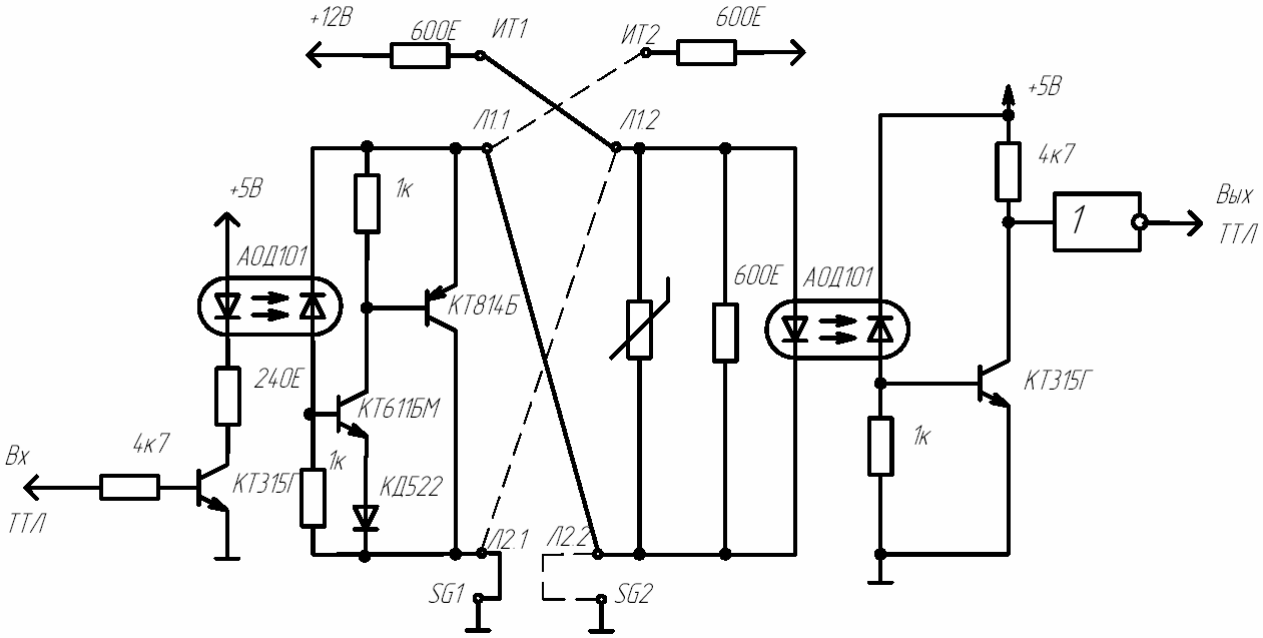
Различия в напряжениях уровня земли на передающих и приёмных концах может вызвать ошибки при детектировании логического уровня сигнала на приемном конце.

Эта проблема решается путём заземления возвратного провода в одной точки гальванической развязкой приёмника и передатчика.



В приведённом примере передавать сигналы лучше не напряжением, а током (в этом случае оказывается не важным падение напряжения на сопротивлении линии $r_Л$)

Такая схема интерфейса называется «токовая петля».



Можно организовать два типа интерфейса «токовая петля»:

а) Активный передатчик — пассивный приёмник. Для этого необходимо соединить между собой проводами клеймы:

ИТ1—Л1.2
Л1.1—Л2.2
Л2.1—SG1

б) Пассивный передатчик — активный приёмник

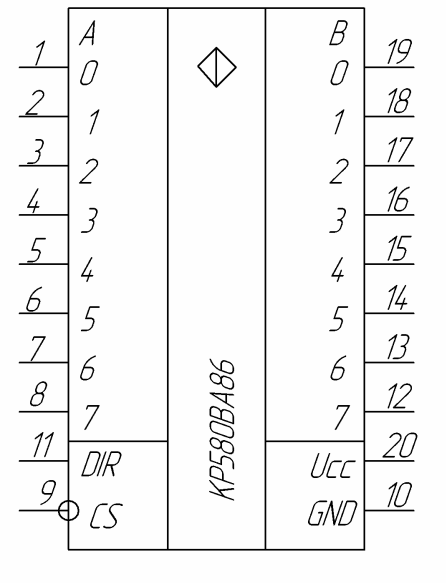
Л1.1— ИТ2
Л2.1— Л1.2
Л2.2—SG2

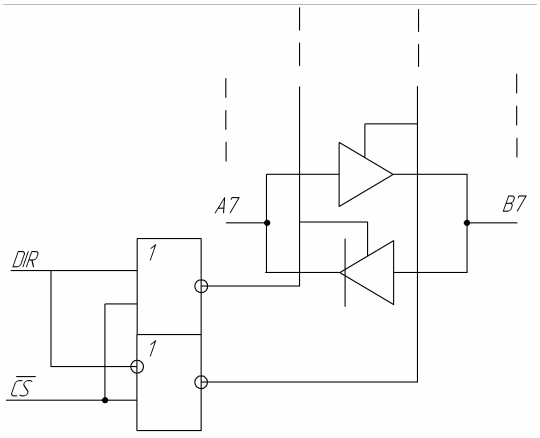
Источником тока в данных схемах является источник +12В с последовательно включенным резистором 600 Ом. Ток короткого замыкания I_{ON}=12/600=20мА.

Л. 10.1. Элементы МПС.

§1 Шинные формирователи (шинные драйверы).

К шинам адреса и данных МП можно подключать около 10 устройств на КМОП-схемах или 1 ТТЛ-микросхему. Если шина имеет протяженность, превышающую 0,3 м, или к ней подключены несколько ТТЛ-устройств, то возникают перегрузки выходных каскадов МП. Для их устранения используют двунаправленные шинные формирователи, предназначенные для обмена данными между МП и системной шиной. В МПК КР580 шинные формирователи представлены микросхемами: КР 580 ВА86 – формирователь без инверсии с тремя состояниями на выходе; КР 580 ВА87 – формирователь с инверсией с тремя состояниями на выходе.





Обозначение вывода	Тип	Функциональное назначение
A0…A7	вх/вых	Информационная шина с малой нагрузочной способностью
B0…B7	вх/вых	Информационная шина с большой нагрузочной способностью
$\overline{CS}$	вх	Разрешение передачи (управление третьим состоянием)
DIR	вх	Направление передачи
GND	-	Общий
U_{cc}	-	Напряжение питания +5 В

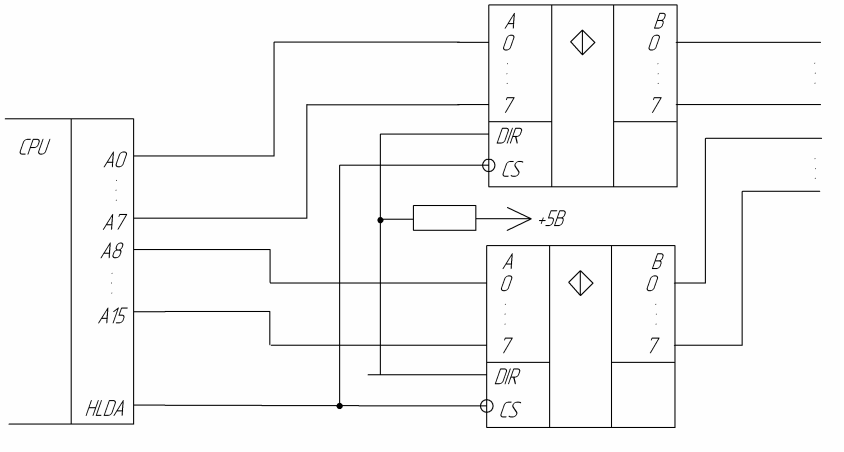
Распиновка микросхемы сверена со справочником.

Микросхема состоит из 8 одинаковых блоков разнонаправленных усилителей-формирователей и схемы управления. В зависимости от сочетания сигналов $\overline{CS}$ и DIR возможны следующие режимы работы:

$\overline{CS}$	DIR	Режим
0	1	передача в направлении A→B
0	0	передача в направлении B→A
1	x	третье(высокоимпедансное состояние)

x – безразличное состояние

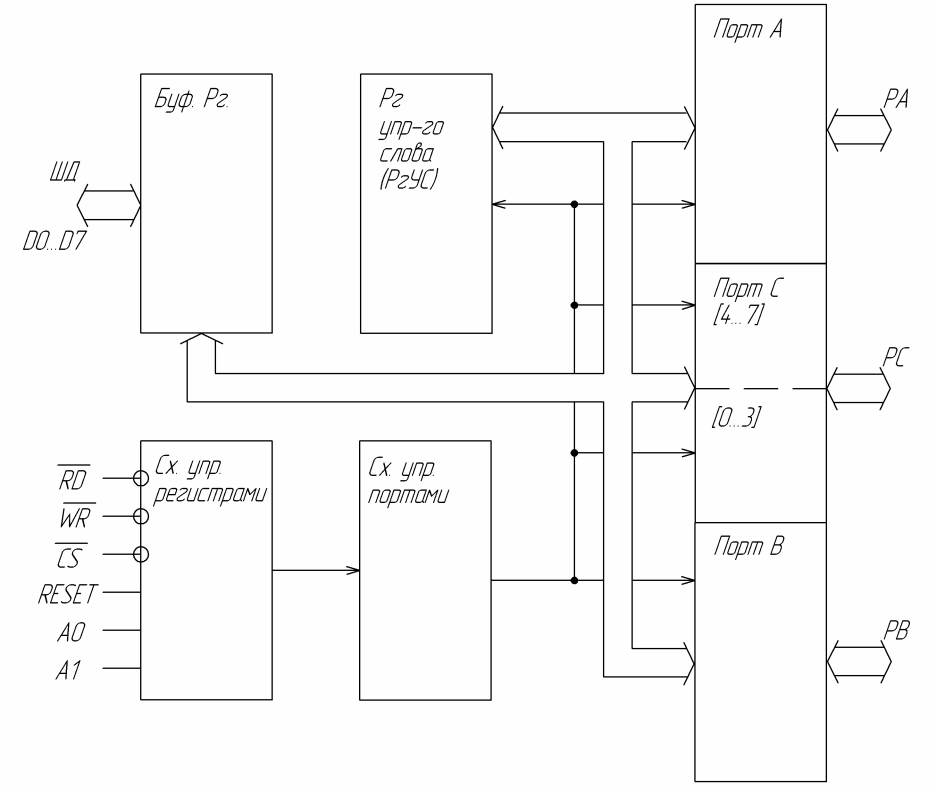
Пример формирования системной шины адреса.



§2 Программируемый периферийный адаптер (ППА) КР580 ВВ55А (параллельный интерфейс).

ППА относится к средствам, обеспечивающим сопряжение периферийных устройств и МПС. ППА является программируемым периферийным модулем.

Структурная схема ППА приведена на следующем рисунке.



Подключение периферийного оборудования осуществляется через три 8-битных порта A,B и C; интерфейс с МПС осуществляется с помощью 14 линий:

D0…D7 –шина данных;

A0…A1 – адреса выбора порта;

$\overline{\text{CS}}$ - выборка кристалла (корпуса);

$\overline{\text{RD}}$ - вход считывания информации из регистра порта на ШД;

$\overline{WR}$ - вход записи данных в адресный регистр ШД;

RESET – сигнал системного сброса, по которому РгУС обнуляется, все порты переводятся в режим ввода.

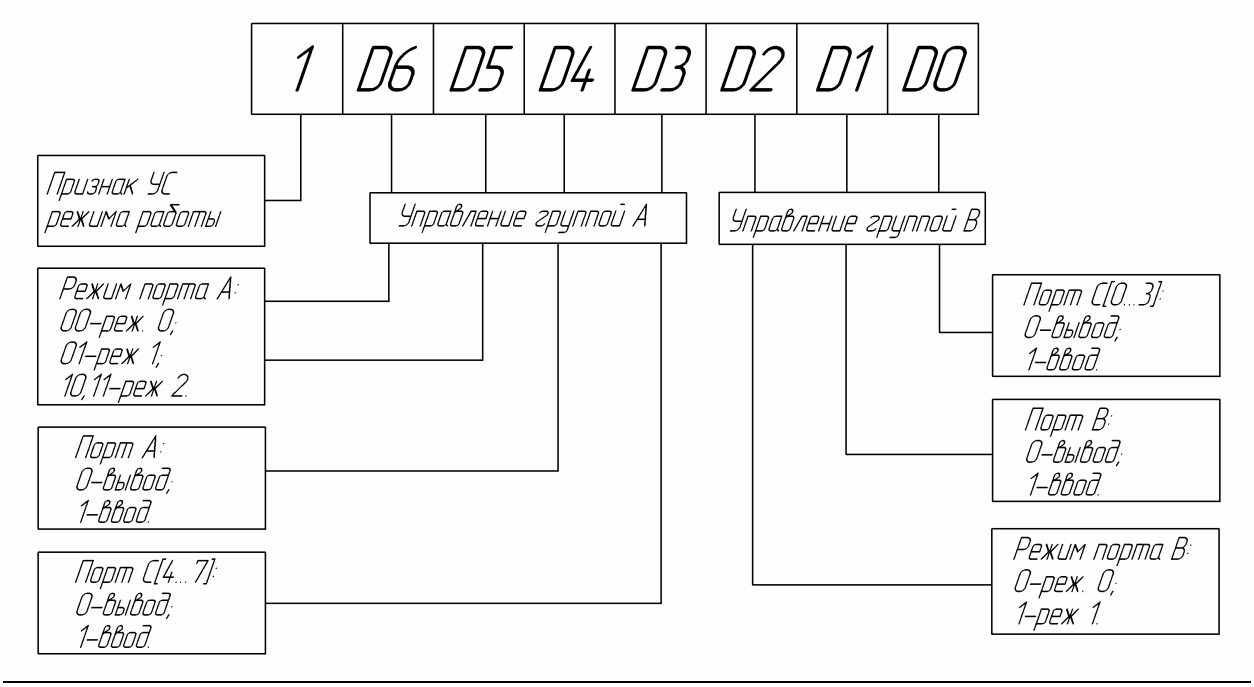
Программирование и обмен данными осуществляется командами ввода-вывода IN и OUT (как правило), при выполнении которых в разрядах ША, соединенных линиями АФ, A1 микросхемы обеспечивается нужная комбинация сигналов (выводы $\overline{\text{RD}}$ и $\overline{WR}$ подключаются к соответствующим выходам системного контроллера $\overline{\text{RD}I/O}$ и $\overline{\text{WRI/O}}$).

Основные режимы работы ППА сведены в таблицу.

	A1	A0	$\overline{\text{RD}}$	$\overline{WR}$	$\overline{\text{CS}}$	Режим
Чтение	0	0	0	1	0	Порт А→ШД
	0	1	0	1	0	Порт В→ШД
	1	0	0	1	0	Порт С→ШД
Запись	0	0	1	0	0	ШД→ПА
	0	1	1	0	0	ШД→ПВ
	1	0	1	0	0	ШД→ПС
	1	1	1	0	0	ШД→РгУС
Третье	x	x	x	x	1	
состояние	x	x	1	1	0	
	1	1	0	1	0	Запрещенная запись

Программирование адаптера заключается в загрузке управляющего слова в РгУС.

Формат управляющего слова.



Адаптер имеет три основных режима:

Режим 0 – основной режим ввода-вывода информации;

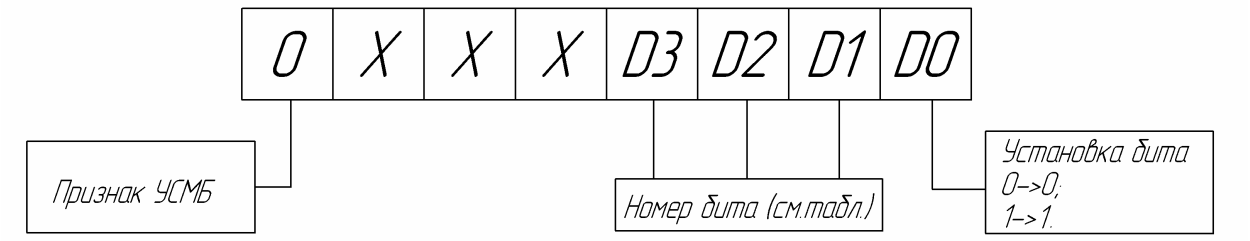
1 – режим стробируемого ввода-вывода информации;

2 – режим двунаправленной шины;

Любой из режимов может быть выбран в ходе выполнения программы загрузкой УС по команде ООТ (например).

Особенностью порта С является то, что он предоставляет возможность побитного изменения своего содержимого путем загрузки в РгУС управляющего слова манипуляции с битами (УСМБ).

Формат УСМБ:



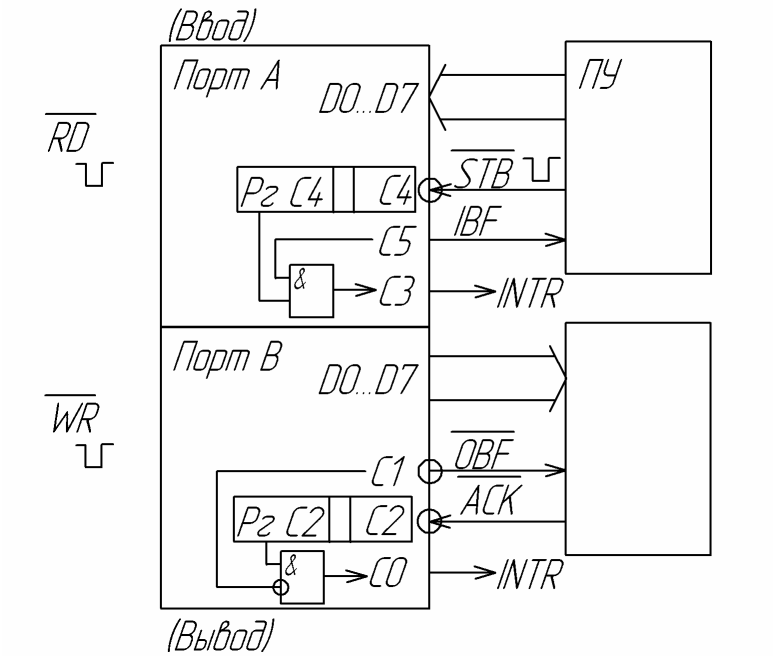
Код D1;D2;D3	№ бита
000 001 … 111	0 1 … 7

Описание режимов работы ППА.

Режим 0. Могут работать все три порта, причем С разделяется на два независимых 4-х битных порта. Выводимые данные фиксируются в

регистрах – защелках, входящих в состав всех портов. Вводимые данные не защелкиваются, т.е. в операции считывания из порта АКК загружается текущее состояние входных линий порта.

Режим 1. Порты А и В образуют каналы ввода или вывода, а порт С используется для накопления и обработки сигналов управления обменом. Т.е. линии порта С уже не могут использоваться для обмена данными (по крайней мере, те, которые работают с портом, включенным в режиме 1).

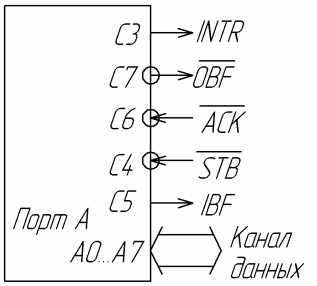


Ввод данных в режиме 1. При поступлении управляющего сигнала $\overline{\text{STB}}$ в буфер загружаются данные, одновременно формируется сигнал о готовности буфера IBF (Input Buffer Full). Если УСМБ 4 бит установлен в 1, то по окончанию $\overline{\text{STB}}$ формируется сигнал запроса, от МПС поступает сигнал чтения $\overline{\text{RD}}$, по переднему спаду которого гасится бит разрешения прерываний и сигнал INTR, а по заднему – сигнал готовности буфера IBF.

Вывод данных в режиме 1. При загрузке порта устанавливается сигнал о готовности выходного буфера $\overline{\text{OBF}}$ (Output Buffer Full). Этот сигнал формируется после завершения сигнала $\overline{\text{WR}}$ и оканчивается по сигналу $\overline{\text{ACK}}$ периферии, подтверждающей восприятие данных. Установкой второго бита порта С в «1» разрешается прерывание. Сигнал INTR устанавливается при $\overline{\text{ACK}}$=1 и $\overline{\text{OBF}}$=1 (т.е. ПУ считало информацию и отправило «квитанцию» в получении) и сбрасывается сигналом $\overline{\text{WR}}$.

Линии запроса прерываний могут и не использоваться. В этом случае программа должна анализировать состояние линий IBF, $\overline{\text{OBF}}$ путем чтения соответствующих бит порта С перед выполнением операций записи или чтения ПУ. В случаен использования линий INTR необходимо всякий раз после обработки прерывания «разрешать» его снова установкой бит 4 и 2 порта С в 1.

Режим 2. (Двунаправленная шина). В этом режиме может работать только порт А. Порт С в этом случае обеспечивает управление потоками данных. Порт В – свободен и может использоваться в режиме 0.



Порядок управления данными и управляющими сигналами как и выше.

В качестве примера использования ППА приведем схему контроллера клавиатуры и дисплея на 9-разрядном семисегментном индикаторе АЛС 318.

В данной схеме ППА запрограммирован следующим образом: порт А, В и $С_{верхн.}$ (4…7) – режим 0, только на вывод; порт $С_{нижн.}$ (0…3) – (режим 0) только ввод. Порт А и $С_{верхн.}$ образуют регистр сканирования, порт В – регистр сегментов. 555ЛН1 и 555АП5 – буферные элементы (для увеличения нагрузочной способности).

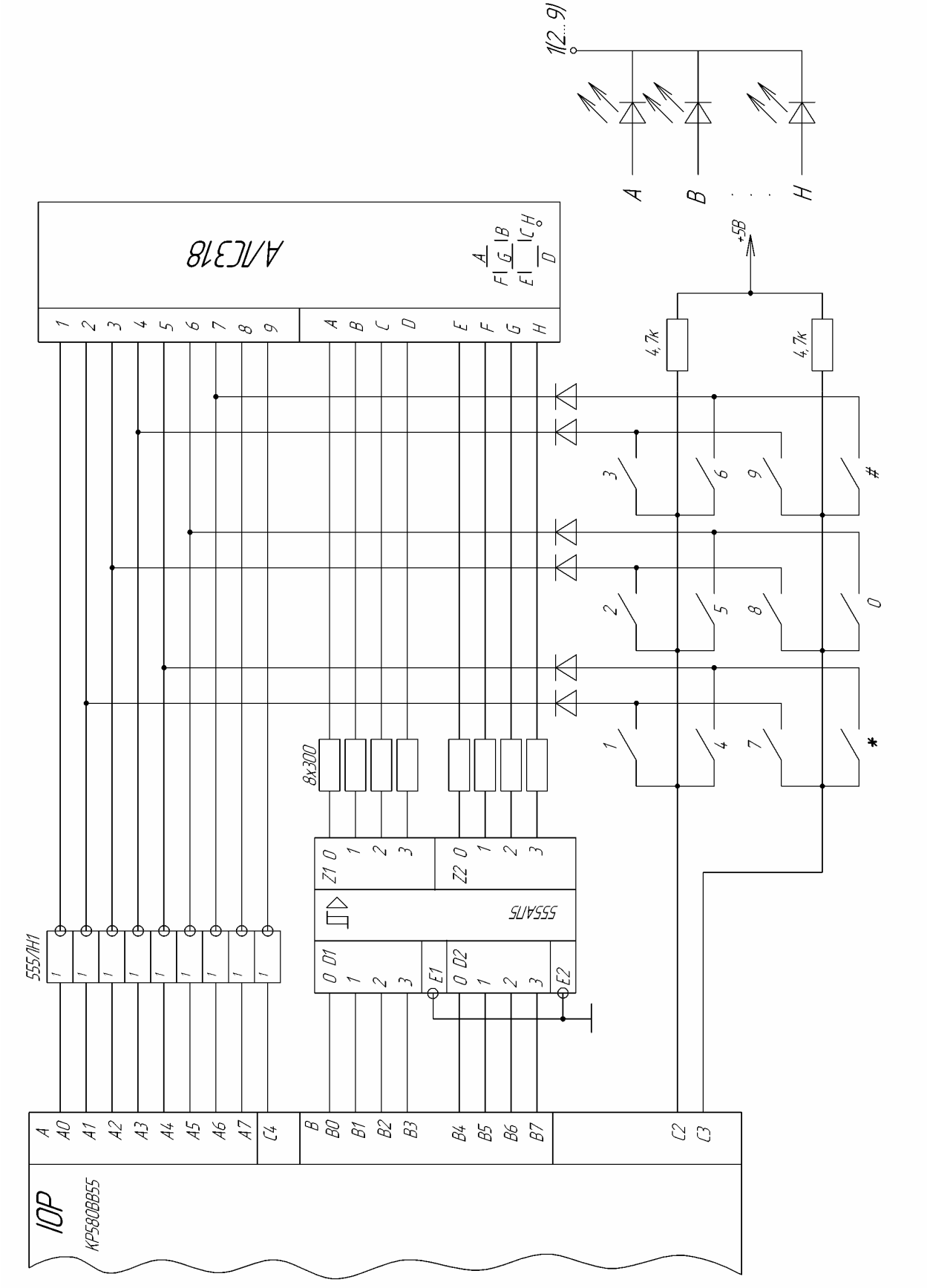
Схема индикации и клавиатуры работает в динамическом режиме, т.е. в каждый момент времени работает только один из девяти семисегментных индикаторов светодиодной матрицы АЛС318. Эффект непрерывного свечения индикаторов обеспечивается многократным повторением вывода символа и световой инерцией глаза наблюдателя (приблизительно 0,018 с). Организация цикла периодического высвечивания символа производится следующим образом:

1) В порт «В» записывается код в соответствие с требуемым изображением символа. Например, для вывода буквы **С** необходимо зажечь сегменты A,D,E,F, следовательно, единицы должны быть записаны в разряды B0, B3, B4, B5. В остальных разрядах должны быть нули (00111001=39h).

2) В порт А и $С_{верхн.}$ Записывается код в соответствии с требуемым номером разряда индикатора, который должен высвечивать букву «С». Например, если нужно высветить символ во второй позиции, то в разряд А1 должна быть записана «1», а в остальные разряды порта А и $С_{верхн.}$ – нули. Единица с выхода А1 ППА инвертируется м/схемой 155ЛН1 в результате чего на вторую позицию АЛС318 подается потенциал близкий к нулю, что вызывает свечение в данной позиции.

3) Производится чтение порта $С_{нижн.}$ (во время горения одного из разрядов индикатора). Если нажата клавиша «1» или «7», то разряды С2 и С3 будут восприняты как 0,1 или 1,0, что является сканирующим кодом этих клавиш при опросе второй позиции индикатора. Эти же коды при опросе, например, четвертой позиции индикатора будут являться скан-кодами клавиш 3 и 9. Нажатие других клавиш не приведет к изменению потенциалов на входах С2 и С3 и будет воспринято как код 11, т.е. «клавиша не нажата».

4) Запись нулей в порт A, B и $C_{верхн.}$ производит гашение индикатора, после чего весь процесс повторяется для следующей позиции индикатора и для следующей пары клавиш.



Л. 10.2. §3. Программируемый интервальный таймер (ПИТ) КР580ВИ53

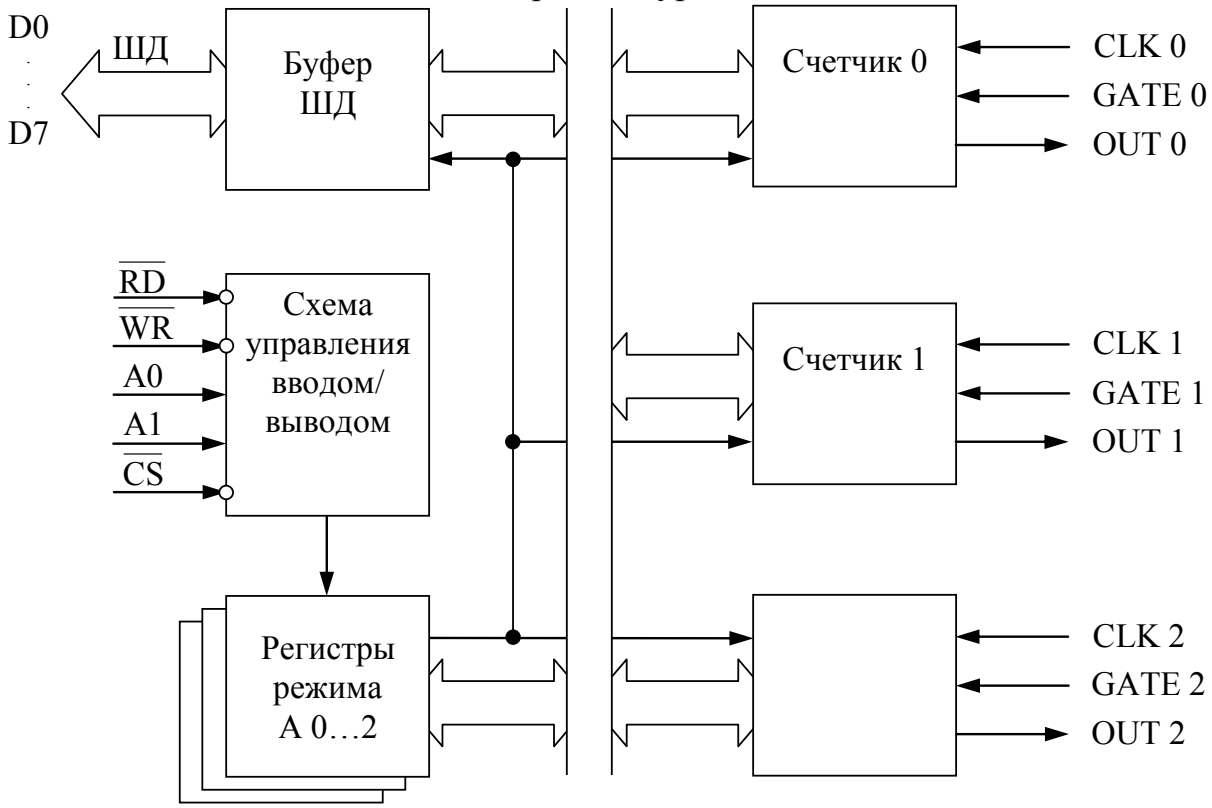
В устройствах и системах автоматики часто требуется формирование программоуправляемых временных функций:

- генерирование сигналов переменной частоты;
- подсчёт числа внешних событий;
- временные задержки и т.д.

Все эти операции можно реализовать программным путём, загружая МП подсчётом числа некоторых операций.

Применение БИС ПИТ позволяет решать все эти задачи специализированными аппаратными средствами и повысить быстродействие и эффективность МПС.

Архитектура ПИТ



В состав ПИТ входит три независимых 16-разрядных вычитающих счётчика, схема управления вводом/выводом, регистры режима работы счётчиков и буфер ШД, через который осуществляется обмен информацией между МПС и ПИТ. Счётчики ПИТ могут работать в диапазоне частот 0…2 МГц в двоичном и двоично-десятичном коде. Через буфер ШД в счётчики ПИТ можно загружать начальные значения по сигналам $\overline{WR}$ и считывать текущие значения по сигналам $\overline{RD}$. Входными сигналами, считаемыми счётчиками, являются сигналы CLK,по заднему фронту которых счётчик выполняет операцию декремента. До тех пор, пока на ПИТ не подан сигнал $\overline{CS}$, операции записи и считывания из МПС не возможны, но сигнал $\overline{CS}$ на

работу счётчиков не влияет. Сигналы A0 и адресуют один из внутренних регистров счётчиков. При $A0 = A1 = 1$ адресуется один из регистров режима (дополнительная информация об его адресе содержится в УС – управляемом слове).

Операции ПИТ

$\overline{WR}$	$\overline{RD}$	A1	A0	$\overline{CS}$	Операция
0	1	1	1	0	ШД → ПИТ: загрузка УС в регистры 0…2
1	0	1	1	0	Нет операции
0	1	0	0	0	Загрузка счётчика 0
		0	1		сч.1
		1	0		сч.2
1	0	0	0	0	Считывание счётчика 0
		0	1		сч.1
		1	0		сч.2
x	x	x	x	1	ПИТ отключен
1	1	x	x	0	---″---

Инициализация и управление работой ПИТ осуществляется с использованием управляющих слов (УС), загружаемых в регистры режимов счётчиков.

Формат УС

D7	D6	D5	D4	D3	D2	D1	D0
SC1	SC0	RL1	RL0	M2	M1	M0	B/CD

Выбор Pr или счётчика («Select Count»)

0	0	Счётчик 0
0	1	Счётчик 1
1	0	Счётчик 2
1	1	Запрещена

Чтение/запись; режим обмена («Read/Load»)

0	0	Функция (счёт «на лету»)
0	1	Младший байт
1	0	Старший байт
1	1	Младший+старший байт

Режим работы («Mode»)

0	0	0	Режим 0
0	0	1	1
x	1	0	2
x	1	1	3
1	0	0	4
1	0	1	Режим 5

Код

0	двоичный
1	двоично-десятичный

Поскольку каждый из счётчиков 16-разрядный, для загрузки исходного показания существует три возможности:

1) Пользователь загружает только младший байт (старший предполагается равным 0).
2) Только старший байт (младший предполагается 0).
3) Сначала старщий байт, затем младший.

Анализ режимов работы ПИТ необходимо проводить с учётом сигнала разрешения GATE, который формируется на периферии МПС (устройством ввода/вывода).

Режим 0 – «Программируемая задержка».

После загрузки числа в счётчик на выходе OUT устанавливается 0. По окончании отсчёта числа на выходе OUT появляется «1» и сохраняется до

загрузки нового числа или нового режима. Сигнал OUT часто используют для прерывания работы МП через определённые программнозадаваемые интервалы времени.

Перезагрузка останавливает счёт и запускает и запускает новый цикл сигнала. Счётчик работает при GATE = 1. Появление GATE = 0 останавливает счёт, сохраняя текущее состояние счётчика и при новом появлении GATE = 1 процесс счёта продолжается.

Режим 1 – «Программируемый одновибратор».

В этом режиме на входе формируется сигнал OUT = 0 длительностью

$$t_{вых} = T_{вх} \cdot N,$$

где $T_{вх}$ – период сигналов CLK;

N – число, загруженное в счётчик.

Одновибратор запускается по переднему фронту GATE. Перезагрузка счётчика новой величиной N не повлияет на длительность текущей выдержки до следующего запуска. Одновибратор является перезапускаемым, т.е. каждый фронт GATE запускает новый счёт, не сбрасывая N.

Режим 2 – «Программируемый делитель частоты».

Счётчик ПИТ работает делитель входной частоты (сигнала CLK) на N – загружаемое число. На входе формируется сигнал с частотой $f_{вх}/N$. Сигнал OUT = 1 имеет длительность $(N-1)\cdot T_{вх}$, сигнал OUT = 0 имеет длительность $T_{вх}$. Перезагрузка счётчика в процессе не меняет текущий период выходного сигнала; следующий период будет определяться новым N.

При задании режима 2 на выходе OUT сразу же устанавливается 1 и сохраняется до загрузки счётчика. Этим обеспечивается программная синхронизация генератора. Сигнал GATE используется для программной синхронизации: GATE = 0 устанавливает OUT =1, при переходе к GATE = 1 (по фронту) инициализируется счёт с начального значения.

Режим 3 – «Генератор Меаидра».

Генератор программируемой частоты со скважностью 2. Период выходного сигнала $T_{вых} = T_{вх} \cdot N$. При этом длительность 1 и 0 уровня равны $T_{вх} \cdot \frac{N}{2}$ (при N - чётном).

При нечётном N длительность OUT = 1 равна

$T_{вх} \cdot \frac{(N+1)}{2}$; OUT = 1 – $T_{вх} \cdot \frac{(N-1)}{2}$.

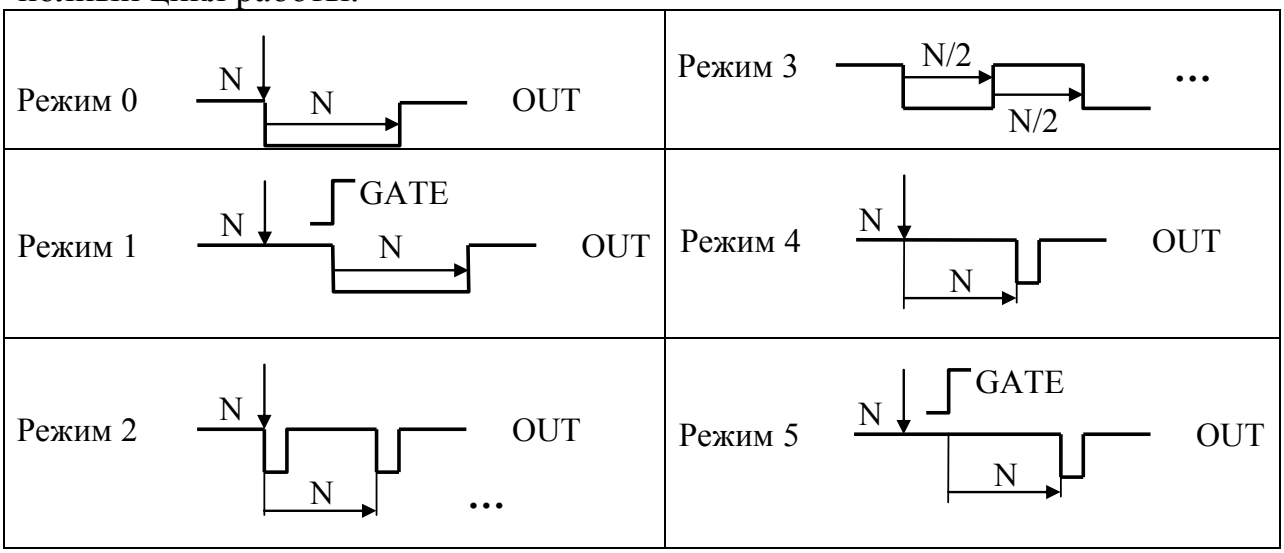
При перезагрузке счётчика новое значение N учитывается в следующем периоде выходного сигнала. Счётчик не работает при N = 3.

Режим 4 – «Строб с программируемым запуском».

По окончании отсчёта числа, загружаемого в счётчик, на выходе формируется сигнал OUT = 0 с длительностью $T_{вх}$, а затем сигнал OUT переходит в исходное состояние OUT = 1. Перезагрузка счётчика запускает новый счёт. При GATE = 0 счёт запрещён, GATE = 1 – разрешён.

Режим 5 – «Строб с аппаратным запуском».

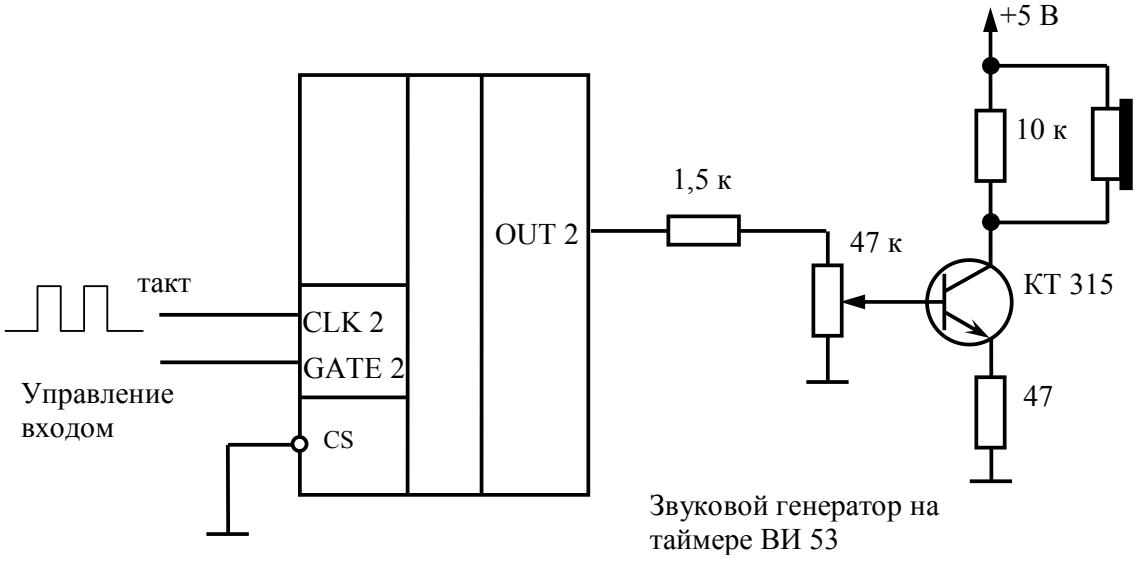
Начало счёта инициируется фронтом сигнала GATE, по достижении нулевого состояния в счётчике устанавливается OUT = 0 влечение $T_{вх}$. Счётчик является перезапускаемым, т.е. любой фронт GATE инициирует полный цикл работы.



Взаимодействие ПИТ с МПС не ограничивается только загрузкой УС и чисел в счётчик. МП может прочитать содержимое любого их счётчиков. Считывание текущего значения счётчика можно выполнить двумя способами:

1) Прямое считывание (командой IN по адресу счётчика). Такой способ требует останова счёта на время операции считывания либо сигналом GATE = 0, либо остановом подачи импульсов синхронизации CLK.
2) Считывание «на лету» – фиксация. Счёт не останавливается. Но перед считыванием нужно загрузить в регистор режима УС $[A1, A0, 0, 0, XXXX]$, а затем считать содержимое счётчика.

Чтение выполняется в том же порядке, в каком производилась и загрузка (порядок загрузки или считывания байтов определяют биты RL1 и RL0 УС).



Л. 12. Проектирование устройств и систем на базе микропроцессоров. (рис. 1!!!)

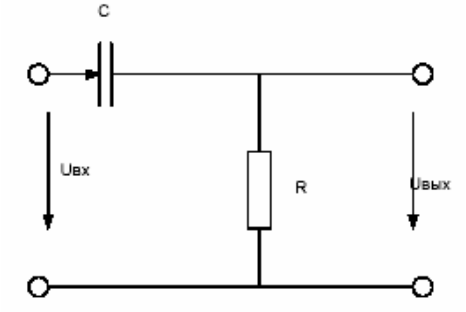
Исследование микропроцессоров при проектировании различных систем позволяет создавать устройства, особенностью которых является то, что аппаратные средства и программное обеспечение существуют здесь в форме неделимого аппаратно-программного комплекса. Разработка такого аппаратно-программного комплекса и состоит из трёх фаз проектирования:

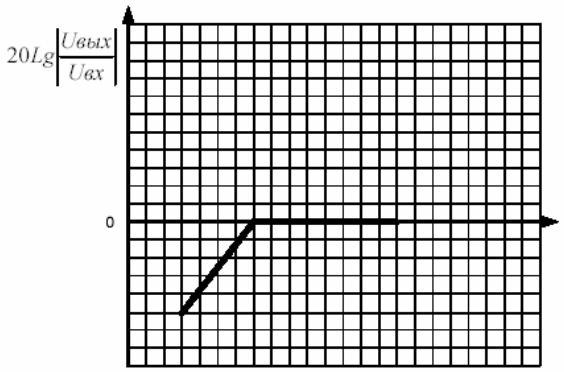
1. Разработки (и/или выбора типовых) аппаратных средств;
2. Разработки прикладного программного обеспечения;
3. Комплексирования аппаратных средств и программного обеспечения и отладки прототипа системы.

Если задача уже поставлена, то наиболее трудоёмким и сложным (из-за тесной связи с областью приложения будущей программы) этапом работы является этап формирования алгоритма поставленной задачи. Связано это с тем, что этот этап практически поддаётся формализации, и следовательно, не может быть автоматизирован обычными средствами. Проектная работа здесь носит глубоко творческий характер и сильно зависит от опыта и квалификации разработчика.

Проиллюстрируем выше сказанное одним из возможных подходов к созданию систем с использованием микропроцессора на примере проектирования цифрового фильтра.

Пусть требуется создать фильтр высокой частоты (ФВЧ) первого порядка (Рис1):





$\omega_C = 1/\tau = 1/R \cdot C$

Рис1.

Составим уравнения состояния и уравнения выхода для указанного фильтра(предполагая что на входе источник ЭДС)

$$i_c = \frac{U\text{вх} - \text{Uc}}{\text{R}}$$

$$C \cdot \frac{dUc}{dt} = \frac{U\text{вх}}{\text{R}} - \frac{\text{Uc}}{\text{R}}$$

$$\frac{dUc}{dt} = \frac{U\text{вх}}{\text{R}\cdot\text{C}} - \frac{\text{Uc}}{\text{R}\cdot\text{C}} = \frac{U\text{вх}}{\tau} - \frac{\text{Uc}}{\tau}$$ **— Уравнение состояния**

$U\text{вых} = U\text{вх} - \text{Uc}$ **— Уравнение состояния**

Заменим дифференциальные уравнения разностным уравнением состояния, выбрав, например, неявную формулу Эйлера:

$$\frac{Uc_{n+1} + Uc_n}{\Delta t} = \frac{U\text{вх}_{n+1}}{\tau} - \frac{\text{Uc}_{n+1}}{\tau},$$

Откуда выразим Uc_{n+1}:

$$Uc_{n+1} + Uc_n = \frac{\Delta t}{\tau} U\text{вх}_{n+1} - \frac{\Delta t}{\tau} \text{Uc}_{n+1}$$

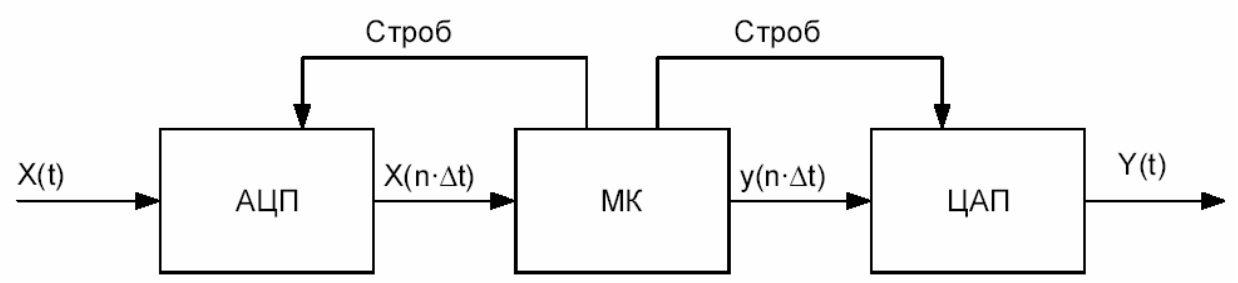
$$\left(1 + \frac{\Delta t}{\tau}\right) Uc_{n+1} = Uc_n - \frac{\Delta t}{\tau} U\text{вх}_{n+1}$$

$$Uc_{n+1} = \frac{\tau}{\tau + \Delta t} Uc_n + \frac{\Delta t}{\tau + \Delta t} U\text{вх}_{n+1}$$

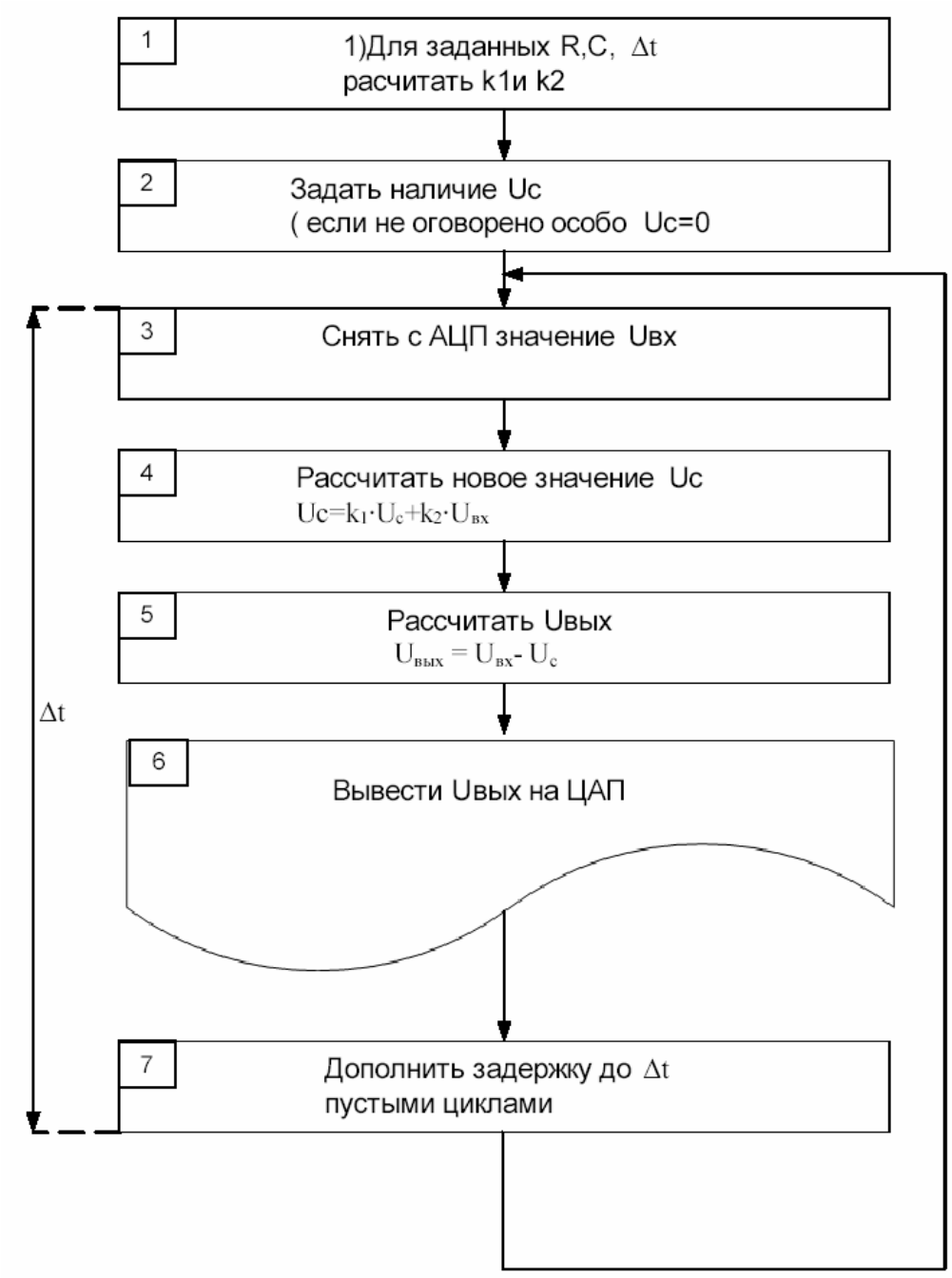
Обозначив $k1 = \frac{\tau}{\tau + \Delta t}; k2 = \frac{\Delta t}{\tau + \Delta t}$ математическое описание упомянутого фильтра можно представить следующей системой уравнений:

$$\begin{cases} \text{Uc}_{n+1} = \text{k}_1 \text{Uc}_n + \text{k}_2 \text{Uвх}_{n+1} - \text{Уравнение состояния} \\ \text{Uввы}_{n+1} = \text{Uвх}_{n+1} - \text{Uc}_{n+1} - \text{Уравнение выхода} \end{cases}$$

Аппаратная реализация предполагает наличие (кроме МП) АЦП для снятия входного напряжения и ЦАП для вывода выходного:



БСА системы может выглядеть так (если не используется таймер)



Самостоятельно: продумать БСА с использованием таймера.

Цифровые устройства, созданные на базе микропроцессоров, имеют ряд преимуществ перед аналоговыми. Приведём некоторые из них на пример рассмотренного выше цифрового фильтра:

1. Нечувствительность характеристик фильтра к разбросу параметров входящих в него элементов, их временному температурному дрейфам.
2. Малые размеры и высокая надёжность фильтра, связанные с использованием БИС.
3. Лёгкость изменения параметров и характеристик цифрового фильтра, что при использовании микропроцессора осуществляется

модификацией программного обеспечения или таблиц коэффициентов.

4. Возможность реализации адаптивных фильтров, т.е. фильтров с изменяющимися в процессе работы параметрами.

Основными недостатками цифровых устройств, реализованных на микропроцессорах, является ограниченное быстродействие, определяемое временем преобразованиям сигнала с помощью АЦП и ЦАП, временем, необходимым для работы программы вычисления, ограниченным периодом замеров входного сигнала и значениями выходного сигнала. Последние необходимо учитывать при выборе элементной базы микропроцессорного устройства.

Для выбора элементной базы микропроцессорного устройства исходными данными являются следующие:

—максимальная частота выдачи управляющих воздействий F;

—максимальное значение измеряемой величины Um;

—инструментальная точность ΔU;

—максимальная скорость изменения измеряемой величины Vm;

—число слов управления и данных N;

—ёмкость одного запоминающего элемента S;

—разрядность одного запоминающего элемента P;

На основании этих исходных данных можно определить основные характеристики устройства, влияющие на выбор элементной базы:

1) Ограничение на время выполнения программы t:

$$t = 1/F$$

2) Требуемая разрядность данных при двоичном кодировании R:

$$R \succ \left| \log_2 \frac{Um}{\Delta U} \right|$$

3) Минимальная частота дискреции измеряемой величины (минимальная тактовая частота всей системы) f:

$$f\min = \frac{Vm}{\Delta U}$$

4) Требуемое число запоминающих элементов L:

$$L \geq \frac{NR}{SP}$$

Ограниченные возможности существующей элементной базы, например, по быстродействию, а также трудности, связанные с программным исполнением некоторых операций обработки, могут привести к необходимости выполнения ряда операций с помощью аналоговых устройств. Разумное сочетание аналоговых и цифровых операций позволяет снизить требования к элементной базе и значительно упростить реализацию всего устройства.

Последовательность лекций

1
2
3
4
5 (рассказать про прерывания)
6
7
7.2 (В методичке по лабам система команд 80А, стр. 16-29)
8.1
8.2
9.1
9.2
8.3
8.4
10
11. (не читать, дать ссылку на мк51)
12

СОДЕРЖАНИЕ