Ю.В. Новиков, О.А. Калашников, С.Э. Гуляев

РАЗРАБОТКА УСТРОЙСТВ СОПРЯЖЕНИЯ ДЛЯ ПЕРСОНАЛЬНЫХ КОМПЬЮТЕРОВ ТИПА IBM PC

Ю.В. Новиков, О.А. Калашников, С.Э. Гуляев

Разработка устройств сопряжения для персонального компьютера типа IBM PC/ Под общей редакцией Ю.В. Новикова.

Книга посвящена проблеме разработки аппаратуры и программных средств для сопряжения персональных компьютеров типа IBM PC с различными внешними устройствами, возникающей при создании компьютерных систем и комплексов. Приведенные справочные данные по интерфейсам ISA, Centronics, RS-232C, а также типичные схемотехнические решения позволяют проектировать устройства сопряжения в наибольшей степени соответствующие особенностям конкретной задачи и удовлетворяющие требованиям стандартов.

Книга предназначена для разработчиков электронной аппаратуры, а также для студентов соответствующих специальностей, но может быть полезна и для специалистов, занимающихся ремонтом и обслуживанием персональных компьютеров.

ВВЕДЕНИЕ

Первый вопрос, который может возникнуть у читателей этой книги: "А зачем все это нужно?". Действительно, ведь если возникает задача сопряжения персонального компьютера с каким-то внешним устройством, то можно воспользоваться огромным количеством имеющихся на рынке стандартных устройств сопряжения (УС). При этом экономится время (но не всегда деньги), и есть гарантия (правда, не стопроцентная) того, что купленная плата не выведет ваш компьютер из строя. А здесь предлагается долгий, трудный путь проектирования, изготовления и отладки своего УС, который еще неизвестно к чему приведет. Тем не менее существует ряд соображений в пользу того, чтобы самому разработать оригинальное УС, несмотря на все трудности такого пути. Перечислим некоторые из этих соображений.

Во-первых, если стоит задача сопряжения компьютера с уникальным внешним устройством, то вполне вероятно, что на рынке может просто не оказаться подходящих модулей. Ведь всего предусмотреть невозможно. Во-вторых, стандартные УС очень часто проектируются исходя из их максимальной универсальности (а следовательно, большого объема выпуска), что нередко приводит к их довольно высокой стоимости по сравнению со специализированными УС. Наконец, в-третьих, кто-то ведь должен разрабатывать новые, более совершенные УС, продающиеся затем на рынке, и получать за это деньги, а не платить свои за покупные УС. Поэтому будем считать, что разработка оригинальных УС все-таки имеет определенный смысл.

Несмотря на важность этой проблемы, в литературе она не получила достаточного освещения. Отчасти это связано с тем, что за рубежом разработкой УС занимается довольно ограниченное число фирм, и информация по данной тематике считается там не предназначенной для широкого пользователя. Отечественная же литература очень часто ориентируется именно на перевод зарубежных источников. И хотя до недавнего времени в

нашей стране было огромное количество разработчиков (в том числе, и очень талантливых), которые имели опыт проектирования УС, сейчас многие из них занялись другими проблемами, что привело к утрате накопленного потенциала. Но продавая и перепродавая пусть даже самые лучшие зарубежные изделия, нельзя решить всех проблем. А проектирование УС стало сейчас доступным практически всем желающим, так как появился доступ к самым различным электронным компонентам, о которых раньше не приходилось и мечтать.

Всю литературу, имеющую отношение к разработке УС, можно разделить на две большие группы. В первую из них входят книги, описывающие устройство процессоров персональных компьютеров и всевозможных микросхем контроллеров, входящих в них. В некоторых из них приводится и описание устройства компьютера в целом, его ремонта и модернизации. Однако очень редко в таких книгах можно встретить даже краткое описание особенностей системных магистралей компьютеров.

Вторая группа книг касается в основном проблем организации программного обеспечения для обмена информацией с внешними устройствами. То есть в этих книгах описываются только начальный и конечный этапы решения задачи сопряжения. Однако внутреннее устройство компьютера — это далеко не самое важное из того, что нужно знать разработчику УС. Для него компьютер — "черный ящик", имеющий несколько внешних разъемов, к которым собственно и подключаются УС. И гораздо важнее для разработчика детальное знание особенностей сигналов на этих разъемах, соглашений об обмене информацией по интерфейсам, правил электрического и временного согласования и т.д.

Очень важно для разработчика также знать типичные примеры конкретной схемотехнической реализации узлов УС, особенно некоторых наиболее ответственных, чтобы не делать грубых ошибок. Это сильно облегчает процесс проектирования. Конечно же, разработчик должен уметь составить программу обмена информацией со своим УС, которая бы в наибольшей степени была ориентирована на учет всех особенностей аппаратуры и которая в ряде случаев очень эффективно может взять на себя многие функции этой аппаратуры, снизив стоимость УС.

И еще одна важная часть проблемы разработки УС. Любое новое УС, особенно УС, ориентированное на сопряжение с системной магистралью, может нарушить работу компьютера вплоть до полного выхода его из строя. Поэтому крайне желательно в процессе разработки использовать эффективные средства отладки УС, особенно если данная разработка — не единичный случай, и вы собираетесь заниматься этим деломи впредь.

Все эти вопросы должны решаться в комплексе. Только в этом случае можно надеяться на успешное решение задачи разработки оригинальных и эффективных УС самого различного назначения.

И несколько слов об этой книге. Она написана сотрудниками Московского инженерно-физического института к.т.н. Ю.В. Новиковым (гл. 1 и 2), к.т.н. О.А. Калашниковым (п. 2.2 и гл. 3) и С.Э. Гуляевым (гл. 4) на основании определенного опыта авторов по проектированию устройств сопряжения самого различного назначения и на основе материалов учебных курсов, преподаваемых в течении ряда лет студентам кафедры Электроники МИФИ.

Практически все схемы, приведенные в книге, разработаны, собраны, отлажены и испытаны в различных режимах самими авторами. Некоторые из описанных устройств успешно работают в течение ряда лет как в МИФИ, так и в других организациях.

Глава 1

Методы подключения устройств сопряжения

В чем состоит особенность проектирования любого устройства сопряжения (УС) по сравнению с другими электронными устройствами? УС, как следует из его названия, подключается к уже готовой системе (в нашем случае — к персональному компьютеру). То есть разработчик УС должен всегда учитывать возможность того, что его устройство может нарушить работу системы в целом, причем не исключено, что только в одном, редко используемом режиме. Поэтому от разработчика УС требуется повышенное внимание при проектировании, а также аккуратная и тщательная отладка УС. При этом свобода разработчика ограничена особенностями внешних интерфейсов компьютера, которые надо знать и максимально использовать.

При разработке УС особенно полезно помнить одно важнейшее правило. Любая работа обычно может быть выполнена двумя путями: медленно и хорошо или быстро и плохо (две другие ситуации: быстро и хорошо, медленно и плохо мы не рассматриваем в силу их очевидности). Так вот, если вы сделаете работу медленно и хорошо, то все очень скоро забудут, что она сделана медленно (хотя могут и побить), но очень долго будут помнить, что она выполнена хорошо. И наоборот, если вы сделаете работу быстро и плохо, то очень скоро все забудут, что она сделана быстро (хотя сначала будут хвалить), но никогда не забудут, что она выполнена плохо. Под словом "все" и в том, и в другом случае понимаются не только ваше начальство, коллеги, заказчики, покупатели но и вы сами.

Прежде чем перейти непосредственно к теме данной главы, вспомним несколько громоздкое, наукообразное, но имеющее, тем не менее, глубокий смысл определение. Интерфейс — это совокупность унифицированных аппаратных, программных и конструктивных средств, необходимых для реализации взаимодействия различных функциональных элементов в системах при условиях, предписанных стандартом и направленных на обеспечение информационной, электрической и конструктивной совместимости указанных элементов. Мы не будем его комментировать, но советуем обратить внимание на все термины, использованные здесь. Невыполнение даже одного требования из перечисленных не позволит вам создать нечто действительно работоспособное, полезное и удобное.

1.1. Сравнение методов подключения устройств сопряжения

К персональному компьютеру типа IBM PC (как,впрочем, и к компьютерам других типов) УС могут быть подключены тремя путями, соответствующими трем типам стандартных внешних интерфейсов, средства которых входят в базовую конфигурацию компьютера:

1. Через системную магистраль или шину, канал — эти термины равнозначны (в нашем случае это ISA — Industrial Standard Architecture);
2. Через параллельный интерфейс Centronics;
3. Через последовательный интерфейс RS-232C.

Отметим, что в данной книге мы не будем подробно рассматривать особенности проектирования УС для других типов интерфейсов, встречающихся в персональных компьютерах рассматриваемого типа, например, EISA (Extended ISA), PCI (Peripheral Component Interconnect), VLB (Video Local Bus) или VESA (Video Electronics Standards

Association), PCMCIA (Personal Computer Memory Card International Association). В частности, это связано с ограниченным объемом книги. Некоторые сведения об этих интерфейсах приведены в приложениях. Выбор же ISA в качестве основной системной магистрали объясняется тем, что она является наиболее распространенной. Разъемы (слоты) ISA имеются как в допотопных IBM PC XT, так и в новейших Pentium-компьютерах. Конечно же, более новые 32-разрядные интерфейсы обеспечивают большую скорость обмена и более высокую гибкость, но, научившись проектировать УС для ISA, разработчик легко сможет освоить как упомянутые магистрали, так и все те, которые появятся в будущем.

Каждый из трех указанных методов подключения имеет свои преимущества и недостатки. Выбор одного из них — важнейший шаг в самом начале процесса проектирования УС. Конечно же, здесь не рассматривается задача подключения к персональному компьютеру внешних устойств, имеющих стандартные интерфейсы Centronics и RS-232C (в этом случае УС представляет собой самый обычный соответствующим образом распаянный кабель, и никакого проектирования не требуется).

В таблице 1.1 приведено сравнение этих трех методов подключения по восьми параметрам, которые надо учитывать при выборе одного из них. Из таблицы видно, что выбор системной магистрали обеспечивает наибольшую скорость обмена. При этом не требуется ни отдельного конструктива (плата УС устанавливается в корпус компьютера), ни дополнительного источника питания (используется тот, который есть в компьютере). В то же время, одноплатное исполнение ограничивает сложность УС, а соседство с быстродействующими и мощными цифровыми узлами компьютера приводит к высокому уровню электромагнитных помех и наводок по цепям питания.

Выбор Centronics или RS-232C позволяет расположить УС (причем УС любой сложности) на большом расстоянии от компьютера. Но при этом достигается гораздо меньшая скорость обмена, а также требуется внешний конструктив и дополнительный источник питания, что существенно увеличивает стоимость системы. Немаловажно и то, что без специальных ухищрений через эти интерфейсы можно подключить только одно УС. Что касается сложности узлов сопряжения (интерфейсной части УС), то понятно, что обмен в параллельном формате гораздо проще, чем в последовательном.

В заключение этого раздела — несколько слов о терминологии. Синонимами термина "устройство сопряжения" являются термины "адаптер", "контроллер" (вообще-то понятие контроллера обычно гораздо шире — это любое управляющее устройство). Иногда УС несколько неправильно называют интерфейсом. Если УС ориентировано на системную магистраль, его еще называют платой (картой) расширения. Сути дела выбор того или иного термина не меняет. Задача — сопряжение компьютера с каким-то внешним устройством, прибором, установкой, комплексом, процессом и т.д. И небольшой словарик.

Задатчик — активное устройство на магистрали, управляющее обменом в данном цикле.

Исполнитель — пассивное устройство на магистрали, к которому обращается задатчик в данном цикле.

Асинхронный обмен — обмен информацией в темпе, определяемом быстродействием исполнителя (то есть с ожиданием задатчиком исполнения требуемой операции).

Синхронный обмен — обмен информацией в темпе задатчика, без учета быстродействия исполнителя.

Установка сигнала — перевод сигнала в активное состояние.

Снятие сигнала — перевод сигнала в пассивное состояние.

Положительный фронт сигнала — переход сигнала из единицы в нуль.

Отрицательный фронт сигнала — переход сигнала из нуля в единицу.

Передний фронт сигнала — переход сигнала из пассивного состояния в активное.

Задний фронт сигнала — переход сигнала из активного состояния в пассивное.

Радиальное прерывание — прерывание, адрес вектора которого определяется только

номером линии запроса прерывания.

Векторное прерывание — прерывание, адрес вектора которого задается устройством, запросившим прерывание.

	Системная магистраль ISA	Интерфейс Centronics	Интерфейс RS-232C
Скорость обмена	Высокая (до 5 Мбайт/с и выше)	Средняя (до 100 Кбайт/с)	Низкая
Длина и тип линии связи с компьютером	Встроенные УС (линия связи отсутствует)	До 2 метров, многопроводный кабель	До 15 метров, одиночныйпровод
Допустимая сложность УС	От малой до средней	Любая	Любая
Сложность узлов сопряжения с интерфейсом	От малой до средней	От малой до средней	От средней до высокой
Необходимость дополнительного конструктива	Не нужен	Нужен	Нужен
Необходимость внешнего источника питания	Не нужен	Нужен	Нужен
Формат и разрядность данных	Параллельный, 8 или 16 разрядов	Параллельный, 8 разрядов	Последова-тельный
Количество УС, подключаемых к компьютеру	До 6	1	1

Табл. 1.1. Сравнение методов подключения УС

1.2. Порядок обмена по системной магистрали ISA.

Стуктура персонального компьютера типа IBM PC с точки зрения разработчика УС, ориентированных на ISA, может быть условно представлена в виде рис. 1.1. Помимо центрального процессора, системной памяти (оперативной и постоянной), стандартных средств ввода/вывода, входящих в любую микропроцессорную систему, здесь следует отдельно выделить встроенные контроллеры прерываний и прямого доступа к памяти (ПДП), перестановщик байтов данных, программируемый таймер и контроллер регенерации памяти. Все эти устройства, расположенные на материнской (системной) плате (motherboard) компьютера или вставленные в слоты ISA (устройства ввода/вывода), участвуют в обмене по магистрали и могут быть использованы разрабатываемыми УС.

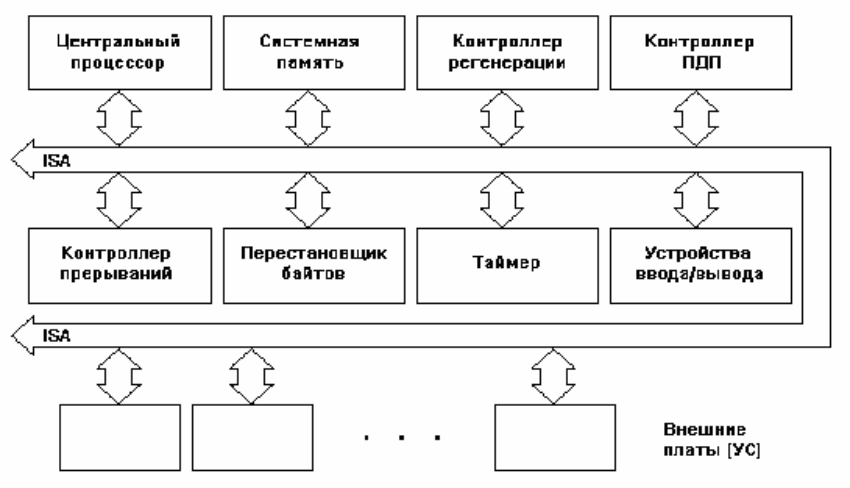


Рис. 1.1. Структура персонального компьютера

Задатчиками (хозяевами) шины могут выступать центральный процессор (самая обычная ситуация), контроллер ПДП, контроллер регенерации и некоторые внешние платы. В каждом цикле обмена задатчиком всегда является только одно устройство. Котроллер ПДП захватывает магистраль (запрещает работу центрального процессора) на время прямой передачи информации между устройством ввода/вывода и памятью (по запросу устройства ввода/вывода). Контроллер регенерации периодически становится задатчиком магистрали для проведения циклов регенерации системной динамической памяти через заданные интервалы времени. Для 32-разрядных компьютеров (386DX, 486, Pentium и т.д.) обмен процессора с памятью (а иногда и с другими устройствами) осуществляется через быстродействующую локальную шину VLB или через PCI.

1.2.1. Особенности магистрали ISA

Магистраль ISA была разработана специально для персональных компьютеров типа IBM PC AT (начиная с процессора i80286) и является фактическим стандартом для всех изготовителей этих компьютеров. В то же время, отсутствие официального международного статуса магистрали ISA (она не утверждена в качестве стандарта ни одним международным комитетом по стандартизации) приводит к тому, что многие производители допускают некоторые, порой существенные отклонения от фирменного стандарта. На это будет обращено особое внимание в книге.

ISA явилась расширением магистрали компьютеров IBM PC и IBM PC XT. В ней было увеличено количество разрядов адреса и данных, увеличено число линий аппаратных прерываний и каналов ПДП, а также повышена тактовая частота. К 62-контактному разъему прежней магистрали был добавлен 36-контактный новый разъем. Тем не менее, совместимость была сохранена, и платы, предназначенные для IBM PC XT, годятся и для IBM PC AT. Характерное отличие ISA состоит в том, что ее тактовый сигнал не совпадает с тактовым сигналом процессора, как это было в XT, поэтому скорость обмена по ней не пропорциональна тактовой частоте процессора.

Магистраль ISA относится к демультиплексированным (то есть имеющим раздельные шины адреса и данных) 16-разрядным системным магистралям среднего быстродействия. Обмен осуществляется 8-ми или 16-ти разрядными данными. На

магистрали реализован раздельный доступ к памяти компьютера и к устройствам ввода/вывода (для этого имеются специальные сигналы). Максимальный объем адресуемой памяти составляет 16 Мбайт (24 адресные линии). Максимальное адресное пространство для устройств ввода/вывода — 64 Кбайта (16 адресных линий), хотя практически все выпускаемые платы расширения используют только 10 адресных линий (1 Кбайт). Магистраль поддерживает регенерацию динамической памяти, радиальные прерывания и прямой доступ к памяти. Допускается также захват магистрали.

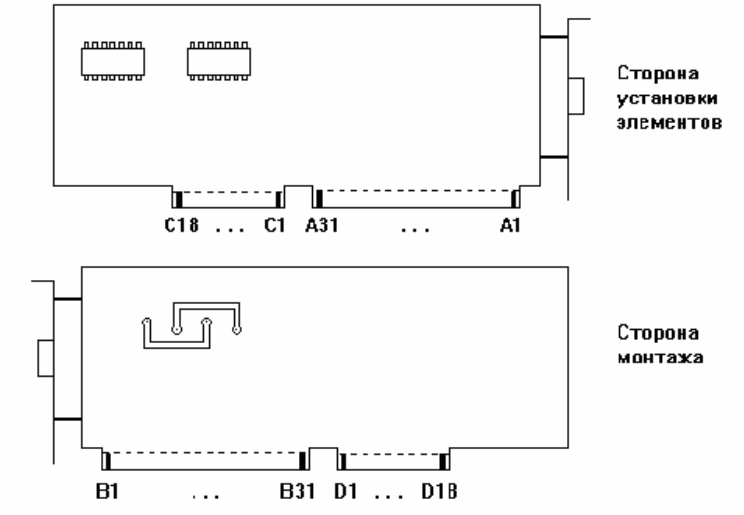


Рис. 1.2. Нумерация контактов разъема ISA (для IBM PC XT — только A1 ... A31 и B1 ... B31)

Наиболее распространенное конструктивное исполнение магистрали — разъемы (слоты), установленные на материнской плате компьютера, все одноименные контакты которых соединены между собой, то есть все разъемы абсолютно равноправны. Особенностью конструктивного решения магистрали является то, что платы расширения (дочерние платы), подключаемые к ее разъемам, могут иметь самые различные размеры (длина платы ограничена снизу размером разъема, а сверху — длиной корпуса компьютера). Платы расширения имеют интерфейсные разъемы магистрали, выполненные печатными проводниками. Количество установочных мест для плат расширения зависит от типа корпуса компьютера и составляет обычно 2-3 для Ultra-slimline корпусов, 3-4 для Slimline корпусов, 5-6 для Desktop корпусов, 4-5 для Mini-tower корпусов, 5-7 для Midi-tower корпусов и более 8 для Big-tower корпусов.

Разъем магистрали ISA разделен на две части, что позволяет уменьшать размеры 8-разрядных плат расширения, а также использовать платы, разработанные для компьютеров IBM PC XT. Внешний вид плат расширения показан на рис. 1.2, а точные их размеры приведены в приложении. Назначение контактов разъемов представлено в таблице 1.2 (здесь знак минус перед названием сигнала говорит о том, что активным уровнем этого сигнала является уровень логического нуля, в противном случае активным уровнем будет уровень логической единицы). Отметим, что в магистрали ISA используется положительная логика на шинах адреса и данных, то есть единице соответствует высокий уровень напряжения, а нулю — низкий). На магистрали присутствуют четыре напряжения питания: +5 В, -5 В, +12 В и -12 В, которые могут использоваться платами расширения.

Контакт	Цепь	I/O	Контакт	Цепь	I/O
A1	-I/O CH CK	I	B1	GND	-
A2	SD7	I/O	B2	RESET DRV	O
A3	SD6	I/O	B3	+5 B	-
A4	SD5	I/O	B4	IRQ9 (IRQ2)	I
A5	SD4	I/O	B5	-5 B	-
A6	SD3	I/O	B6	DRQ2	I
A7	SD2	I/O	B7	-12 B	-
A8	SD1	I/O	B8	0WS	I
A9	SD0	I/O	B9	+12 B	-
A10	I/O CH RDY	I	B10	GND	-
A11	AEN	O	B11	-SMEMW	O
A12	SA19	I/O	B12	-SMEMR	O
A13	SA18	I/O	B13	-IOW	I/O
A14	SA17	I/O	B14	-IOR	I/O
A15	SA16	I/O	B15	-DACK3	O
A16	SA15	I/O	B16	DRQ3	I
A17	SA14	I/O	B17	-DACK1	O
A18	SA13	I/O	B18	DRQ1	I
A19	SA12	I/O	B19	- REFRESH	I/O
A20	SA11	I/O	B20	SYSCLK	O
A21	SA10	I/O	B21	IRQ7	I
A22	SA9	I/O	B22	IRQ6	I
A23	SA8	I/O	B23	IRQ5	I
A24	SA7	I/O	B24	IRQ4	I
A25	SA6	I/O	B25	IRQ3	I
A26	SA5	I/O	B26	-DACK2	O
A27	SA4	I/O	B27	T/C	O
A28	SA3	I/O	B28	BALE	O
A29	SA2	I/O	B29	+5 B	-
A30	SA1	I/O	B30	OSC	O
A31	SA0	I/O	B31	GND	-

Табл. 1.2. Назначение контактов разъема ISA (продолжение на следующей странице)

Контакт	Цепь	I/O	Контакт	Цепь	I/O
C1	-SBHE	I/O	D1	-MEM CS16	I
C2	LA23	I/O	D2	-I/O CS16	I
C3	LA22	I/O	D3	IRQ10	I
C4	LA21	I/O	D4	IRQ11	I
C5	LA20	I/O	D5	IRQ12	I
C6	LA19	I/O	D6	IRQ15	I
C7	LA18	I/O	D7	IRQ14	I
C8	LA17	I/O	D8	-DACK0	O
C9	-MEMR	I/O	D9	DRQ0	I
C10	-MEMW	I/O	D10	-DACK5	O
C11	SD8	I/O	D11	DRQ5	I
C12	SD9	I/O	D12	-DACK6	O
C13	SD10	I/O	D13	DRQ6	I
C14	SD11	I/O	D14	-DACK7	O
C15	SD12	I/O	D15	DRQ7	I
C16	SD13	I/O	D16	+5 B	-
C17	SD14	I/O	D17	-MASTER	I
C18	SD15	I/O	D18	GND	-

Табл. 1.2. (продолжение) Назначение контактов разъема ISA (I — входной сигнал, O — выходной сигнал, I/O — двунаправленный сигнал)

1.2.2. Сигналы магистрали ISA.

Рассмотрим теперь назначение сигналов магистрали ISA и их особенности.

SA0...SA19 — фиксируемые адресные разряды (они действительны в течение всего цикла обмена). Используются для передачи 20 младших разрядов адреса памяти и для адресов устройств ввода/вывода. При обращении к устройствам ввода/вывода действительны только сигналы SA0...SA15 (но практически все платы расширения работают только с SA0...SA9). Распределение адресов устройств ввода/вывода представлено в таблице 1.3, а распределение адресов памяти — в таблице 1.4. Легко заметить, что значительная часть этих адресов занята стандартными устройствами компьютера. При регенерации памяти действительны только сигналы SA0...SA7, состояния старших разрядов не определены.Логика всех сигналов SA0...SA19 — положительная. В режиме MASTER эти сигналы вырабатывает устройство, захватившее магистраль. Тип выходных каскадов — три состояния.

LA17...LA23 — нефиксируемые адресные разряды. Используются для адресации памяти и выработки сигнала -MEM CS 16. Действительны только в начале цикла обмена. Исполнитель должен фиксировать их по отрицательному фронту сигнала BALE. При обращении к устройствам ввода/вывода эти сигналы имеют уровень логического нуля. Логика положительная. Тип выходного каскада — три состояния. Для фиксации необходимо использовать регистр типа "защелка" (с записью по уровню), стробируемый сигналом BALE (например, КР1533ИР33, К555ИР22). При прямом доступе к памяти эти сигналы действительны в течение всего цикла обмена, как и SA0...SA19. В режиме MASTER эти сигналы вырабатывает устройство, захватившее магистраль. Тип выходных каскадов — три состояния.

BALE (Bus Address Latch Enable — разрешение защелкивания адреса) — сигнал стробирования адресных разрядов. Его отрицательный фронт соответствует действительности адреса на линиях SA0...SA19 и LA17...LA23. Может использоваться устройствами ввода/вывода для заблаговременной подготовки к предстоящему обмену информацией (применяется редко). Тип выходного каска-да — ТТЛ.

Адреса	Назначение
000...01F	Контроллер ПДП 1
020...03F	Контроллер прерываний 1
040...05F	Программируемый таймер
060...06F	Контроллер клавиатуры
070...07F	Часы реального времени
080...09F	Регистр страницы ПДП
0A0...0BF	Контроллер прерываний 2
0C0...0DF	Контроллер ПДП 2
0F0...0FF	Математический сопроцессор
170...177	Накопитель на жестком диске (второй)
1F0...1F7	Накопитель на жестком диске (первый)
200...207	Игровой порт (джойстик)
278...27F	Параллельный порт LPT2
2C0...2DF	Адаптер EGA 2
2F8...2FF	Последовательный порт COM2
300...31F	Прототипные платы
320...32F	Накопитель на жестком диске XT
360...36F	Резервные адреса
370...377	Накопитель на гибком диске (второй)
378...37F	Параллельный порт LPT1
380...38F	Контроллер бисинхронного обмена SDLC2
3A0...3AF	Контроллер бисинхронного обмена SDLC1
3B0...3DF	Адаптер VGA
3B0...3BF	Адаптер монохромного дисплея MDA и принтера
3C0...3CF	Адаптер EGA 1
3D0...3DF	Адаптер CGA
3F0...3F7	Накопитель на гибком диске (первый)
3F8...3FF	Последовательный порт COM1

Табл. 1.3. Распределение адресов устройств ввода/вывода ISA (адреса даны в 16-ричном коде)

Адреса памяти	Назначение
000000...0003FF	Таблица векторов прерываний
000000...09FFFF	Память DOS и пользовательских программ
0A0000...0AFFFF	Память дисплея EGA или VGA
0B0000...0B7FFF	Память монохромного дисплея MDA
0B8000...0BFFFF	Память дисплея CGA
0C0000...0C3FFF	ПЗУ BIOS для EGA/VGA
0C8000...0DFFFF	ПЗУ устройств ввода/вывода
0E0000...0EFFFF	Резерв ПЗУ BIOS на материнской плате
0F0000...0FFFFF	ПЗУ BIOS на материнской плате

Табл. 1.4. Распределение адресов памяти

Задатчик (процессор)			Исполнитель (УВВ)		Выполняемый цикл		
Размер данных	-SBHE	SA0	Размер данных	-CS16	Размер данных	Операция: чтение	запись
8	1	0	8	1	8	L=>L	L=>L
8	0	1	8	1	8	L=>H	H=>L
8	1	0	16	0	8	L=>L	L=>L
8	0	1	16	0	8	H=>H	H=>H
16	0	0	8	1	8	L=>L	L=>L
16	0	0	16	0	16	L=>L H=>H	L=>L H=>H

Табл. 1.5. Тип выполняемых операций в зависимости от сигналов -SBHE и SA0 при программном обмене (L — младший байт, H — старший байт, УВВ — устройство ввода/вывода)

УВВ	Контроллер ПДП		Память		Выполняемый цикл		
Размер данных	-SBHE	SA0	Размер данных	-CS16	Размер данных	Операция: чтение	запись
8	1	0	8	1	8	L=>L	L=>L
8	1	0	16	0	8	L=>L	L=>L
8	X	1	8	1	8	L=>L	L=>L
8	X	1	16	0	8	H=>L	L=>H
16	0	0	8	1	8	Запрещено	
16	0	0	16	0	16	L=>L H=>H	L=>L H=>H

Табл. 1.6. Тип выполняемых операций в зависимости от сигналов -SBHE и SA0 при ПДП (L — младший байт, H — старший байт, УВВ — устройство ввода/вывода)

-SBHE (System Bus High Enable — разрешение старшего байта) — определяет тип цикла передачи данных (8-ми или 16-ти разрядный). Вырабатывается параллельно с сигналами SA0...SA19 и может рассматриваться как дополнительный разряд адреса. Становится активным при передаче старшего байта или 16-разрядного слова (определяется сигналом SA0), пассивен при передаче младшего байта. В режиме MASTER источником этого сигнала является устойство, которое захватило магистраль. Тип выходного каскада — три состояния. В таблице 1.5 приведены типы выполняемых операций при различных значениях сигналов -SBHE и SA0 в случае программного обмена, а в таблице 1.6 — в случае прямого доступа к памяти.

SD0...SD15 — разряды данных. По линиям SD0...SD7 передается младший байт, по линиям SD8...SD15 — старший байт. Обмен данными с 8-разрядными платами расширения осуществляется по линиям SD0...SD7. Устройство может активизировать шину данных, если к нему идет обращение с циклом чтения или если оно захватило магистраль (в режиме MASTER). Логика сигналов положительная. Тип выходных каскадов — три состояния.

-SMEMR, -MEMR (Memory Read — чтение памяти) — стробы чтения данных из памяти. Память должна выставлять данные при активизации этих сигналов. Сигнал -SMEMR вырабатывается только при обращении к адресам, не превышающим FFFFF (в пределах 1 Мбайта), сигнал -MEMR — при обращении ко всем адресам. В режиме MASTER эти сигналы вырабатывает устройство, захватившее магистраль. Тип выходных каскадов — три состояния.

-SMEMW, -MEMW (Memory Write — запись памяти) — стробы записи данных в память. Память должна принимать данные по положительному (заднему) фронту этих сигналов. Сигнал -SMEMW вырабатывается только при обращении к адресам, не превышающим FFFFF (в пределах 1 Мбайта), сигнал -MEMW — при обращении ко всем адресам. В режиме MASTER эти сигналы вырабатывает устройство, захватившее магистраль. Тип выходных каскадов — три состояния.

-IOR (I/O Read) — строб чтения данных из устройств ввода/вывода. Устройство ввода/вывода должно выставлять свои данные при активизации сигнала -IOR и снимать их при снятии -IOR. В режиме MASTER этот сигнал вырабатывает устройство, захватившее магистраль. Тип выходного каскада — три состояния.

-IOW (I/O Write) — строб записи данных в устройства ввода/вывода. Устройство ввода/вывода должно принимать данные по положительному (заднему) фронту сигнала -IOW. В режиме MASTER этот сигнал вырабатывает устройство, захватившее магистраль. Тип выходного каскада — три состояния.

-MEM CS16 (Memory Cycle Select — выбор цикла для памяти) — сигнал выставляется памятью для сообщения задатчику о том, что она имеет 16-разрядную организацию. При отсутствии этого сигнала выполняется 8-разрядный обмен. Сигнал вырабатывается при распознавании памятью своего адреса на линиях LA17...LA23. Процессор фиксирует его по заднему фронту сигнала BALE. Тип выходного каскада — открытый коллектор.

-I/O CS16 (I/O Cycle Select — выбор цикла для устройства ввода/вывода) — сигнал выставляется устройством ввода/вывода для сообщения задатчику о том, что оно имеет 16-разрядную организацию. При отсутствии этого сигнала выполняется 8-разрядный обмен. Сигнал вырабатывается при распознавании устройством ввода/вывода своего адреса на линиях SA0...SA15. Тип выходного каскада — открытый коллектор.

I/O CH RDY (I/O Channel Ready — готовность канала ввода/вывода) — сигнал снимается (делается низким) исполнителем (устройством ввода/вывода или памятью) по переднему фронту сигналов -IOR и -IOW в случае, если он не успевает выполнить требуемую операцию в темпе задатчика. При этом реализуется асинхронный обмен. Если исполнитель успевает работать в темпе задатчика, сигнал не снимается (фактически не устанавливается в низкий уровень). Цикл обмена в ответ на снятие этого сигнала продлевается на целое число периодов сигнала SYSCLK. Сигнал I/O CH RDY не должен

сниматься на время, большее заданного в данном компьютере (по стандарту — 15 мкс), иначе компьютер переходит к обработке немаскируемого прерывания. Тип выходного каскада — открытый коллектор.

-I/O CH CK (I/O Channel Check — проверка канала ввода/вывода) — сигнал вырабатывается любым исполнителем (устройством ввода/вывода или памятью) для информирования задатчика о фатальной ошибке, например, об ошибке четности при доступе к памяти. Сигнал вызывает немаскируемое прерывание. Тип выходного каскада — открытый коллектор.

-0WS (0 Wait States — 0 тактов ожидания) — выставляется исполнителем для информирования задатчика о необходимости проведения цикла обмена без вставки такта ожидания, если длительность стандартного цикла обмена велика для него. Вырабатывается после перехода сигнала BALE в низкий уровень. Должен быть синхронизован с сигналом SYSCLK. Используется редко. Тип выходного каскада — открытый коллектор.

-REFRESH (Refresh — регенерация) — сигнал выставляется контроллером регенерации для информирования всех устройств на магистрали о выполнении циклов регенерации динамического ОЗУ компьютера (каждые 15 мкс). При регенерации выполняется псевдочтение из одного из 256 адресов ОЗУ (активизируются только разряды адреса SA0...SA7). Полный цикл регенерации — около 4 мс. Тип выходного каскада — открытый коллектор.

RESET DRV (Reset of Driver — сброс устройства) — сигнал сброса в начальное состояние всех устройств на магистрали. Вырабатывается центральным процессором при включении или сбое питания, а также при нажатии на кнопку RESET компьютера. Внешние платы должны в ответ на этот сигнал (длительностью не менее 1 мс) перевести все свои выходы в высокоимпедансное состояние. Тип выходного каскада — ТТЛ.

SYSCLK (System Clock — системный такт) — сигнал системного тактового генератора со скважностью 2 (меандр). В большинстве компьютеров его частота равна 8 МГц независимо от тактовой частоты процессора. Если в программе SETUP предусмотрена возможность изменения тактовой частоты магистрали, пользователь может задавать ее в широких пределах. Но для обеспечения наибольшей совместимости со всеми имеющимися платами расширения ISA не рекомендуется поднимать эту частоту выше 8 МГц. К тому же на производительность новых компьютеров в целом она влияет незначительно. В компьютерах XT сигнал SYSCLK — это тактовый сигнал процессора. Тип выходного каскада — три состояния.

OSC — не синхронизированный с SYSCLK сигнал кварцевого генератора с частотой 14,31818 МГц со скважностью 2. Может использоваться платами расширения в качестве тактового сигнала, так как его частота одинакова для всех компьютеров с магистралью ISA. Тип выходного каскада — ТТЛ.

IRQ (Interrupt Request — запрос прерывания) — сигналы запроса радиальных прерываний. Запросом является положительный переход на соответствующей линии IRQ. Сигнал должен удерживаться до начала обработки процессором запрошенного прерывания. Тип выходного каскада — ТТЛ. На каждой линии IRQ должен быть один выход. Иногда в литературе можно встретить рекомендацию применять выходы с тремя состояниями, но все равно больше одного выхода на линию быть не должно во избежание конфликтов сигналов. Многие входы IRQ заняты системными ресурсами компьютера (см.

табл. 1.7). Сигналы IRQ0...IRQ2, IRQ8 и IRQ13 задействованы на системной плате и недоступны платам расширения. В компьютере используются два 8-разрядных контроллера прерываний. Сигналы IRQ0...IRQ7 относятся к первому из них, а IRQ8...IRQ15 — ко второму. Для каскадирования второго контроллера прерываний задействован вход IRQ2. В связи с этим запросы прерывания имеют следующие приоритеты в порядке возрастания: IRQ7, IRQ6, IRQ5, IRQ4, IRQ3, IRQ15, IRQ14, IRQ12, IRQ11, IRQ10, IRQ9.

DRQ (DMA Request — запрос ПДП) — сигналы запросов прямого доступа к памяти (ПДП). Запросом является положительный переход на соответствующей линии DRQ. Сигнал должен удерживаться до получения ответного сигнала -DACK с тем же номером. Тип выходного каскада — ТТЛ. На каждой линии DRQ должен быть один выход. В компьютере используются два контроллера ПДП. Каналы ПДП, соответствующие первому контроллеру (сигналы DRQ0...DRQ3) предназначены для 8-битного обмена, а соответствующие второму котроллеру (DRQ5...DRQ7) — для 16-битного. Канал DRQ4 используется для каскадирования контроллеров и недоступен пользователям. DRQ0 имеет наивысший приоритет, DRQ7 — наинизший. В IBM PC XT канал DRQ0 использовался для регенерации динамической памяти. Канал DRQ1 зарезервирован для контроллера бисинхронного обмена SDLC, а канал DRQ2 — для контроллера гибкого диска

-DACK (DMA Acknowledge — подтверждение ПДП) — сигналы подтверждения предоставления прямого доступа. Вырабатываются в ответ на соответствующий сигнал DRQ в случае, если прямой доступ предоставлен данному каналу. Удерживаются до окончания прямого доступа. Тип выходного каскада — ТТЛ.

Номер прерывания IRQ	INT	Назначение
0	08h	Программируемый таймер
1	09h	Контроллер клавиатуры
2	0Ah	Каскадирование второго контроллера
8	70h	Часы реального времени (только AT)
9	71h	Программно переадресовано на IRQ2
10	72h	Резерв
11	73h	Резерв
12	74h	Резерв
13	75h	Математический сопроцессор
14	76h	Контроллер жесткого диска
15	77h	Резерв
3	0Bh	Последовательный порт COM2
4	0Ch	Последовательный порт COM1
5	0Dh	Параллельный порт LPT2
6	0Eh	Контроллер гибкого диска
7	0Fh	Параллельный порт LPT1

Табл. 1.7. Назначение аппаратных прерываний ISA

AEN (Address Enable — разрешение адреса) — используется в режиме ПДП для сообщения всем платам расширения, что производится цикл ПДП. Устанавливается и снимается параллельно с адресом. При его переходе в активное состояние все платы расширения, не участвующие в данном ПДП, должны отключаться от магистрали (переходить в пассивное состояние). Тип выходного каскада — ТТЛ.

T/C (Terminal Count — окончание счета) — устанавливается в режиме ПДП тогда, когда по текущему каналу ПДП закончен счет циклов пересылок данных. Тип выходного каскада — ТТЛ.

-MASTER (Master — хозяин, задатчик) — используется платой расширения, желающей стать задатчиком магистрали. В этом случае надо выставить сигнал DRQ и, получив в ответ сигнал -DACK, установить сигнал -MASTER, а затем через минимум один период SYSCLK можно выставлять адрес и через минимум два периода SYSCLK можно вырабатывать стробы обмена. Если -MASTER удерживается более 15 мкс, то динамическое ОЗУ компьютера требует регенерации (разрешения сигнала -REFRESH). Тип выходного каскада — открытый коллектор.

Стандартом магистрали ISA установлены ограничения на максимальное значение тока, потребляемого каждой платой расширения (они связаны только с возможностями используемого разъема). Значения этих токов для всех напряжений питания приведены в таблице 1.8. Отметим, что максимальный ток потребления всеми используемыми платами расширения определяется типом источника питания данного компьютера и не стандартизован. Вообще же мощность блока питания зависит от класса компьютера и может варьироваться от 100-150 Вт (для slim-корпусов) до 300-330 Вт (для big-tower). Некоторые современные "зеленые" компьютеры имеют блоки питания с мощностью не более 75 Вт. Но наиболее типичные параметры источника питания IBM PC AT мощностью 200 Вт приведены в таблице 1.9.

Напряжение	8-разядная плата (XT)	16-разрядная плата
+5 В	3,0 А	4,5 А
-5 В	1,5 А	1,5 А
+12 В	1,5 А	1,5 А
-12В	1.5 А	1,5 А

Табл. 1.8. Максимальные токи потребления платами расширения

Напряжение питания источника	Допустимый ток нагрузки
+5В	7,0...19.8 А
-5 В	0,0...0,3 А
+12 В	2,5...7,3 А
-12 В	0,0...0,3 А

Табл. 1.9. Допустимые токи потребления от источника питания

Выходные напряжения источника достигают номинального уровня за время не более 100 мс после включения питания. Источники, как правило, имеют встроенную защиту от перегрузок, которая включается за время 20 мс. Источник должен быть обязательно нагружен по напряжениям +5 В и +12 В. Если по этим выходам не будет обеспечен минимальный ток потребления, это воспринимается как перегрузка. Для выхода из перегрузки надо выключить и снова включить питание источника через время не менее 1с.

1.2.3. Циклы магистали ISA.

В режиме программного обмена информацией на магистрали ISA выполняются

четыре типа циклов:

— цикл записи в память;
— цикл чтения из памяти;
— цикл записи в устройство ввода/вывода;
— цикл чтения из устройства ввода/вывода.

Наиболее часто УС проектируются как устройства ввода/вывода. Временные диаграммы циклов обмена для этого случая приведены на рис. 1.3 (все временные параметры приведены для частоты SYSCLK, равной 8 МГц). Циклы начинаются с выставления задатчиком адреса на линиях SA0...SA15 и сигнала -SBHE. Отметим, что несмотря на потенциальную возможность адресации по 16 линиям адреса, чаще всего используются только 10 младших линий SA0...SA9, так как большинство разработанных ранее плат расширения используют только их, и, следовательно, за исключением особых случаев нет смысла обрабатывать старшие разряды SA10...SA15. Это будет подробнее рассмотрено в главе 2. В ответ на получение адреса исполнитель, распознавший свой адрес, должен сформировать сигнал -I/O CS16 в случае, если обмен должен быть 16-разрядным.

Далее следует собственно команда чтения или записи. При цикле чтения задатчик выставляет сигнал -IOR, в ответ на который исполнитель (УС) должен выдать данные на шину данных. Эти данные должны быть сняты исполнителем после окончания сигнала -IOR. В цикле записи задатчик выставляет записываемые данные и сопровождает их стробом записи -IOW. Здесь надо отметить, что хотя в соответствии со стандартом установка записываемых данных предшествует выставлению -IOW, в некоторых компьютерах реализуется обратный порядок: сначала выставляется -IOW, а затем появляются данные. Поэтому при проектировании УС надо рассматривать как момент действительности данных только задний (положительный) фронт сигнала -IOW.

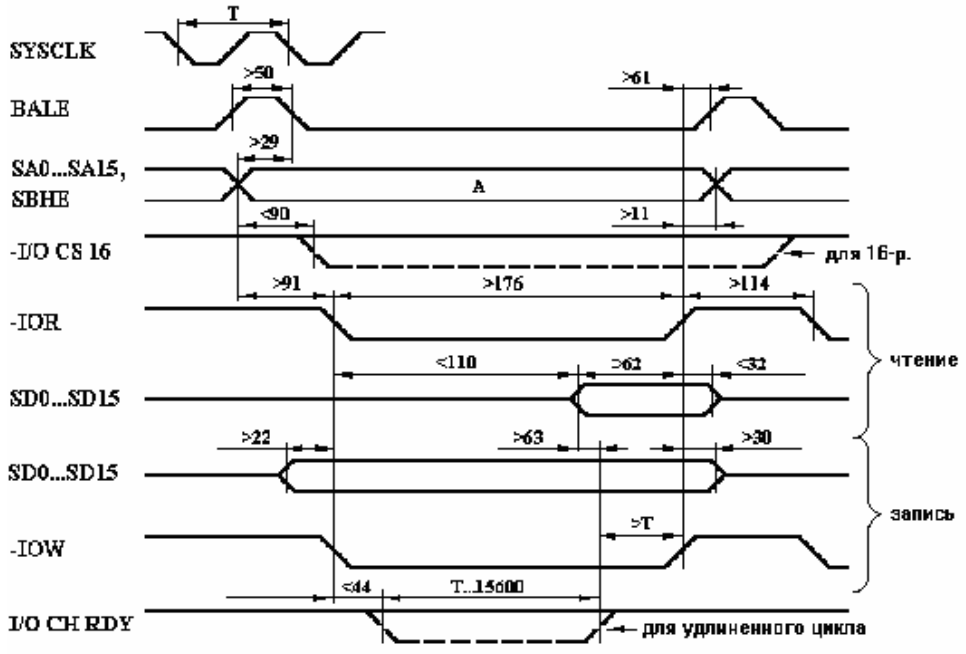


Рис. 1.3. Временные диграммы циклов программного обмена с устройствами ввода/вывода (все временные интервалы в наносекундах)

В случае, когда УС не успевает выполнить требуемую от него команду в темпе

магистрали, оно может приостановить на целое число периодов сигнала SYSCLK завершение цикла чтения или записи с помощью снятия (перевода в низкий уровень) сигнала I/O CH RDY (так называемый удлиненный цикл). Это производится в ответ на получение сигнала -IOR или -IOW. Сигнал I/O CH RDY может удерживаться низким не более 15,6 мкс, в противном случае процессор переходит в режим обработки немаскируемого прерывания. Отметим, что некоторые изготовители персональных компьютеров указывают в сопроводительной документации другие допустимые величины этого временного интервала (например, 2,5 мкс), так что не следует ориентироваться на максимальную величину, указанную в стандарте, иначе нет гарантии работы УС во всех компьютерах.

На рис. 1.4 приведены временные диаграммы циклов обмена с памятью (указаны только временные интервалы, отличающиеся от аналогичных на рис. 1.3). Для асинхронного режима обмена (удлиненного цикла) здесь также используется сигнал I/O CH RDY. Отметим, что УС, работающее как память, должно обрабатывать все адресные разряды, включая LA17...LA23.

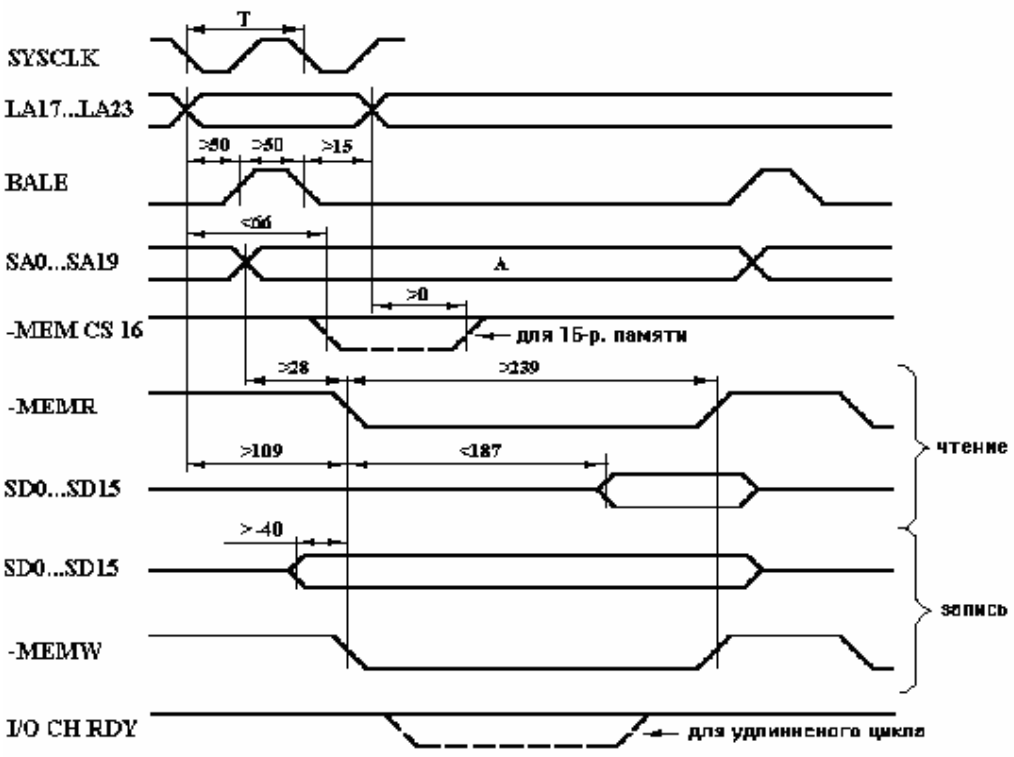


Рис. 1.4. Временные диаграммы циклов программного обмена с памятью (все временные интервалы в наносекундах)

Помимо циклов программного обмена на магистрали ISA могут выполняться также циклы прямого доступа к памяти (ПДП). Временная диаграмма для этого случая показана на рис. 1.5. Так как магистраль ISA имеет раздельные стробы чтения и записи для устройств ввода/вывода и для памяти, пересылка данных в режиме ПДП производится за один машинный цикл. То есть если данные надо переслать из устройства ввода/вывода в память, то одновременно производится чтение данных из устройства ввода/вывода (по сигналу -IOR) и их запись в память (по сигналу -MEMW). Аналогично производится пересылка данных из памяти в устройство ввода/вывода (по сигналам -MEMR и -IOW).

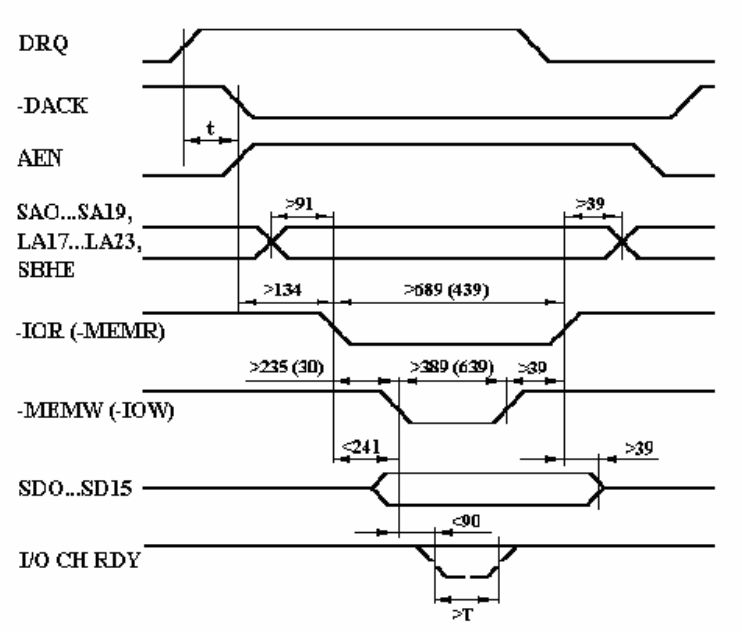


Рис.1.5. Временные диаграммы циклов ПДП (t — время предоставления ПДП, T — период сигнала SYSCLK; все временные интервалы в наносекундах)

Цикл ПДП начинается с запроса ПДП от исполнителя, желающего произвести обмен, с помощью одного из сигналов DRQ. После освобождения магистрали текущим задатчиком (например, процессором) контроллер ПДП формирует соответствующий сигнал -DACK, говорящий о предоставлении ПДП запросившему его. Затем контроллер ПДП вырабатывает адрес ячейки памяти, с которой будет проиводиться обмен в текущем цикле, и сигнал AEN, который говорит устройству ввода/вывода о том, что к нему идет обращение в режиме ПДП. После этого выставляется строб чтения (-IOR или -MEMR), в ответ на который источник передаваемых данных выставляет свою информацию на шину данных, и строб записи (-MEMW или -IOW), по которому данные записываются в приемник данных. Здесь так же как и в обычном цикле возможен асинхронный обмен (удлиненный цикл) с использованием сигнала I/O CH RDY.

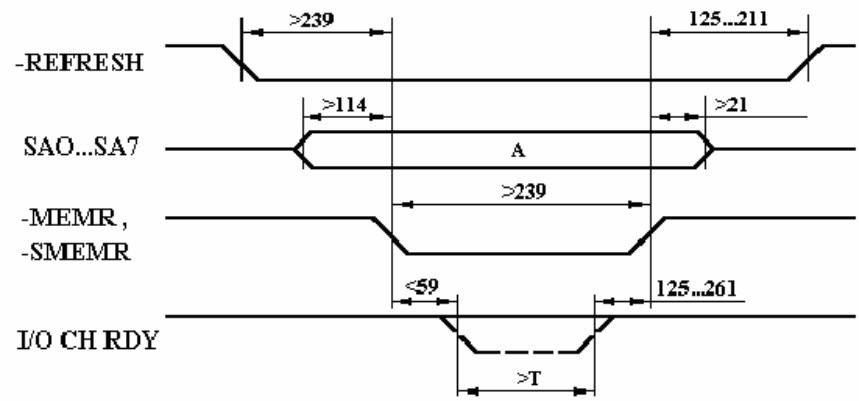


Рис.1.6. Временные диаграммы циклов регенерации (T — период сигнала SYSCLK, все временные интервалы в наносекундах)

Одной из особенностей магистрали ISA является необходимость проведения регенерации динамической памяти компьютера с помощью специальных циклов регенерации на магистрали. Временная диаграмма цикла регенерации покказана на рис. 1.6. Эти циклы выполняет входящий в состав материнской платы компьютера контроллер регенерации, который должен для этого получать управление магистралью каждые 15 микросекунд. Во время цикла регенерации производится чтение одной из 256 ячеек памяти (для адресации используются только восемь младших разрядов адреса SA0...SA7). При этом читаемая информация нигде не используется, то есть это цикл псевдочтения. Проведение 256 циклов регенерации, то есть псевдочтение из 256 последовательных адресов ОЗУ, обеспечивает полное освежение информации в ОЗУ и ее непрерывное сохранение. Если по каким-то причинам цикл регенерации не производится вовремя, то возмлжна утеря информации в ОЗУ. Цикл регенерации включает в себя выставление сигналов -REFRESH, адреса SA0...SA7 и -MEMR. В случае необходимости может использоваться сигнал I/O CH RDY.

1.2.4. Электрические характеристики линий ISA.

При проектировании УС помимо протоколов обмена по магистрали надо учитывать также электрические характеристики сигналов. Стандарт магистрали определяет требования к входным и выходным токам приемников и источников сигнала каждой из плат расширения. Несоблюдение этих требований может нарушить функционирование всего компьютера и даже вывести его из строя.

Выходные каскады передатчиков магистральных сигналов УС должны выдавать ток низкого уровня не меньше 24 мА (это относится ко всем типам выходных каскадов), а ток высокого уровня — не меньше 3 мА (для выходов с тремя состояниями и ТТЛ).

Входные каскады приемников магистральных сигналов должны потреблять входной ток низкого уровня не больше 0,8 мА, а входной ток высокого уровня — не больше 0,04 мА.

Кроме этого необходимо учитывать, что максимальная длина печатного проводника от контакта магистрального разъема до вывода микросхемы не должна превышать 65 миллиметров, а максимальная емкость относительно земли по каждому контакту магистрального разъема не должна быть больше 20 пФ.

К некоторым линиям магистрали подключены нагрузочные резисторы, идущие на шину питания +5 В. К линиям -IOR, -IOW, -MEMR, -MEMW, -SMEMR, -SMEMW, -I/O CH CK подключены резисторы 4,7 кОм, к линиям -I/O CS 16, -MEM CS 16, -REFRESH, -MASTER, -OWS — 300 Ом, а к линии I/O CH RDY — 1 кОм. Кроме того к некоторым линиям магистрали подключены последовательные резисторы: к линиям -IOR, -IOW, -MEMR, -MEMW, -SMEMR, -SMEMW и OSC — резисторы номиналом 22 Ом, а к линии SYSCLK — 27 Ом.

1.3. Порядок обмена по интерфейсу Centronics.

Основным назначением интерфейса Centronics (аналог — ИРПР-М) является подключение к компьютеру принтеров различных типов. Поэтому распределение контактов разъема, назначение сигналов, программные средства управления интерфейсом ориентированы именно на это использование. В то же время, с помощью данного интерфейса можно подключать к компьютеру и другие внешние устройства, имеющие разъем Centronics, а также специально разработанные УС.

Основным достоинством использования Centronics для подключения УС по

сравнению с ISA является значительно меньший риск вывести компьютер из строя. Главный недостаток этого подхода — значительно меньшая скорость обмена. Назначение 36 контактов разъема Centronics приведено в таблице 1.10.

Сигналы Centronics имеют следующее назначение (тип выходных каскадов для всех сигналов — ТТЛ):

D0...D7 — 8-разрядная шина данных для передачи из компьютера в принтер. Логика сигналов положительная.

-STROBE — сигнал стробирования данных. Данные действительны как по переднему, так и по заднему фронту этого сигнала. Сигнал говорит приемнику (принтеру), что можно принимать данные.

-ACK — сигнал подтверждения принятия данных и готовности приемника (принтера) принять следующие данные. То есть здесь реализуется асинхронный обмен.

BUSY — сигнал занятости принтера обработкой полученных данных и неготовности принять следующие данные. Активен также при переходе принтера в состояние off-line или при ошибке, а также при отсутствии бумаги. Компьютер начинает новый цикл передачи только после снятия -ACK и после снятия BUSY.

Контакт разъема компьютера	Цепь	I/O	Контакт разъема принтера
1	-STROBE	O	1
2	D0	O	2
3	D1	O	3
4	D2	O	4
5	D3	O	5
6	D4	O	6
7	D5	O	7
8	D6	O	8
9	D7	O	9
10	-ACK	I	10
11	BUSY	I	11
12	PE	I	12
13	SLCT	I	13
14	-AUTO FD	O	14
15	-ERROR	I	32
16	-INIT	O	31
17	-SLCT IN	O	36
18...25	GND	-	16, 17, 19...30, 33

Табл. 1.10. Назначение контактов разъемов Centronics (I — входной сигнал компьютера, O — выходной сигнал)

-AUTO FD — сигнал автоматического перевода строки. Получив его, принтер переводит каретку на следующую строку.

Остальные сигналы не являются, вообще говоря, обязательными.

PE — сигнал конца бумаги. Получив его, компьютер переходит в режим ожидания. Если в принтер вставить лист бумаги, то сигнал снимается.

SLCT — сигнал готовности приемника. С его помощью принтер говорит о том, что

он выбран и готов к работе. У многих принтеров имеет постоянно высокий уровень.

-SLCT IN — сигнал принтеру о том, что он выбран, и последует передача данных.

-ERROR — сигнал ошибки принтера. Активен при внутренней ошибке, переходе принтера в состояние off-line или при отсутствии бумаги. Как видим здесь многие сигналы дублируют друг друга.

-INIT — сигнал инициализации (сброса) принтера. Его длительность не менее 2,5 мкс. Поисходит очистка буфера печати.

Временная диаграмма цикла передачи данных представлена на рисунке 1.7.

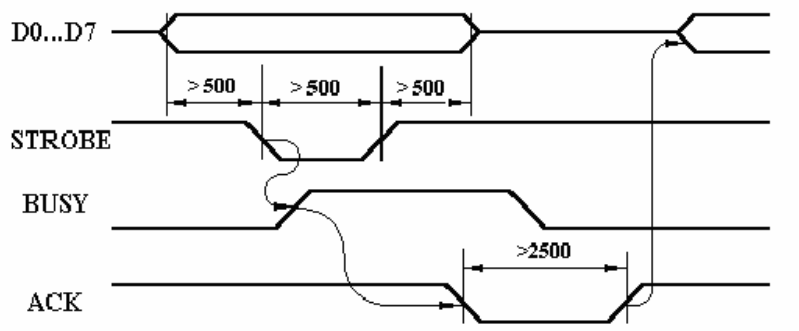


Рис. 1.7. Временные диаграммы цикла передачи данных в Centronics (все временные интервалы в наносекундах)

Перед началом цикла передачи данных компьютер должен убедиться, что сняты сигналы BUSY и -ACK. После этого выставляются данные, формируется строб, снимается строб, и снимаются данные. Принтер должен успеть принять данные с выбранным темпом. При получении строба принтер формирует сигнал BUSY, а после окончания обработки данных выставляет сигнал -ACK, снимает BUSY и снимает -ACK. Затем может начинаться новый цикл.

Все сигналы интерфейса Centronics передаются в уровнях ТТЛ и рассчитаны на подключение одного стандартного входа ТТЛ. Максимальная длина соединительного кабеля по стандарту — 1,8 м.

Как видно из таблицы 1.10, при использовании интерфейса Centronics для подключения к компьютеру произвольных УС мы можем использовать 17 линий, назначение которых можно выбирать по своему усмотрению.

Формирование и прием сигналов интерфейса Centronics производится путем записи и чтения выделенных для него портов ввода/вывода. В компьютере может использоваться три порта Centronics, обозначаемых LPT1 (базовый адрес 378h), LPT2 (базовый адрес 278h) и LPT3 (базовый адрес 3BCh). При этом LPT3 используется в том случае, когда контроллер принтера находится на плате графического адаптера Hercules или EGA. Прерывания портов принтеров (IRQ5 для LPT2 и IRQ7 для LPT1) используются очень редко.

Базовый адрес порта используется для передачи принтеру байта данных. Установленные на линиях данные можно считать из этого же порта.

Следующий адрес (базовый + 1) служит для чтения битов состояния принтера (бит 3 соответствует сигналу -ERROR, бит 4 — сигналу SLCT, бит 5 — сигналу PE, бит 6 — сигналу -ACK, бит 7 — сигналу BUSY). Последний используемый адрес (базовый + 2) используется для записи битов управления принтером (бит 0 соответствует сигналу -STROBE, бит 1 — сигналу -AUTO FD, бит 2 — сигналу -INIT, бит 3 — сигналу -SLCT IN и наконец бит 4, равный единице, разрешает прерывание от принтера).

1.4. Порядок обмена по интерфейсу RS-232C.

Интерфейс RS232-C предназначен для подключения к компьютеру стандартных внешних устройств (принтера, сканера, модема, мыши и др.), а также для связи компьютеров между собой. Основными преимуществами использования RS-232C по сравнению с Centronics являются возможность передачи на значительно большие расстояния и гораздо более простой соединительный кабель. В то же время работать с ним несколько сложнее. Данные в RS-232C передаются в последовательном коде побайтно. Каждый байт обрамляется стартовым и стоповыми битами. Данные могут передаваться как в одну, так и в другую сторону (дуплексный режим).

Компьютер имеет 25-контактный (DB25P) или 9-контактный (DB9P) разъем для подключения RS-232C. Назначение контактов разъема приведено в таблице 1.11.

Цепь	Контакт (25-контактный разъем)	Контакт (9-контактный разъем)	I/O
FG	1	-	-
-T x D	2	3	O
-R x D	3	2	I
RTS	4	7	O
CTS	5	8	I
DSR	6	6	I
SG	7	5	-
DCD	8	1	I
DTR	20	4	O
RI	22	9	I

Табл. 1.11. Назначение контактов разъемов интерфейса RS-232C (I — входной сигнал компьютера, O — выходной сигнал)

Назначение сигналов следующее.

FG — защитное заземление (экран).

-TxD — данные, передаваемые компьютером в последовательном коде (логика отрицательная).

-RxD — данные, принимаемые компьютером в последовательном коде (логика отрицательная).

RTS — сигнал запроса передачи. Активен во все время передачи.

CTS — сигнал сброса (очистки) для передачи. Активен во все время передачи. Говорит о готовности приемника.

DSR — готовность данных. Используется для задания режима модема.

SG — сигнальное заземление, нулевой провод.

DCD — обнаружение несущей данных (детектирование принимаемого сигнала).

DTR — готовность выходных данных.

RI — индикатор вызова. Говорит о приеме модемом сигнала вызова по телефонной сети.

Наиболее часто используются трех- или четырехпроводная связь (для двунаправленной передачи). Схема соединения для четырехпроводной линии связи показана на рисунке 1.8.

Для двухпроводной линии связи в случае только передачи из компьютера во внешнее устройство используются сигналы SG и TxD. Все 10 сигналов интерфейса задействуются только при соединении компьютера с модемом.

Формат передаваемых данных показан на рисунке 1.9. Собственно данные (5, 6, 7 или 8 бит) сопровождаются стартовым битом, битом четности и одним или двумя стоповыми битами. Получив стартовый бит, приемник выбирает из линии биты данных через определенные интервалы времени. Очень важно, чтобы тактовые частоты приемника и передатчика были одинаковыми (допустимое расхождение — не более 10%). Скорость передачи по RS-232C может выбираться из ряда: 110, 150, 300, 600, 1200, 2400, 4800, 9600, 19200, 38400, 57600, 115200 бит/с.

Все сигналы RS-232C передаются специально выбранными уровнями, обеспечивающими высокую помехоустойчивость связи (рис. 1.10). Отметим, что данные передаются в инверсном коде (логической единице соответствует низкий уровень, логическому нулю — высокий уровень).

Для подключения произвольного УС к компьютеру через RS-232C обычно используют трех- или четырехпроводную линию связи (рис. 1.8), но можно задействовать и другие сигналы интерфейса.

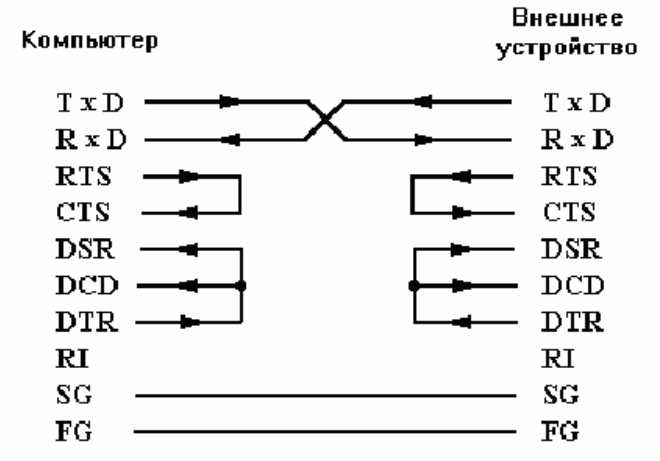


Рис. 1.8. Схема 4-проводной линии связи для RS-232C

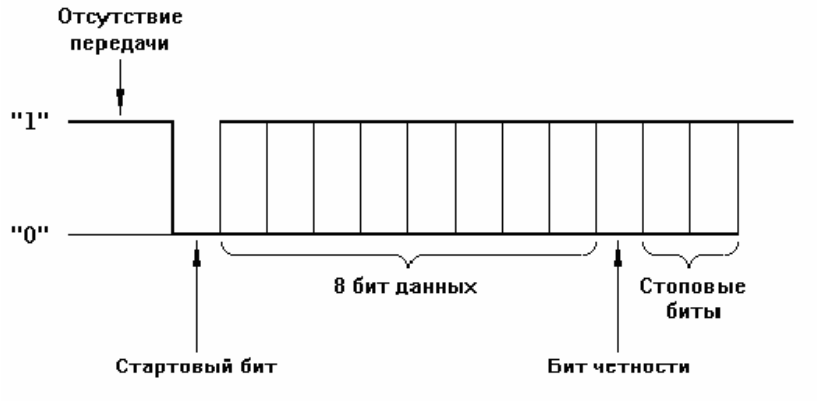


Рис. 1.9. Формат данных RS-232C

Обмен по RS232-C осуществляется с помощью обращений по специально выделенным для этого портам COM1 (адреса 3F8h...3FFh, прерывание IRQ4), COM2 (адреса 2F8h...2FFh, прерывание IRQ3), COM3 (адреса 3E8h...3EFh, прерывание IRQ10), COM4 (адреса 2E8h...2EFh, прерывание IRQ11). Форматы обращений по этим адресам можно найти в многочисленных описаниях микросхем контроллеров последовательного обмена UART (Universal Asynchronous Receiver/Transmitter), например, i8250, КР580ВВ51. Здесь же мы не имеем возможности описывать все возможные режимы их работы.

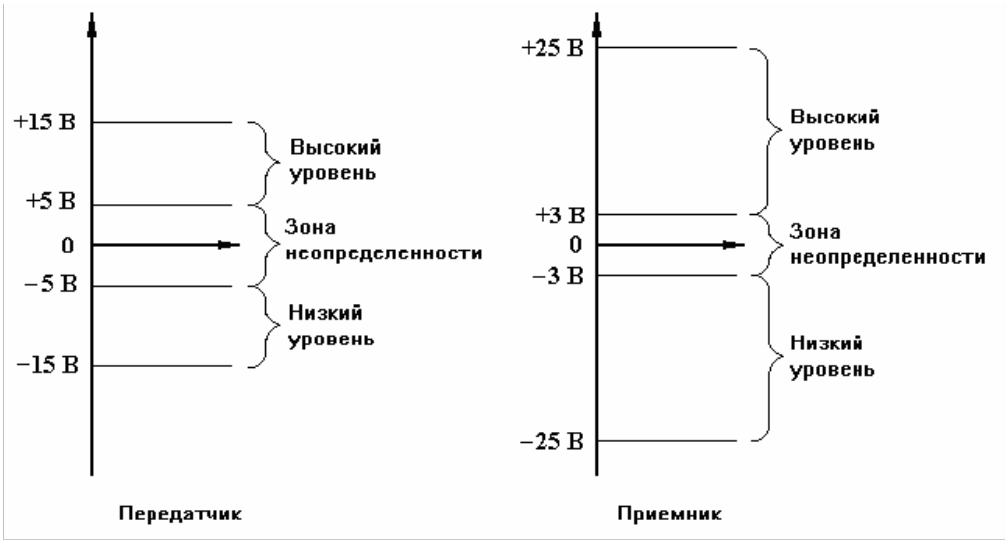


Рис. 1.10. Уровни сигналов RS-232C на передающем и принимающем концах линии связи.

Глава 2
Разработка устройств сопряжения для ISA

При разработке УС необходимо в первую очередь сформулировать требования, предъявляемые к нему, проанализировать функции,которые компьютер должен выполнять с помощью данного УС. В общем случае эти функции могут быть реализованы как аппаратурой УС, так и программным обеспечением компьютера. В свою очередь, УС может включать в себя процессор, однокристальный контроллер, микропрограммный автомат или жесткую логику. Сочетания перечисленных решений различаются между собой стоимостью, сложностью решаемых задач, гибкостью (способностью перенастройки на другую задачу), быстродействием (рис. 2.1).

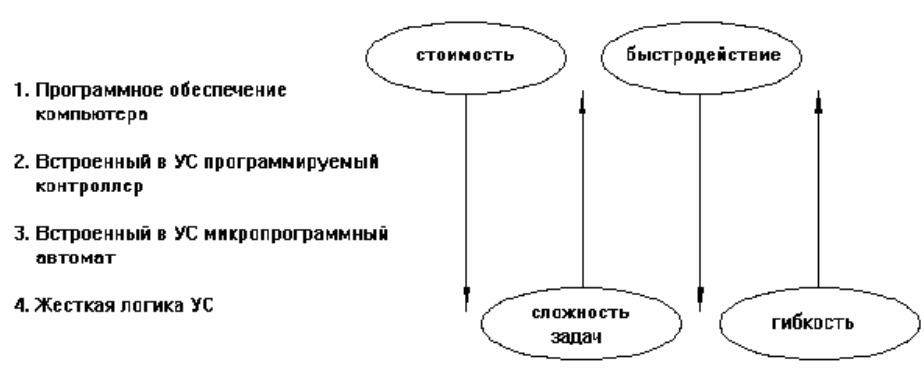


Рис. 2.1. Сравнение методов реализации функций УС

Отметим, что изменение каждого параметра показано на рисунке в предположении, что остальные параметры не изменяются. При выборе того или иного решения надо учитывать также степень трудоемкости его реализации. Очень часто значительные трудности вызывает составление программного обеспечения для встроенного контроллера или микропрограммного обеспечения для встроенного микропрограммного автомата. Для этого обычно требуются сложные системы разработки. В то же время, проектирование жесткой логики требует большого мастерства по части схемотехники. Вместе с тем, при прочих равных условиях представляется наиболее разумным сочетание программных средств компьютера и жесткой логики УС с перенесением сложной обработки на компьютер и достижением высокого быстродействия с помощью жесткой логики. К сожалению, это удается не всегда.

Функции, выполняемые УС, можно разделить на две группы. К первой группе относятся интерфейсные функции, то есть те, которые обеспечивают обмен с выбранным интерфейсом компьютера (ISA, Centronics, RS-232C или какие-нибудь еще). Вторую группу образуют операционные или основные функции, ради которых собственно и создается УС. Строго говоря, если УС предназначено для сопряжения компьютера с каким-нибудь другим устройством, также имеющим стандартный интерфейс, то добавляются еще и функции обмена с этим интерфейсом, но мы о них говорить не будем.

Соответственно с этими двумя выделенными группами функций в структуре УС можно также выделить две части: интерфейсную и операционную. При этом подходы к проектированию этих двух имеют принципиальные отличия. Ведь операционные части различных УС могут быть самыми разнообразными. Здесь простор для творчества,

нестандартных решений, оптимизации и экспериментирования с целью наилучшего решения той уникальной задачи, для которой создается УС. А вот интерфейсные части практически у всех УС одинаковы или очень похожи между собой, так как интерфейсные функции жестко определяются протоколом выбранного стандартного интерфейса. Конечно, интерфейсные части могут быть более или менее сложными в зависимости от задачи, решаемой УС, но все-таки все они состоят из одного и того же набора блоков и узлов, реализующих одинаковые функции и строящиеся, как правило, по стандартным схемам. Поэтому основное внимание в этой и следующих главах будет уделено проектированию именно интерфейсных частей, однако будут также рассмотрены несколько конкретных схем УС, имеющих наиболее характерные операционные части.

Прежде чем перейти к методам построения отдельных блоков интерфейсных частей УС, ориентированных на ISA, надо выделить основные интерфейсные функции, определяемые стандартом магистрали. В соответствии с определением интерфейса, приведенным в предыдущей главе, мы должны обеспечить информационную, электрическую и конструктивную совместимость. О конструктивной совместимости мы здесь говорить не будем (она относится к этапу проектирования печатной платы и сводится к точному соблюдению всех размеров платы, разъемов и крепежных элементов). Информационная совместимость предполагает точное выполнение протоколов обмена и правильное использование сигналов магистрали. Электрическая совместимость подразумевает согласование уровней входных, выходных и питающих напряжений и токов.

2.1. Проектирование аппаратуры для сопряжения с ISA

При проектировании узлов УС, особенно входящих в интерфейсную часть УС, необходимо учитывать временные диаграммы ISA (рис. 1.3 — рис. 1.6). Наиболее важными при проектировании УС, работающих как устройства ввода/вывода, являются следующие временные интервалы:

— задержка между выставлением адреса и передним фронтом строба обмена (не менее 91 нс) — определяет время распознавания своего адреса проектируемым УС;

— длительность строба обмена (не менее 176 нс);

— задержка между передним фронтом сигнала -IOR и выставлением УС читаемых данных (не более 110 нс) — определяет требования к быстродействию буфера данных УС;

— задержка между задним фронтом сигнала -IOW и снятием записываемых данных (не менее 30 нс) — определяет требования к быстродействию принимающих данные узлов УС.

При работе УС в циклах обмена с памятью берутся аналогичные временные интервалы из рис. 1.4.

2.1.1. Буферирование сигналов магистрали

Буферирование магистральных сигналов применяется для электрического согласования и выполняет две основные функции: электрическая развязка (для всех сигналов) и передача сигналов в нужном направлении (только для двунаправленных сигналов). Это первая и наиболее очевидная интерфейсная функция любого УС. Иногда с помощью буферирования реализуется также мультиплексирование сигналов. Для буферирования наиболее часто используются микросхемы мигистральных приемников, передатчиков, приемопередатчиков, называемые также нередко буферами или драйверами.

Электрическая развязка подразумевает обеспечение нужных входных и выходных токов (уровни напряжения на ISA — ТТЛ). Как уже упоминалось, входные каскады УС должны обеспечивать уровень входного тока не более 0,8 мА, а выходные и двунаправленные каскады должны выдавать выходной ток не менее 24 мА (при нулевом выходном сигнале). Несоблюдение этого правила может привести к сбоям в работе компьютера и даже к выходу из строя его отдельных узлов. При этом, строго говоря, все определяется конфигурацией системы. Если к магистрали компьютера подключена только одна плата расширения (ваше УС), то требования к ней будут гораздо мягче, чем в случае использования нескольких плат. Но всегда надо рассчитывать на возможность развития системы и включения дополнительных плат. Поэтому лучше все-таки придерживаться указанных величин.

Выбор типа драйвера для каждого магистрального сигнала (приемник, передатчик или приемопередатчик) определяется назначением этого сигнала и возможными режимами работы УС. Так, например, в случае, когда УС работает в режиме программного обмена, приемники используются для сигналов адреса SA0 ... SA9 и для управляющих сигналов -IOR, -IOW, AEN, BALE, -SBHE, передатчики используются для I/O CH RDY и -I/O CS 16. Для сигналов данных могут использоваться приемники (если УС работает только в режиме записи), передатчики (если УС работает только в режиме чтения) или приемопередатчики (если УС работает как в режиме чтения, так и в режиме записи). Если возможен обмен по прерываниям, то добавляется передатчик для сигнала IRQ, а если применяется ПДП, то применяется передатчик для сигнала DRQ и приемник для сигнала DACK.

Остановимся подробнее на характеристиках микросхемам, которые могут применяться для буферирования.

Приемники магистральных сигналов должны удовлетворять двум основным требованиям: малые входные токи и высокое быстродейтсвие (они должны успевать отрабатывать в течение отведенных им временных интервалов циклов обмена). Конкретное значение допустимых времен задержек определяется используемой схемой интерфейсной части УС в целом, но можно определенно сказать, что микросхемы обычных (не быстродействующих) КМОП серий здесь не годятся, несмотря на их малые входные токи. Не подходят и микросхемы серии К155 (SN74) из-за их больших входных токов.

Требованиям, предъявляемым к приемникам, удовлетворяют следующие серии микросхем: КР1533 (SN74ALS), К555 (SN74LS) и КР1554 (74AC). Величины входных токов логического нуля для них составляют соответственно 0,2 мА, 0,4 мА и 0,2 мА, а величины временных задержек не привышают соответственно 15 нс, 20 нс и 10 нс. Помимо этих серий в качестве приемников можно использовать специальные микросхемы магистральных приемников серии КР559 (входной ток не более 0,12 мА, задержка не более 30 нс). Требованиям, предъявляемым к приемникам, удовлетворяют также микросхемы электрически программируемых ППЗУ и ПЛМ серии КР556 (I36, N82S, DM87S, HM76). Это тоже немаловажно, так как их очень удобно использовать в схемах селекторов адреса УС. Входные токи этих микросхем не превышают 0,25 мА. Пример входного буфера показан на рис. 2.2.

Отметим, что малые входные токи микросхем серий КР1533 и КР1554 позволяют подключать к линии магистрали даже два входа таких микросхем.

Теперь переходим к передатчикам. Требования к ним: большой выходной ток и высокое быстродействие. Часто они должны иметь также отключаемый выход (например, для шины данных), то есть иметь выход с открытым коллектором или с тремя состояниями. Это связано с необходимостью перехода УС в пассивное состояние в случае отсутствия обращения к нему. Выбор микросхем передатчиков гораздо больше, такие микросхемы есть практически в каждой серии (К155, К555, КР1533, К559 и т.д.).

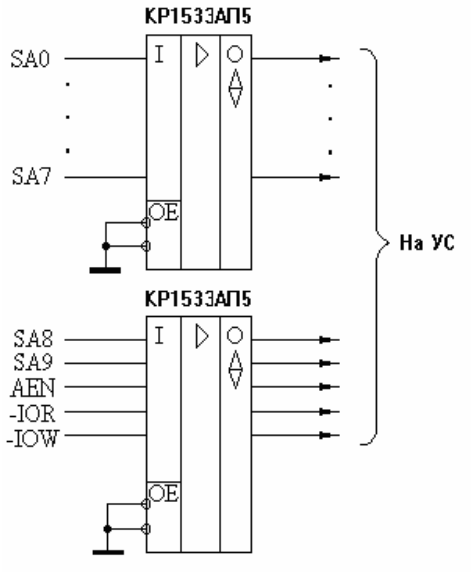


Рис. 2.2. Пример входного буфера

Передатчики часто выполняют функцию мультиплексирования данных, которые должны поступать на шину данных ISA от различных источников. На рис. 2.3. упрощенно показано два наиболее распространенных подхода к решению данный задачи (для 8-разрядной шины данных). Отметим, что при использовании микросхем мультиплексоров надо брать те из них, которые имеют выходы с тремя состояниями и большие выходные токи.

И, наконец, приемопередатчики. Требования к ним включают в себя требования к приемникам и передатчикам, то есть малый входной ток, большой выходной ток, высокое быстродействие и обязательное отключение выходов. Надо отметить, что в простейшем случае (когда разрядов немного) приемопередатчики могут быть построены на микросхемах приемников и передатчиков с отключаемыми выходами. Однако при большом количестве разрядов надо использовать специальные микросхемы приемопередатчиков. Эти микросхемы бывают двух основных типов (рис. 2.4.): с двумя двунаправленными шинами или с тремя шинами (одной двунаправленной, одной входной шиной и одной выходной шиной).

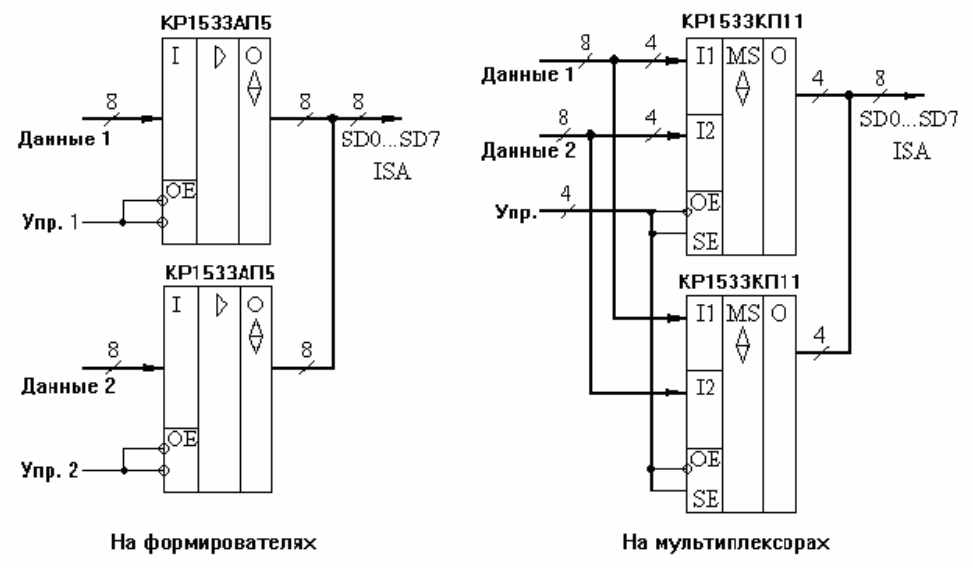


Рис. 2.3. Мультиплексирование шины данных

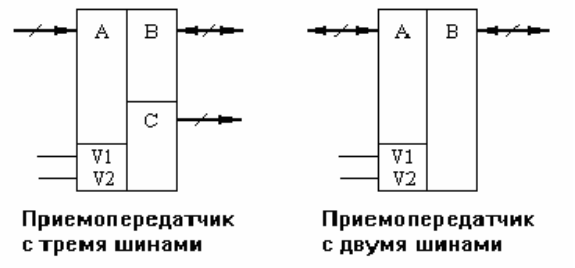


Рис. 2.4. Типы приемопередатчиков

	Шина A	Шина B	Шина C
КР559ИП3 (DS8641) 50 нс	Вход, 4 разряда, 1,8 мА	Вход/выход, ОК, 4 разряда, 0,2/80 мА	Выход, ТТЛ, 4 разряда,16 мА
КР531АП2 40 нс	Вход, 4 разряда, 0,15 мА	Вход/выход, ОК, 4 разряда, 0,15/60 мА	Выход, ОК, 4 разряда,60 мА
К589АП16 (I8216) 30 нс	Вход, 4 разряда, 0,25 мА	Вход/выход, 3С, 4 разряда, 0,25/50 мА	Выход, 3С, 4 разряда,15 мА
КР580ВА86 (I8286) 30 нс	Вход/выход, 3С, 8 разрядов, 0,2/16 мА	Вход/выход, 3С, 8 разрядов, 0,2/32 мА	—
КР1533АП6 (SN74ALS245) 10 нс	Вход/выход, 3С, 8 разрядов, 0,1/30 мА	Вход/выход, 3С, 8 разрядов, 0,1/30 мА	—
КР559ИП13 (DP8307) 18 нс	Вход/выход, 3С, 8 разрядов, 0,35/100 мА	Вход/выход, 3С, 8 разрядов, 0,35/100 мА	—

Назначение и параметры шин

V2	V1	ИП3	АП2	АП16	ВА86	АП6	ИП13
0	0	A=>B	A=>B, B=>C, запр.	A=>B	B=>A	B=>A	запр.
1	0	B=>C	A=>B	B=>C	A=>B	A=>B	A=>B
0	1	B=>C	B=>C	откл.	откл.	откл.	B=>A
1	1	B=>C	откл.	откл.	откл.	откл.	откл.

Управление

Табл. 2.1. Микросхемы приемопередатчиков

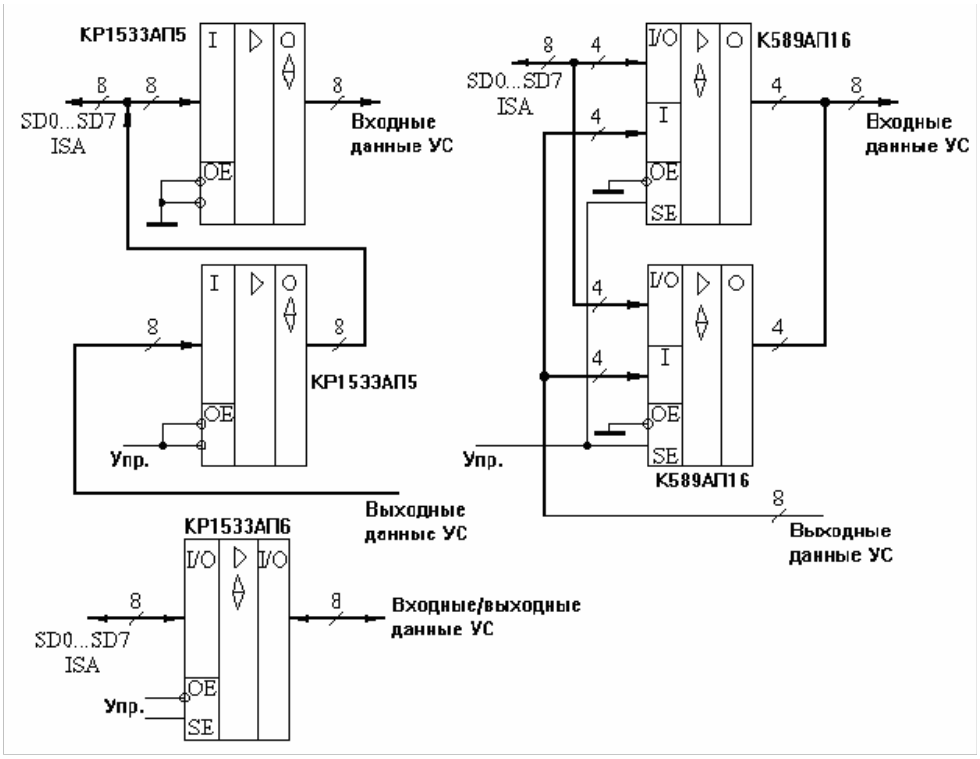


Рис. 2.5. Варианты построения приемопередатчиков данных

Для управления работой приемопередатчиков используются два управляющих сигнала. Характеристики некоторых приемопередатчиков сведены в таблицу 2.1. В ней указаны разрядность шин, величины задержек и входных/выходных токов всех шин микросхем, а также приведены режимы работы в зависимости от управляющих сигналов. Отметим такую особенность микросхемы КР59ИП3, как невозможность одновременного отключения ее двунаправленной и выходной шин. В таблице использованы следующие обозначения: ОК — выход с открытым коллектором, 3С — выход с тремя состояниями. Отметим, что если приемопередатчики с открытым коллектором используются для буферирования шины данных, то на их выходах необходимо включать резисторы на шину +5В (если они не работают на линию, к которой эти резисторы уже подключены). Поэтому их применение иногда оказывается нежелательным. Это однако совсем не означает, что они не могут быть использованы, например, в операционной части УС. Особенностью микросхемы КР580ВА86 (87) является то, что шины имеют различные выходные токи, и

только одна из них (B) удовлетворяет требованиям стандарта ISA. У других микросхем все двунаправленные шины выдают требуемые выходные токи. Те или иные сигналы управления могут быть более или менее удобны в каждом конкретном случае.

На рис. 2.5. показаны три схемы реализации приемопередатчиков для шины данных: на приемнике и передатчике, на приемопередатчике с двумя шинами и на приемопередатчике с тремя шинами (для 8-разрядных данных).

Отметим, что чаще нужны приемопередатчики с раздельными входными и выходными шинами данных УС, но при использовании многоразрядных микросхем ОЗУ или сдвиговых регистров типа КР1533ИР24 (SN74ALS299), которые имеют двунаправленную шину данных, удобнее применять приемопередатчики с совмещенными входными/выходными данными УС.

2.1.2. Построение селекторов адреса

Второй основной интерфейсной функцией, выполняемой УС, работающими в режиме программного обмена, является селектирование или дешифрация адреса. Эту функцию выполняет узел, называемый селектором адреса, который должен выработать сигналы, соответствующие выставлению на шине адреса магистрали кода адреса, принадлежащего данному УС, или одного из зоны адресов данного УС. Обобщенная схема селектора адреса для УС, работающего как устройство ввода/вывода, показана на рис. 2.6. Здесь шина A — это шина адреса магистрали, шина AS — внутренняя шина УС, на которой присутствуепт код, сравниваемый с адресом магистрали (может отсутствовать), ADR — выходные сигналы селектора адреса, формируемые при обращении по магистрали к данному УС.

Отметим, что совсем не обязательно дешифровать все линии адресной шины магистрали. Довольно часто для упрощения схемы УС удобно часть этих линий отбросить, не заводить на селектор адреса. При этом важно, чтобы адреса проектируемого УС не перекрывались с адресами, занятыми другими устройствами компьютера. Наиболее часто отбрасывают младшие разряды адреса.

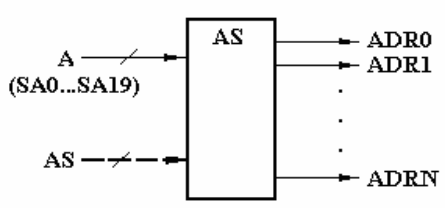


Рис. 2.6. Структура селектора адреса

Рассмотрим небольшой пример. Пусть мы выбрали для нашего УС свободную зону из 16 адресов в пространстве устройств ввода/вывода 360 ... 36F. Пусть наше УС должно иметь 4 адреса для 8-разрядного обмена. Тогда мы можем каждому адресу УС поставить в соотвествие четыре магистральных адреса (то есть шестнадцать адресов выбранной зоны разделили на четыре адреса УС и получили четыре). Тогда на селектор адреса можно завести не 10, а только 8 адресных линий (SA2 ... SA9), отбросив два младших адреса. При этом, например, первому адресу УС будут соответствовать магистральные адреса 360 ... 363. При обращении к любому из них селектор адреса будет распознавать первый адрес УС.

Однако при данном подходе надо соблюдать осторожность и не захватывать слишком больших зон адресов, так как иначе может не остаться возможностей для

расширения системы. Как уже отмечалось, по стандарту ISA, устройства ввода/вывода адресуются 16 разрядами адресной шины SA0 ... SA15, но большинство плат расширения работают только с SA0 ... SA9, поэтому обычно нет смысла обрабатывать разряды SA10 ... SA15. Однако иногда разрабатываемое УС должно иметь очень много адресов. В таком случае оно может дешифровать все 16 разрядов, но свободными будут не все дополнительные адреса, а только окна, соответствующие свободным зонам в 1К-байтном пространстве 000 ... 3FF. Например, свободному окну 300 ...31F в 64К-байтном пространстве (0000 ... FFFF) будут соответствовать свободные окна 0300 ... 031F, 0F00 ... 0F1F, 1300 ... 131F, 1F00 ... 1F1F и т.д. (всего 64 окна). Так что советуем подумать, следует ли выбирать этот путь, существенно усложняющий селектор адреса.

Несколько слов о селекторе адреса для УС, работающего в адресном пространстве памяти. В этом случае мы должны обрабатывать 20 разрядов адресной шины (при полном объеме памяти до 1 Мбайта) или все 24 разряда адресной шины (при полном объеме памяти до 16 Мбайт). Надо сказать, что разработка УС, работающего как устройство ввода/вывода гораздо проще. Переход в адресное пространство памяти вызывается обычно необходимостью ускорения обмена с внутренним ОЗУ или ПЗУ, входящим в состав УС. Но в этом случае селектор адреса не должен обрабатывать столько младших разрядов адреса, сколько адресных входов имеет это ОЗУ или ПЗУ. Например, если внутреннее ОЗУ имеет организацию 1К x 8 (десять адресных входов), то десять младших разрядов адреса SA0 ... SA9 должны подаваться не на селектор адреса, а (через соответствующие буфера) непосредственно на адресные входы ОЗУ. Разряды адреса LA17...LA23 перед подачей на селектор адреса должны быть зафиксированы на все время цикла обмена (рис. 2.7). Отметим, что при использовании микросхемы регистра с малыми входными токами можно обойтись без входных буферов как для сигналов LA17...LA23, так и для сигнала ALE.

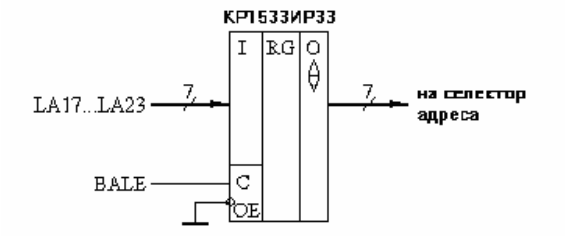


Рис. 2.7. Обработка сигналов LA17 ... LA23

Помимо сигналов, показанных на рис. 2.6, на селектор адреса часто подают сигнал AEN, который при этом используется для запрещения выработки выходных сигналов. То есть если по магистрали идет прямой доступ к памяти, то устройство ввода/вывода (в нашем случае — УС) должно быть обязательно отключено от магистрали и не должно реагировать на выставляемые на шине адреса коды (пока мы говорим об УС, ориентированных только на программный обмен).

А теперь рассмотрим несколько наиболее характерных схемотехнических решений селекторов адреса. Но сначала отметим требования, предъявляемые к ним:

— высокое быстродействие (селектор адреса должен иметь задержку не более, чем интервал между выставлением адреса и началом сигнала строба обмена);

— возможность изменения селектируемых адресов (особенно важно для устройств ввода/вывода из-за малого количества свободных адресов);

— малые аппаратурные затраты.

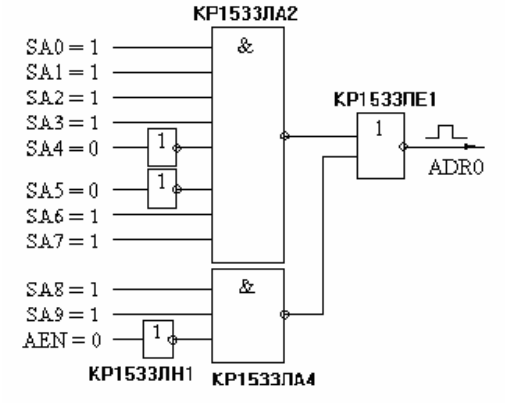


Рис. 2.8. Селектор адреса на логических элементах

Самое простое решение при построении селектора адреса — использование только микросхем логических элементов. Например, на рис. 2.8 показана схема, реагирующая на единственный адрес 3CF. Основным достоинством такого подхода является высокое быстродействие (для схемы на рис. 2.8 задержка не превышает 30 нс). При использовании микросхем с малыми входными токами можно обойтись без буферов. Но есть и недостатки: необходимость проектирования схемы заново для каждого нового адреса, невозможность смены адреса, сложность организации выбора нескольких адресов. Если надо иметь возможность изменять выбираемый адрес, то можно предусмотреть использование отключаемых инверторов для всех линий адреса. Тогда, подключая или отключая нужные инверторы с помощью перемычек или переключателей, мы получаем возможность перестраивать в некоторых пределах наш селектор адреса. Другой путь — применение элементов "Исключающее ИЛИ", работающих как управляемые инверторы. На рис. 2.9 показан тот же, что и на рис. 2.8 селектор адреса, но выбирающий в зависимости от кода на шине AS, задаваемого перемычками, адреса 3CF, 2CF, 1CF, 0CF и т.д. (всего 8 возможных адресов).

Селекторы адреса могут быть реализованы также на микросхемах дешифраторов. Вообще говоря, можно построить селектор адреса только на этих микросхемах, но объем аппаратуры получается при этом очень большим. Поэтому более правильным решением будет обработка старших адресных разрядов какой-то другой схемой (например, одним или несколькими логическими элементами), а младшие — с помощью одной микросхемы дешифратора. Примером может служить селектор адреса на рис. 2.10, сигналы на выходах которого соответствуют выбору шестнадцати адресов в пределах зоны, задаваемой другой частью схемы (обозначена AS). Совсем не обязательно использовать дальше все сигналы ADR0 ... ADR15, можно с помощью перемычек применять их для изменения адресов нашего УС. Отметим такое достоинство этого подхода по сравнению с рассмотренным ранее как возможность селектирования нескольких адресов.

Следующий метод реализации селектора адреса — использование микросхем компараторов кодов, на одну входную шину которых подается адрес из магистрали, а на другую входную шину — код AS, соответствующий селектируемому адресу. Очевидно, что каскадируя эти микросхемы, можно построить селектор адреса исключительно на них, но это приведет к неоправданным аппаратурным затратам. Гораздо эффективнее применять компараторы кодов для изменения селектируемых адресов. На рис. 2.11 показана схема селектора адреса с использованием компаратора кодов и дешифратора. Здесь разряды SA0 .... SA2 определяют один из восьми адресов УС, SA3 ... SA5 жестко должны быть равными единице, а значения SA6 ... SA9 выбираются переключателями. Отметим, что время задержки этой схемы — не более 57 нс.

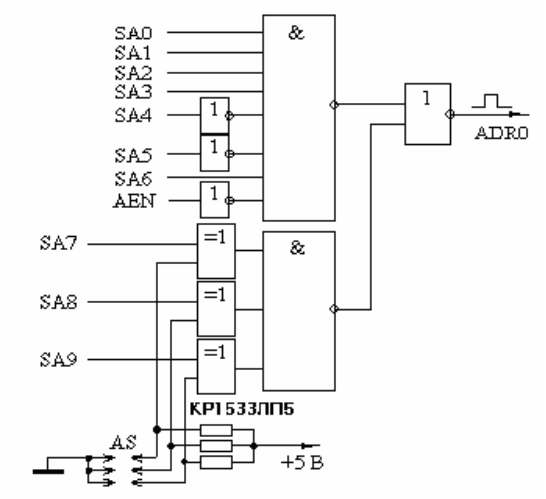


Рис. 2.9. Использование элементов " Исключающее ИЛИ" для изменения селектируемого адреса

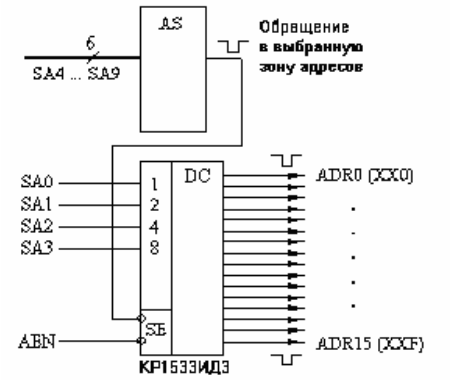


Рис. 2.10. Селектор адреса с использованием микросхемы дешифратора

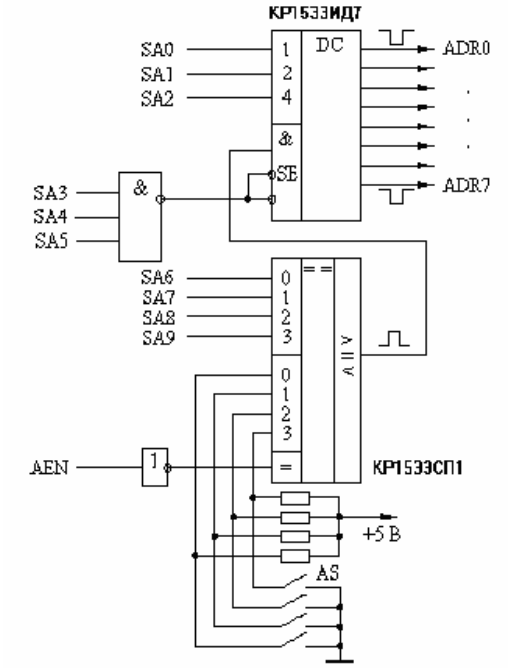


Рис. 2.11. Использование компаратора кода для изменения селектируемого адреса

Наконец, наиболее универсальными являются селекторы адреса на базе ППЗУ и ПЛМ. В данном случае селектируемый адрес (или селектируемые адреса) зависит не от схемотехнических решений и не от кода, задаваемого переключателями, а от прошивки ППЗУ или ПЛМ. Такой подход обеспечивает, как правило, малые аппаратурные затраты, а также простую реализацию выбора нескольких адресов или зон адресов. Изменить селектируемый адрес (или адреса) можно заменой ППЗУ (ПЛМ), устанавливаемого в контактирующее устройство (сокет). Однако это может сделать только пользователь, имеющий набор ППЗУ (ПЛМ) для разных адресов или имеющий программатор (устройство для программирования). На рис. 2.12 показана схема селектора адреса на одной микросхеме ППЗУ (нулевой разряд адреса SA0 не задействован, а SA9 всегда должен быть равен нулю).

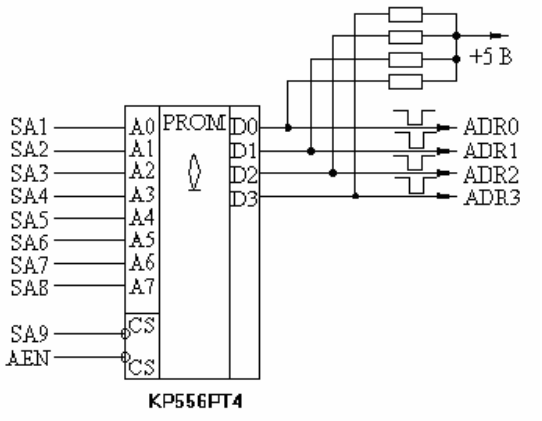


Рис. 2.12. Селектор адреса на ППЗУ

Возможно также комбинирование ППЗУ (ПЛМ) с другими микросхемами, например, с дешифраторами или компараторами кодов. Как уже отмечалось, малые

входные токи микросхем ППЗУ серии КР556 позволяют отказаться от входных буферов адреса. Задержка микросхем ППЗУ (ПЛМ) этой серии не превышает 50 ... 80 нс. Если необходимо обрабатывать больше разрядов адреса, чем имеется адресных входов у микросхем ППЗУ, то можно каскадировать две или более микросхемы, объединяя их, как показано на рис. 2.13. Здесь каждая из микросхем обрабатывает свои разряды адреса магистрали. Отметим, что вторая схема работает только на микросхемах, имеющих выходы с открытым коллектором и обеспечивает только высокие активные уровни выходных сигналов ADR. Полная задержка первой схемы складывается из времени выборки адреса и времени выбора и составляет около 80 ... 100 нс. У второй схемы полная задержка определяется только временем выборки адреса и не превышает 50 ...80 нс.

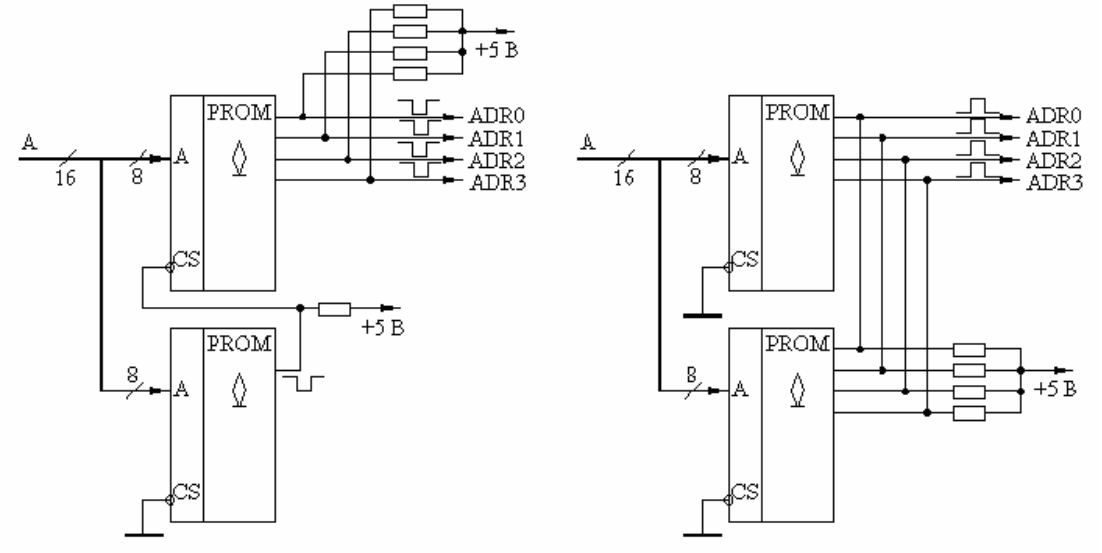


Рис. 2.13. Объединение микросхем ППЗУ в селекторе адреса

2.1.3. Выработка внутренних стробирующих сигналов

Следующая важная функция интерфейсной части УС — выработка внутренних стробирующих сигналов синхронно с магистральными командными сигналами (-IOR, -IOW, -MEMR, -MEMW) в случае обращения по адресам нашего УС. Условно узел, выполняющий эту функцию, может быть представлен в следующем виде (рис. 2.14). На его вход подаются сигналы ADR0 ... ADRN с выхода селектора адреса, SBHE (в случае необходимости разделения 8 и 16-разрядных циклов), а также буферированные магистральные стробы записи и чтения (R и W). Выходы — это сигналы STR0 ... STRn, соответствующие обращениям с записью или чтением по всем адресам или группам адресов УС. Рассмотрим несколько методов построения этого узла.

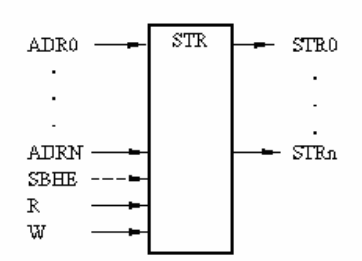


Рис. 2.14. Структура блока выработки внутренних стробов

Самый простейший подход — использование логических элементов — удобен в случае малого числа внутренних стробов STR. На рис. 2.15 показана схема для двух адресов УС, доступных по чтению и записи. Достоинства такого подхода — малое число элементов и высокое быстродействие, а недостаток состоит в том, что приходится разрабатывать новую схему для каждого УС.

В случае необходимости выработки большого числа внутренних стробирующих сигналов удобно использовать микросхемы дешифраторов. Пример такого решения представлен на рис. 2.16. Здесь два младших разряда адреса подаются не на селектор адреса, а непосредственно на дешифратор, верхняя половина которого управляется сигналом с селектора адреса и сигналом -IOR, а нижняя — сигналом с селектора адреса и -IOW. Таким образом, выходы STR0 ... STR3 соответствуют циклам чтения из четырех последовательных адресов, а STR4 ... STR7 — записи в эти адреса. Отметим, что не обязательно надо использовать все выходы дешифратора. Достоинства этого подхода — однотипность схемы рассматриваемого узла для всех УС и малые аппаратурные затраты при необходимости получения большого количества внутренних стробов обмена.

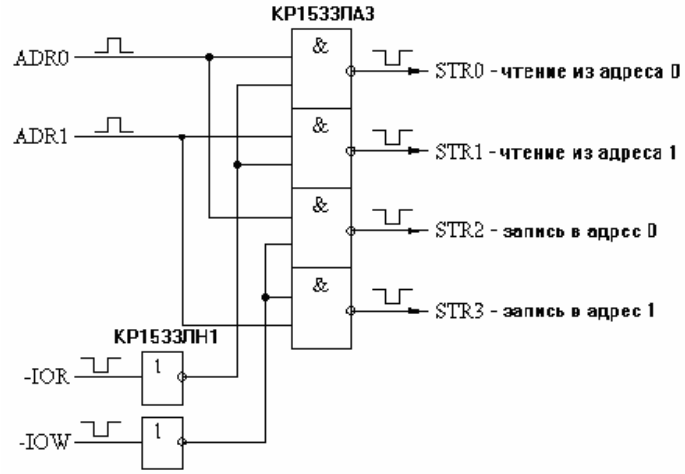


Рис. 2.15. Выработка внутренних стробов с помощью логических элементов

В некоторых случаях удобно не разделять интерфейсную часть УС на селектор адреса и формирователь внутренних стробов. Пусть, например, наше УС должно работать только в циклах записи по его адресам (или только в циклах чтения). При этом оба рассмотренных узла могут быть выполнены на одной микросхеме ППЗУ (рис. 2.17). Здесь к моменту прихода магистрального строба обмена ППЗУ уже успеет сформировать выходные сигналы (закончится время выборки адреса). Поэтому внутренние стробы обмена будут задержаны относительно магистральных стробов только на время выбора ППЗУ. Такой недостаток всех микросхем ППЗУ, как неопределенность выходных

сигналов в течение некоторого времени после любого изменения адреса, здесь не сказывается на работе схемы. Однако не следует надеяться, что схема будет работать также нормально при подаче одного или обоих магистральных стробов обмена (-IOR и -IOW) на адресные входы ППЗУ

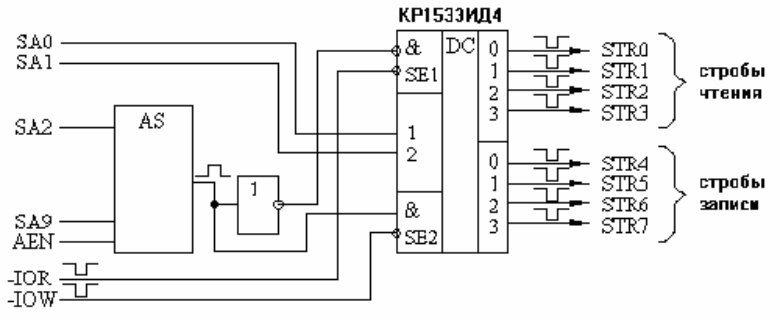


Рис. 2.16. Использование микросхемы дешифратора для выработки внутренних стробов

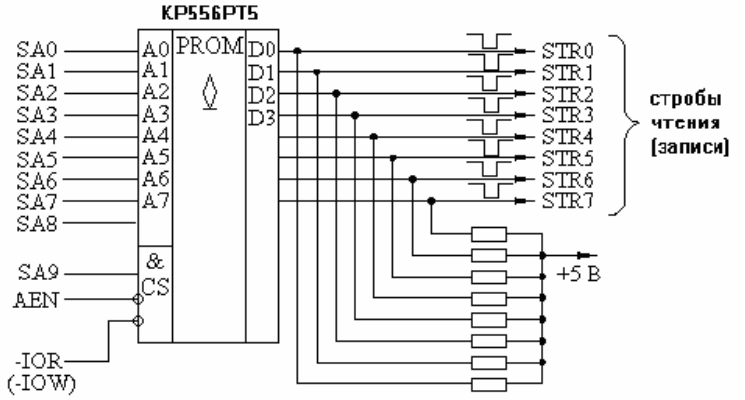


Рис. 2.17. Объединение селектора адреса и формирователя внутренних стробов

.

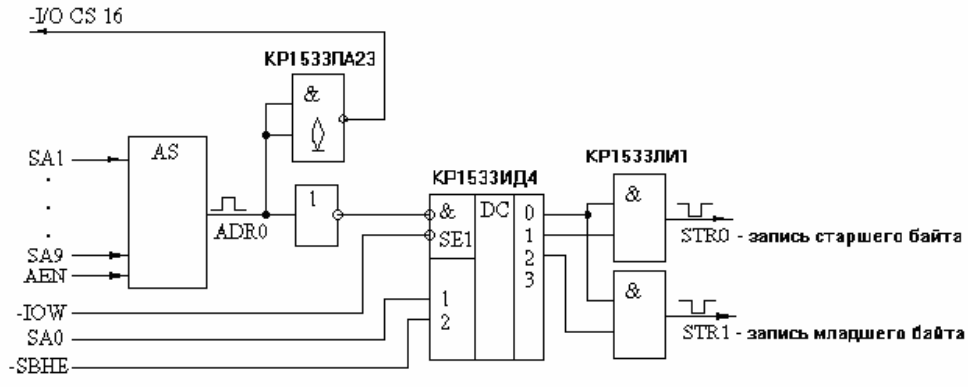


Рис. 2.18. Реализация 8 и 16-разрядной записи в 16-разрядное УС

Особо следует остановиться на организации 16-разрядного обмена и разделении пересылок старшего и младшего байтов. Здесь участвуют два сигнала магистрали, которые не используются при 8-битном обмене: -SBHE и -I/O CS 16 (или -MEM CS 16). При этом сигнал -SBHE должен обрабатываться УС только в случае необходимости как 16, так и 8-

разрядного обмена (вспомним, что он определяет тип цикла обмена совместно с сигналом SA0 в соответствии с таблицей 1.3). На рис 2.18 в качестве примера приведена схема формирователя внутренних стробов для 16-разрядного УС, работающего только в цикле записи 16-разрядного слова, старшего байта или младшего байта. Выходной строб формирователя STR0 соответствует записи старшего байта или слова, а выходной строб STR1 — записи младшего байта или слова. Сигнал -I/O CS 16 вырабатывается при любом обращении к нашему УС, детектируемом селектором адреса. Отметим, что этот сигнал может формироваться и элементом с тремя состояниями, но в этом случае надо обеспечить активный нулевой уровень при селектировании адреса и высокоимпедансное состояние в противном случае (рис. 2.19). Это предотвратит конфликт на линии -I/O CS 16 сигналов от разных плат расширения.

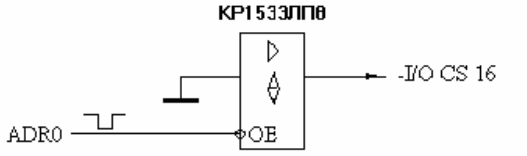


Рис. 2.19. Использование элемента с тремя состояниями для формирования сигнала - I/O CS 16

2.1.4. Асинхронный обмен по ISA

Основным типом обмена по ISA является синхронный обмен, то есть обмен в темпе задатчика без учета быстродействия исполнителя. Однако возможен и асинхронный обмен, при котором "медленный" исполнитель приостанавливает работу задатчика на время выполнения им требуемой команды. В этом случае надо использовать сигнал I/O CH RDY, снятие которого (установка в состояние логического нуля) говорит о неготовности исполнителя к окончанию цикла обмена. Как уже отмечалось, приостановка производится на целое число периодов SYSCLK и не может быть дольше системного времени ожидания 15,6 мкс (для некоторых компьютеров — 2,5 мкс).

Рассмотрим некоторые аппаратурные решения для асинхронного обмена. Прежде всего здесь можно выделить две ситуации: когда существует внутренний сигнал УС, говорящий об окончании выполнения функции записи или чтения, и когда такого сигнала нет. В качестве этого сигнала (обозначим его DK) может выступать, например, сигнал окончания преобразования (готовности данных) АЦП, входящего в состав УС. На рис. 2.20 приведена структура УС с сигналом DK. DK может быть потенциальным (то есть сниматься после окончания стробов обмена) или импульсным (то есть окончанию выполнения функции соответствует фронт сигнала DK). Временные диаграммы и схемы для этих двух случаев показаны на рис. 2.21 и 2.22 (для упрощения считаем, что строб обмена — единственный).

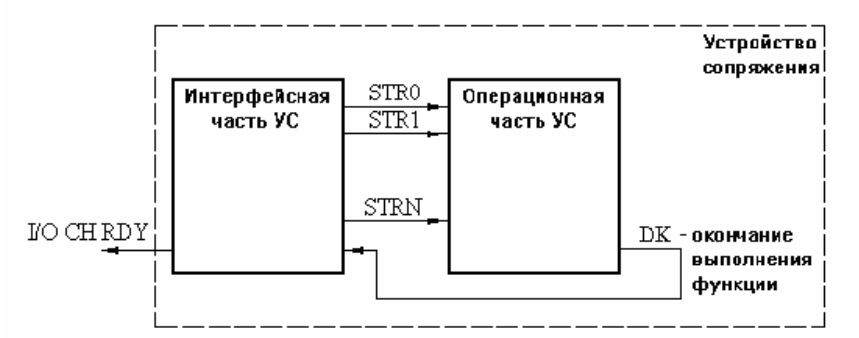


Рис. 2.20. Структура УС, использующая асинхронный обмен

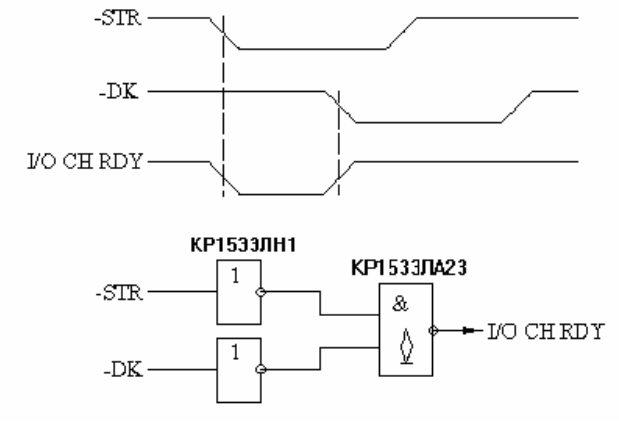


Рис. 2.21. Реализация асинхроннного обмена при потенциальном DK (уровень логического нуля)

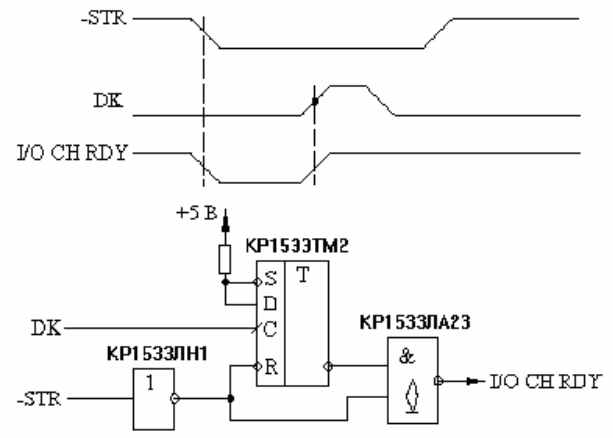


Рис. 2.22. Реализация асинхронного обмена при импульсном DK (положительный фронт)

Если сигнал DK отсутствует в явном виде, но известно время выполнения функции или его верхний предел, то необходимо сформировать задержку в самой интерфейсной части. В схеме на рис. 2.23 слева эта задержка определяется временем выдержки одновибратора. Надо отметить, что при проектировании УС одним из показателей мастерства разработчика является количество использованных им одновибраторов или RC-цепочек (естественно, эти величины обратно пропорциональны друг другу). Это связано с тем, что любые аналоговые цепи подвержены действию помех и требуют настройки.

Поэтому, если есть возможность, то надо формировать задержки, временные сдвиги, интервалы с помощью магистральных тактовых сигналов SYSCLK и OSC или внутренних тактов УС. На рис. 2.23 справа приведена схема с использованием линии задержки на сдвиговом регистре, задержка которой определяется номером замкнутого переключателя и задается с точностью до периода сигнала SYSCLK. Но, в принципе, в данном случае требования к точности времени задержки невысоки, и использование одновибратора и даже простой RC-цепочки вполне допустимо.

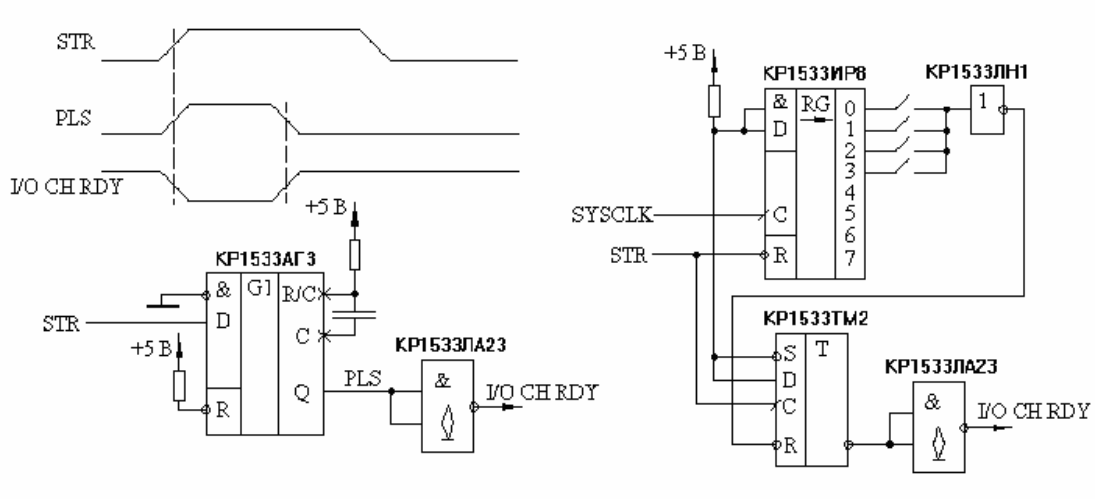


Рис. 2.23. Формирование задержки с помощью одновибратора и линии задержки

И в завершение рассматриваемой темы отметим, что асинхронный режим обмена по ISA можно реализовать и на более высоком уровне: путем опроса задатчиком флага готовности исполнителя и путем использования прерываний. Эти решения удобны в случае очень медленных УС, то есть тех, время исполнения функции которыми превышает предельное системное время задержки.

А теперь попробуем изобразить обобщенную структурную схему интерфейсной части УС, включающей в себя все рассмотренные узлы (рис. 2.24). Здесь использованы входные буфера (не обязательны), двунаправленный буфер данных (в общем случае должен быть разделен на два для каждого байта), выходной буфер, селектор адреса, формироватепль внутренних стробов и формирователь сигнала асинхронного обмена I/O CH RDY (DK).

Какими могут быть предельные значения времен задержек всех узлов интерфейсной части? Здесь надо рассмотреть две ситуации. Если наше УС работает только в режиме записи в него информации, то желательно, чтобы задержка сигнала STR относительно сигнала -IOW и задержки сигналов данных были примерно одинаковыми. Ни в коем случае задержка сигнала STR не должна превышать задержку данных более, чем на 30 нс, иначе УС примет неверные данные (см. рис. 1.3.). Разность задержки буферирования и селектирования адреса и задержки буферирования сигнала -IOW не должна превышать 91 нс, иначе УС не будет реагировать на свой адрес. Если наше УС работает только в режиме чтения из него информации, то сумма задержки сигнала STR относительно сигнала -IOR и задержки буфера данных не должна превышать 110 нс, иначе процессор примет неправильные данные от УС. Требования к буферу адреса и селектору адреса те же. Если же наше УС должно работать как в режиме чтения, так и в режиме записи, то оно должно удовлетворять всем перечисленным требованиям.

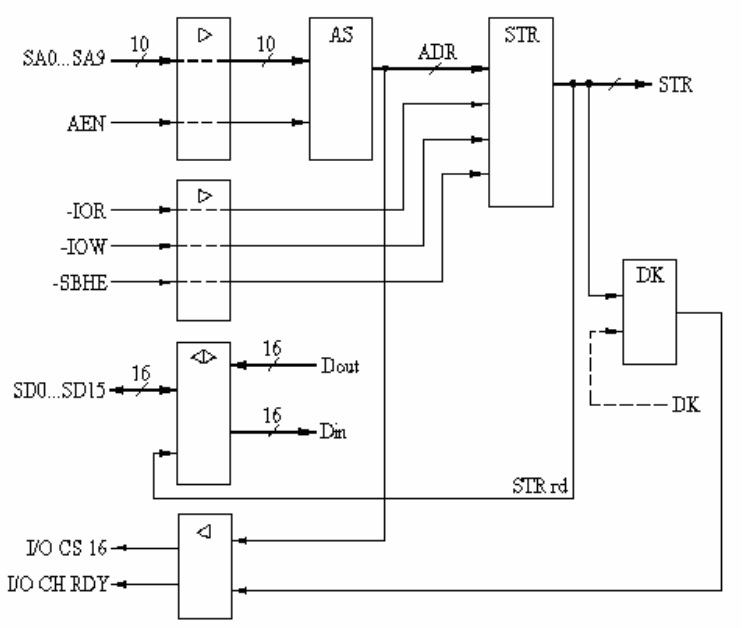


Рис. 2.24. Обобщенная структурная схема интерфейсной части УС

2.1.5. Особенности использования аппаратных прерываний

Аппаратные прерывания, как известно, применяются для информирования исполнителем задатчика о готовности к обмену в случае, когда нежелательно организовывать программный опрос флага готовности. Часто можно услышать мнение, что использование прерываний увеличивает быстродействие компьютера. Здесь надо четко понимать, о чем идет речь. Да, в случае применения прерываний процессор освобождается от опроса УС и может заниматься вместо этого другими задачами. Если необходимость в обмене возникает достаточно редко, то выигрыш в быстродействии всей системы в целом может быть довольно значительным. Однако использование прерываний ни в коем случае не увеличивает скорости обмена с УС, наоборот уменьшает ее. Это связано с тем, что реакция на прерывание гораздо медленнее, чем на выставление флага готовности, ведь, чтобы обслужить прерывание, процессор должен завершить текущий цикл, сохранить в стеке текущие значения своих регистров и только потом перейти на программу обработки прерывания. Поэтому в том случае, когда надо обеспечить максимальную скорость реакции на внешнее событие, гораздо лучше использовать опрос флага готовности. Точно так же надо поступать и при достаточно частом возникновении необходимости в обмене, ведь помимо временных затрат на переход к обработке прерывания требуется также время на инициализацию контроллера прерываний, входящего в состав компьютера.

Так что перед тем, как заложить в схему проектируемого УС использование прерываний, стоит хорошенько подумать. Часто имеет смысл продублировать прерывание флагом готовности, что позволит при написании программы, обслуживающей УС, более гибко учитывать особенности обмена с данным УС.

Как уже отмечалось, все прерывания в IBM PC AT — радиальные, то есть для перевода процессора в режим обработки прерывания достаточно послать запрос, в качестве которого выступает положительный фронт сигнала на одной из линий IRQ. Однако из этого совсем не следует, что на эту линию можно подать сколь угодно короткий импульс положительной полярности, так как никакой гарантии, что этот импульс дойдет

до контроллера прерываний, будет им обработан и вызовет нужную реакцию, при этом не существует.

Самым оптимальным решением, на наш взгляд, здесь будет перевод сигнала IRQ в логическую единицу при возникновении необходимости обмена (то есть аппаратно), а сброс его в исходное состояние логического нуля по команде процессора, которую он исполняет в ходе программы обработки прерывания (то есть программно). При этом мы можем быть полностью уверены, что наш запрос действительно принят и обработан.

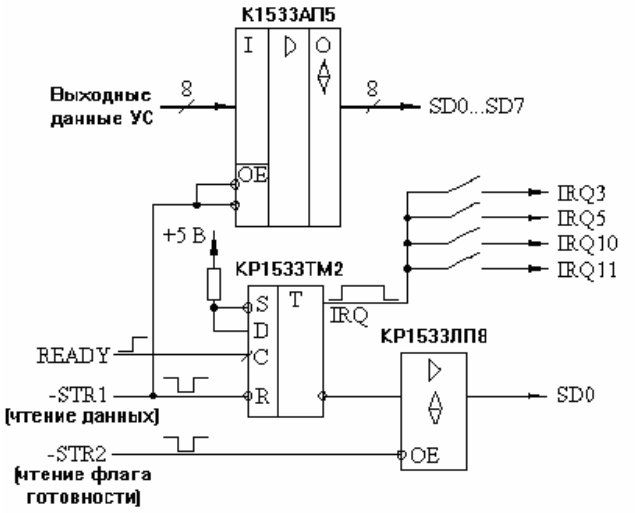


Рис. 2.25. Реализация прерывания

Пусть, например, наше УС принимает данные от внешнего устройства и после их приема нам надо сформировать прерывание, обработка которого сводится только к чтению данных из УС. Схема для этого случая показана на рис. 2.25. Здесь READY — сигнал поступления данных и готовности к выдаче их процессору. По этому сигналу триггер устанавливается в единицу, и его выход используется как сигнал запроса прерывания. Номер используемой линии IRQ выбирается одним из четырех переключателей (такой выбор в том или ином виде необходим, так как свободных линий IRQ в стандартной конфигурации не так уж много). В исходное состояние триггер сбрасывается стробом чтения данных, вырабатываемым при выполнении программы обработки прерывания. Инверсный выход триггера используется как флаг готовности, который программно опрашивается процессором с помощью сигнала -STR2.

Иногда имеет смысл предусмотреть в схеме УС возможность запрещения прерывания от него, хотя это можно сделать и путем маскирования данного прерывания в контроллере прерываний (программным путем).

В заключение еще раз отметим, что тип выходного каскада линии IRQ — это обычный ТТЛ, поэтому необходимо тщательно контролировать, чтобы на каждую из линий IRQ поступал сигнал только от одной платы расширения, иначе может выйти из строя выходной каскад одной из конфликтующих плат.

2.1.6. Применение прямого доступа

С использованием прямого доступа также связано несколько недоразумений. Многие считают, что прямой доступ обеспечивает максимальный темп выдачи или приема информации. Однако это не совсем так. Прямой доступ осуществляется по системной магистрали ISA, и, следовательно, его скорость ограничена, в первую очередь, принятым

для ISA протоколом обмена, а также невысоким быстродействием используемого в компьютере динамического ОЗУ. Даже в режиме блочной передачи пересылка одного байта (слова) требует нескольких тактов SYSCLK и занимает около 1 мкс. Что же касается режима одиночной передачи, другого возможного режима ПДП, то он вообще не дает никакого выигрыша в скорости.

Наибольшую скорость выдачи или приема данных обеспечивают не УС с прямым доступом, а УС с так называемой разделяемой памятью, в которых быстрая буферная память (конечно же, статическая), расположенная на плате УС, доступна как со стороны внешнего устройства, так и со стороны магистрали ISA. При этом процессор рассматривает эту буферную память как часть системного ОЗУ. В этом случае прием информации в ОЗУ компьютера или выдача ее оттуда может осуществляться со скоростью до 50 Мбайт/с и даже выше (при принятии специальных мер), что определяется исключительно быдстродействием буферного ОЗУ. В качестве этого буферного ОЗУ надо использовать не медленную динамическую память со временем выборки около 70 ... 100 нс, а гораздо более быструю статическую со временем выборки 20 нс и менее. Ведь здесь основным фактором становится не стоимость микросхем памяти (как для компьютера), а их быстродействие (и, кстати, отсутствие регенерации).

Следует однако учитывать, что указанная высокая скорость достигается только в пределах передачи блока информации, размер которого определядется размером буферного ОЗУ. При необходимости же передачи больших объемов информации интегральная (средняя) скорость передачи будет определяться уже быстродействием компьютера в целом и будет гораздо ниже. Но об этом несколько позже.

Еще к вопросу о выигрыше в скорости при использовании прямого доступа. Особенностью процессоров IBM PC AT является наличие так называемых цепочечных или строковых команд пересылки данных, в частности, цепочечных команд ввода и вывода. Их использование позволяет обеспечить скорость не меньшую, чем применение прямого доступа. Однако существенное преимущество цепочечных команд — отсутствие необходимости инициализации контроллера прямого доступа, то есть в данном случае не требуются дополнительные временные затраты. Использование же их совместно с прерываниями сводит на нет и такое преимущество прямого доступа, как начало обмена по инициативе исполнителя (нашего УС). Поэтому многие разработчики считают нецелесообразным ориентацию своих УС на прямой доступ.

Стоит также отметить, еще одно соображение против применения ПДП. Даже стандартные платы расширения, выпускаемые солидными фирмами и использующие прямой доступ к памяти, ведут себя правильно не во всех компьютерах. Изготовители часто указывают, в компьютерах каких фирм их платы гарантированно работают. Связано это прежде всего с недостаточной стандартизацией магистрали ISA вообще и режима ПДП на ней в частности.

Тем не менее в ряде случаев применение прямого доступа вполне удовлетворительно решает функции обмена УС с системной памятью, поэтому мы все-таки рассмотрим пример реализации запроса прямого доступа. В отличие от прерываний, где требуется только один сигнал — IRQ, здесь в диалоге "запрос прямого доступа — предоставление прямого доступа" участвуют два сигнала: DRQ и DACK. Кроме того, в обмене участвует сигнал AEN. Пример схемы запроса прямого доступа показан на рис. 2.26. Сигнал READY — это готовность УС к обмену данными в режиме прямого доступа. Он перебрасывает триггер, выход которого — сигнал DRQ. В исходное состояние триггер сбрасывается сигналом DACK. Если прямой доступ предоставлен нашему УС, то сигнал AEN должен интерпретироваться как наш адрес, поэтому мы объединяем его с сигналом с выхода селектора адреса (считаем, что возможен также режим программного обмена). Точно так же, как в случае прерываний, мы должны следить, чтобы на каждую линию DRQ поступал только один сигнал, а также, чтобы совпадали номера используемых нашим УС

сигналов DRQ и DACK.

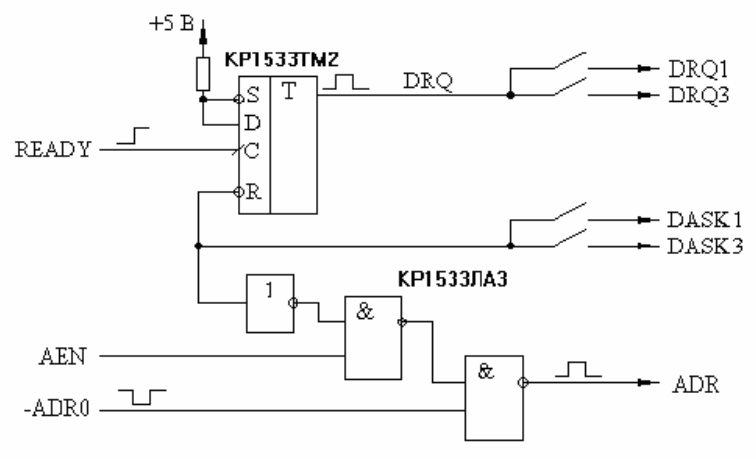


Рис. 2.26. Реализация прямого доступа

2.1.7. Буферные ОЗУ устройств сопряжения

Мы рассмотрели основные узлы интерфейсной части УС, которые входят практически во все УС. Операционные (основные) части УС гораздо более разнообразны, поэтому выделить какие-то общие принципы их проектирования очень трудно, но кое-какие универсальные подходы все-таки предложить можно.

Значительная часть УС содержит в своем составе буферные ОЗУ, используемчые для промежуточного хранения данных при пересылке их из компьютера во внешнее устройство или из внешнего устройства в компьютер. В каких случаях нужны буферные ОЗУ?

1. При медленных внешних устройствах (то есть компьютер может выдавать или принимать данные в нужном темпе) буферные ОЗУ нужны в двух ситуациях. Во-первых, если необходимо поддерживать постоянный темп выдачи (приема) данных. Когда требования к точности этого темпа невелики, можно воспользоваться программными задержками, опросом флага готовности или опросом внутреннего системного таймера компьютера. Но во всех этих случаях на темп выдачи (приема) будут влиять непредсказуемые внешние факторы: прерывания, прямой доступ, а также регенерация. К тому же дискрет времени задержки в этом случае не может быть меньше выполнения нескольких команд процессором. Если же требования к точности частоты выдачи (приема) данных высоки, например, погрешность не должна превышать долей процента, то можно использовать буферное ОЗУ и его опрос по кварцевому генератору УС (простейший пример — синтез низкочастотного аналогового сигнала с точной частотой).

Во-вторых, буферное ОЗУ очень полезно при необходимости выдачи (приема) больших объемов данных для того, чтобы освободить процессор для решения других задач. То есть процессор в данном случае только запускает процесс выдачи (приема), а УС само осуществляет обмен с внешним устройством, информируя процессор только о его окончании (выставлением флага готовности или прерыванием).

2. Если внешние устройства — быстрые, то есть компьютер не может обеспечить требуемого темпа выдачи (приема) данных. Минимальный период выдачи (приема) данных компьютером составляет около 1 мкс. Если нужен больший темп, то без буферного ОЗУ никак не обойтись. А если еще необходима и высокая точность поддержания этого темпа, то применение буферного ОЗУ требуется и при гораздо меньшем темпе.

Таким образом, буферные ОЗУ в составе УС нужны во многих случаях. Теперь посмотрим особенности схемотехнических решений таких УС.

По способу обмена с внешним устройством УС, имеющие буферные ОЗУ, могут быть разделены на две большие группы:

1. УС с непрерывным режимом обмена с внешним устройством, то есть буферное ОЗУ непрерывно выдает на внешнее устройство или принимает от него данные, а процессор в определенные моменты соответственно записывает или считывает необходимые ячейки этого ОЗУ. Примером может слукжить контроллер телевизионного монитора, в котором необходимо проводить непрерывную регенерацию изображения. Другой пример — контроллер телевизионной камеры.

2. УС с периодическим режимом обмена с внешним устройством, то есть буферное ОЗУ может находиться в одном из двух режимов: в режиме обмена с компьютером (запись или чтение содержимого ОЗУ) или в режиме обмена с внешним устройством (прием или выдача). При этом, естественно, обмен с компьютером осуществляется в темпе компьютера, а обмен с внешним устройством — в темпе внешнего устройства. Устройств этой второй группы гораздо больше: контроллер дисковода, синтезатор аналогового сигнала, цифровой осциллограф, контроллер локальной сети и т.д. Например, в случае контроллера сети при передаче пакета в сеть надо сначала записать пакет в буферное ОЗУ (в темпе компьютера), а затем выдать его в сеть в темпе сети, выступающей в данном случае качестве внешнего устройства. В принципе, могут быть и промежуточные ситуации, когда во время обмена с компьютером проводится и обмен с внешним устройством.

По методу доступа к буферному ОЗУ со стороны компьютера УС могут быть разделены на следующие группы:

1. УС с параллельным доступом к буферному ОЗУ. В этом случае каждой ячейке буферного ОЗУ соответствует свой адрес в адресном пространстве компьютера. Это как раз то, что называется разделяемой памятью. Любой задатчик магистрали (процессор, контроллер ПДП и т.д.) может общаться с буферным ОЗУ как с системным, используя для этого все средства, все методы адресации, команды обработки строк. Естественно, это наиболее быстрый метод общения с буферным ОЗУ (а также и с внешним устройством), так как в данном случае не требуется времени для перекачки данных из системной памяти в буферное ОЗУ УС или наоборот. Схематически этот метод доступа можно представить следующим образом (рис. 2.27). В адресном пространстве памяти ISA выделяется окно, в которое проецируются адреса буферного ОЗУ. Адресное пространство устройств ввода/вывода в данном случае ипсользовать нецелесообразно, так как в нем нет достаточно больших непрерывных зон свободных адресов. Кроме того возможности процессора по работе с памятью гораздо богаче, чем по общению с устройствами ввода/вывода. Схемотехнически этот метод доступа реализуется довольно сложно, но преимущества его очень велики.

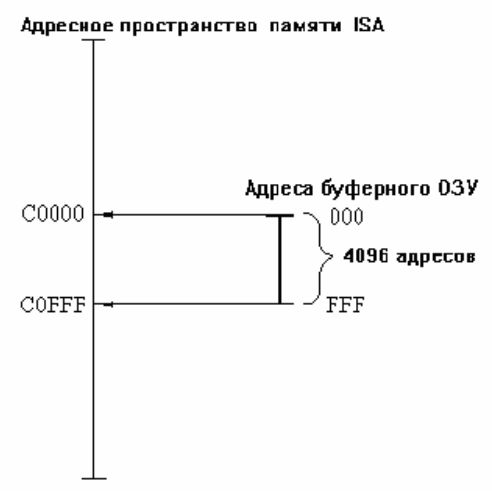


Рис. 2.27. Параллельный доступ к буферному ОЗУ

2. УС с последовательным доступом к буферному ОЗУ. В данном случае все ячейки буферного ОЗУ по очереди проецируются в один адрес в адресном пространстве компьютера (или реже в несколько адресов). То есть процессор при обращении по одному и тому же адресу общается в разное время с разными ячейками буферного ОЗУ. Недостаток этого подхода — резкое снижение темпа обмена, а самое очевидное преимущество — экономия адресов магистрали. Интересно, что буферное ОЗУ при этом может быть даже больше, чем потенциально возможная системная память компьютера. К тому же в данном случае можно перейти в адресное пространство устройств ввода/вывода, что позволяет существенно упростить селектор адреса.

Обмен с буферным ОЗУ осуществляется при таком подходе в два этапа: сначала определяем, с каким адресом буферного ОЗУ будет производиться обмен и только затем этот обмен производим. Схематически этот метод иллюстрируется рис. 2.28.

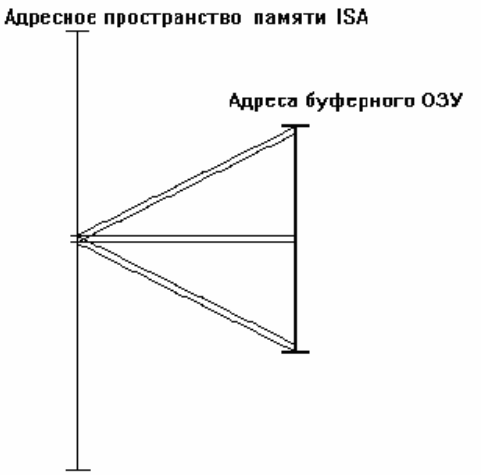


Рис. 2.28. Последовательный доступ к буферному ОЗУ

Метод последовательного доступа может быть реализован несколькими путями, отличающимися скоростью обмена и сложностью аппаратуры, а также удобством доступа. Самый простой путь — использование адресного регистра буферного ОЗУ (рис. 2.29 —

очень условная схема). Адрес буферного ОЗУ задается содержимым регистра, доступного со стороны магистрали в цикле записи (сигнал -STR0). Обмен с ОЗУ производится по стробам -STR1 и -STR2. Здесь УС должно иметь два адреса в адресном пространстве компьютера: по одному из них записывается код адреса ячейки буферного ОЗУ, с которым будет производиться обмен, по другому — читается или записывается эта ячейка. Чтобы прочитать (записать) все буферное ОЗУ требуется в два раза больше обращений к УС, чем имеется адресов ОЗУ. Использование адресного регистра — это самый медленный способ обмена.

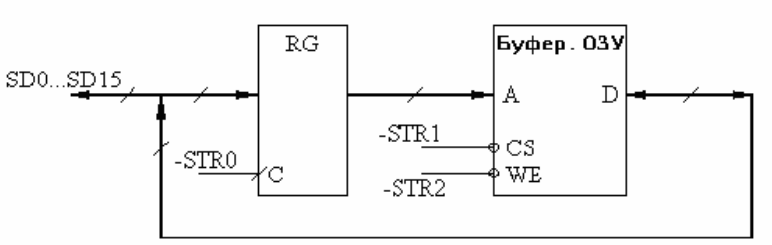


Рис. 2.29. Использование адресного регистра для последовательного доступа

Второй путь реализации метода последовательного доступа — это применение адресного счетчика (рис. 2.30а — схема очень условная). В данном случае нельзя прочитать (записать) произвольную ячейку буферного ОЗУ, а можно только читать (записывать) все его содержимое. Перед началом обмена по сигналу STR1 сбрасываем адресный счетчик. Затем начинаем запись или чтение ОЗУ по сигналам -STR0 и -STR2. При этом каждое обращение к ОЗУ наращивает содержимое счетчика на единицу. Произведя столько обращений, сколько имеется адресов ОЗУ, мы заканчиваем обмен. Достоинством данного подхода является то, что чтение (запись) всех адресов ОЗУ требует в два раза меньше обращений к УС по сравнению с предыдущим случаем. Недостаток этого пути — невозможность обмена с произвольными ячейками ОЗУ.

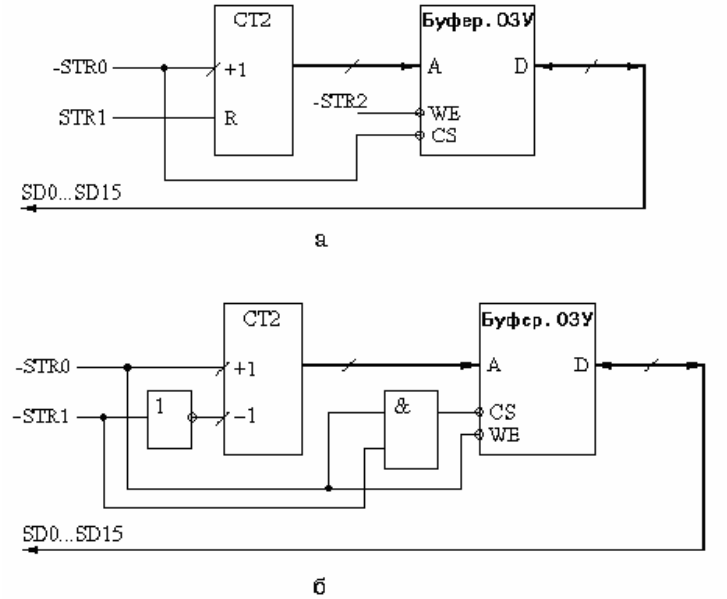


Рис. 2.30. Использование адресного счетчика для последовательного доступа к буферному ОЗУ (а) и организация стекового ОЗУ (б)

Частный случай данного подхода — реализация буферного ОЗУ стекового типа

(рис. 2.30б — схема опять же очень условная). Обмен с ОЗУ производится по сигналам -STR0 (запись) и -STR1 (чтение). При этом после записи состояние счетчика наращивается на единицу, а перед чтением — уменьшается на единицу (то есть реализуется постинкремент и предекремент). Интересно, что здесь не нужен сброс счетчика, так как его начальное состояние совершенно не важно.

И, наконец, третий путь реализации последовательного доступа к буферному ОЗУ — объединение двух рассмотренных подходов, позволяющее соединить их преимущества (рис. 2.31). В данном случае используется счетчик с параллельной записью, в который перед началом обмена записывается по сигналу -STR0 начальный адрес ОЗУ. Затем производится обмен с ОЗУ по сигналам -STR1 и -STR2. Здесь можно перебирать ячейки последовательно, но можно и задавать адрес ячейки произвольно.

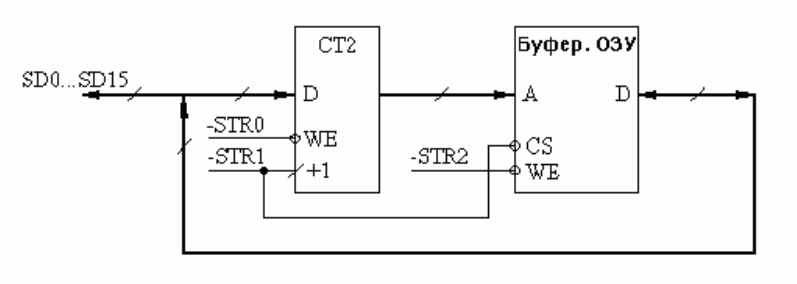


Рис. 2.31. Использование счетчика с параллельной записью для последовательного доступа к буферному ОЗУ

При любой схеме включения буферного ОЗУ возникают две основные проблемы: организация задания адреса ОЗУ во всех режимах работы и организация прохождения потоков данных. Рассмотрим несколько примеров решения этих проблем.

В случае УС с непрерывным режимом обмена с внешним устройством для задания адреса ОЗУ приходится применять две схемы формирования адреса и мультиплексор для выбора одной из них. На рис. 2.32 верхний счетчик формирует адрес, по которому осуществляется обмен с внешним устройством (он работает постоянно). Нижний счетчик (или регистр) задает адрес ячейки ОЗУ, с которой производится обмен со стороны магистрали. При обращении со стороны магистрали по сигналу -STR0 мультиплексор переключается на передачу выходного кода нижнего счетчика. По этому же сигналу переключается режим работы ОЗУ. При этом необходимо принять меры для предотвращения нарушения обмена с внешним устройством в момент обращения со стороны магистрали (например, путем введения разделения этих циклов обмена во времени).

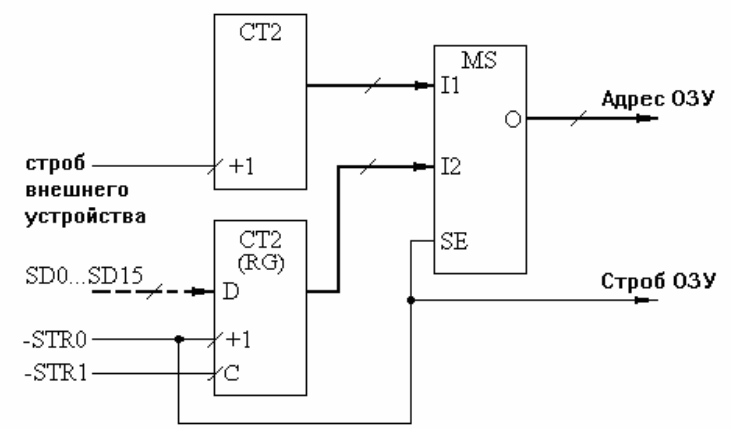


Рис. 2.32. Задание адреса буферного ОЗУ при непрерывном режиме обмена с внешним устройством

В случае УС с периодическим режимом обмена с внешним устройством схема может быть значительно проще (рис. 2.33). Здесь используется один счетчик, а мультиплексор подключает к его входу или строб внешнего устройства (при обмене с внешним устройством) или строб обмена — STR1 (при обмене с магистральною). После окончания обмена с магистралью по сигналу -STR0 схема переходит в режим обмена с внешним устройством (перебрасывается триггер). Возвращение к режиму обмена с магистралью происходит после полной отработки цикла обмена с внешним устройством и формирования счетчиком сигнала переноса.

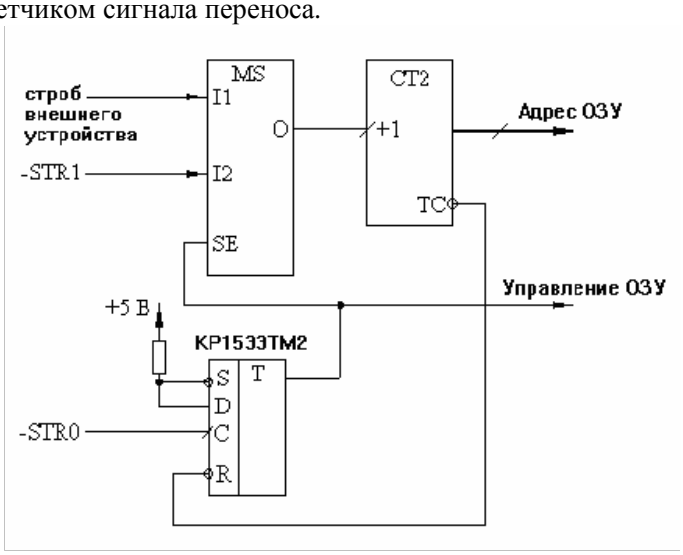


Рис. 2.33. Задание адреса буферного ОЗУ при периодическом режиме обмена с внешним устройством

С точки зрения организации прохождения потоков данных, можно выделить четыре режима работы схемы: запись ОЗУ со стороны магистрали, чтение ОЗУ со стороны магистрали, прием данных в ОЗУ со стороны внешнего устройства и выдача данных из ОЗУ на внешнее устройство. Как правило, в УС применяются микросхемы многоразрядного ОЗУ (с организацией N x 8), которые имеют двунаправленную шину данных, что накладывает ограничения на схему организации прохождения потоков данных. Пример схемы для УС, в котором реализованы все четыре перечисленных режима работы, показан на рис. 2.34. Здесь для обмена 8-разрядного буферного ОЗУ с

магистралью используется двунаправленный приемопередатчик типа КР1533АП6, а для обмена с внешним устройством — однонаправленный передатчик типа КР1533АП5. Конечно же, эта схема, как и все приведенные в этом разделе, далеко не единственно возможная. К тому же все эти схемы достаточно условны.

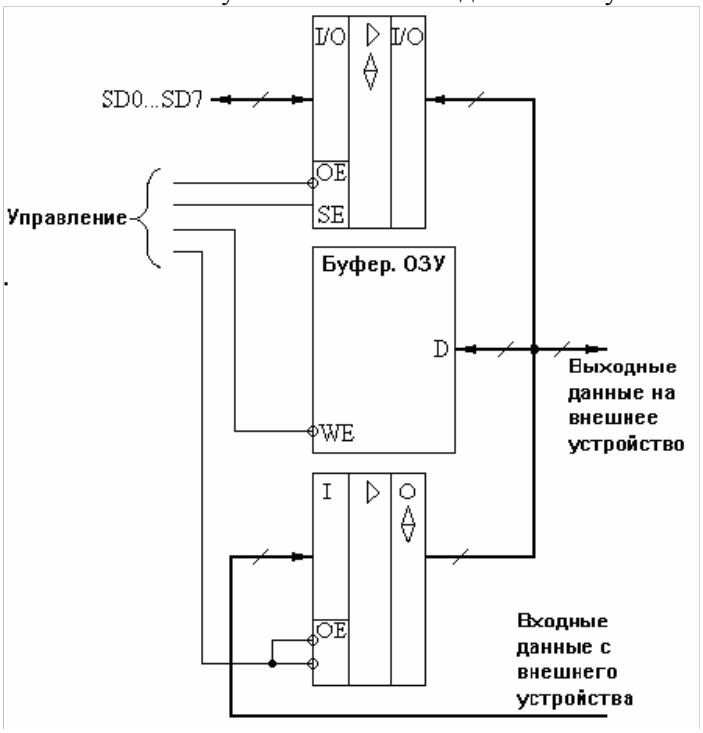


Рис. 2.34. Организация прохождения потоков данных

2.1.8. Микропрограммные автоматы

Еще одна функция, которую часто нужно реализовать при проектировании операционной части УС — это формирование сигналов синхронизации требуемой длительности и в заданной последовательности. Эти сигналы используются для временного согласования работы различных частей схемы УС. В простейшем случае можно решить данную задачу с помощью одновибраторов или элементов задержки, но такой подход хорош только тогда, когда надо сформировать 1-2 сигнала, и требования к точности временных задержек невысоки. Если же это не так, да к тому же алгоритм выработки сигналов довольно сложен, то гораздо эффективнее использовать микропрограммный автомат.

Микропрограммные автоматы обеспечивают в целом ряде случаев удачный компромисс между сложностью алгоритма работы и быстродействием. По сравнению с жесткой логикой они, обычно позволяют при минимуме аппаратурных затрат реализовать достаточно сложные алгоритмы работы и довольно легко их менять, а по сравнению с программной реализацией (например, на базе однокристальной микроЭВМ) они имеют, как правило, заметно большее быстродействие (см. рис. 2.1).

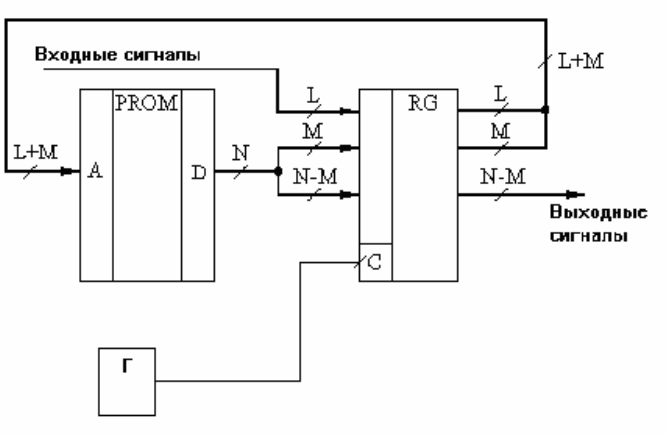


Рис. 2.35. Структура микропрограммного автомата

Наиболее универсальная схема микропрограммного автомата строится на основе ППЗУ или ПЛМ и регистра, охваченных обратной связью. В этом случае алгоритм работы автомата определяется исключительно прошивкой ППЗУ или ПЛМ и может быть при необходимости очень просто изменен заменой микросхемы. Обобщенная структура автомата на ППЗУ представлена на рис. 2.35. Она включает в себя ППЗУ с (L + M) адресными входами и N разрядами данных, (N + L)-разрядный регистр и тактовый генератор Г. Часть выходных разрядов ППЗУ, (N — M), используется для формирования выходных сигналов, а остальные M разрядов участвуют совместно с L входными сигналами в формировании адреса ППЗУ в следующем такте тактового генератора. Отметим, что в данной схеме необходимо применять только микросхемы регистров со стробированием по фронту (а не по уровню, не регистры-защелки), так как в противном случае при переходе регистра в режим пропускания входной информации замкнется обратная связь и возможна даже неконтролируемая автогенерация. Примеры микросхем регистров: КР1533ИР23, КР1533ИР27, КР1533ИР37 и т.д. Максимальная тактовая частота автомата, определяющая его ббыстродействие, должна быть такой, чтобы суммарная задержка регистра и ППЗУ не превышала периода тактовой частоты.

Отметим, что дискрет временных сдвигов между выходными сигналами микропрограммного автомата равен периоду тактовой частоты, а задержка реакции на изменение входного сигнала имеет случайный характер, но не превышает периода тактовой частоты (если не принимать специальных мер по синхронизации тактового генератора и входных сигналов). Эти особенности необходимо иметь в виду при выборе схемы и составлени микропрограммы.

Из возможных режимов работы микропрограммного автомата можно выделить следующие:

— последовательный перебор адресов ППЗУ (аналог счетчика);

— останов в заданном адресе ППЗУ с бесконечным ожиданием;

— переход на другой адрес при изменении входного сигнала (или нескольких входных сигналов);

— циклическое прохождение, перебор одних и тех же адресов ППЗУ;

— отключение реакции на входные сигналы (то есть автомат никак не реагирует на изменение входных сигналов в течение какого-то времени).

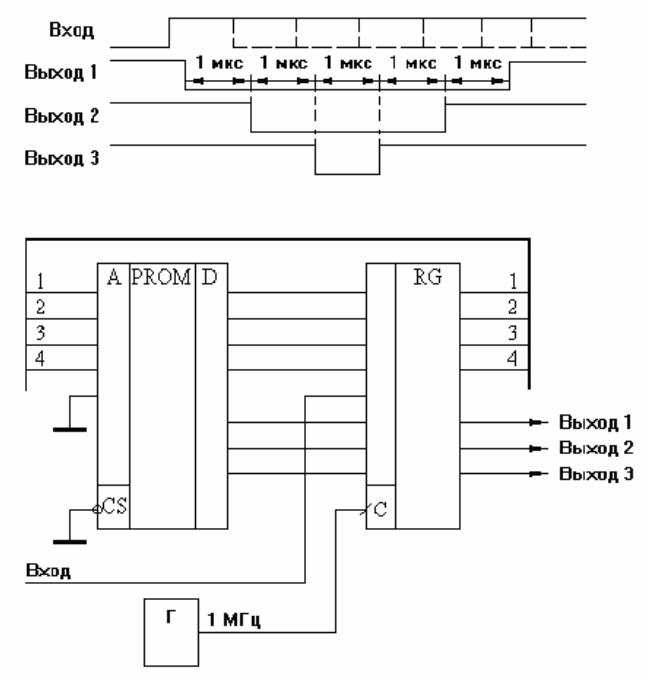


Рис. 2.36. Пример микропрограммного автомата

Рассмотрим небольшой пример проектирования микропрограммного генератора, работающего в соответствии с временной диаграммой рис. 2.36. В ответ на положительный фронт входного сигнала надо выработать три выходных сигнала, "вложенных" один с другой. Кстати, формирование таких "вложенных" циклов довольно часто требуется при проектировании различных цифровых устройств, в том числе, УС. Во время формирования выходной последовательности автомат не должен реагировать на входной сигнал, а после ее окончания должен ожидать следующего положительного фронта на входе. Для упрощения задачи примем все временные сдвиги равными 1 мкс. Поэтому частота тактового генератора должна быть 1 Мгц.

Адрес ППЗУ				Данные ППЗУ						Комментарий
Вх.	Адрес			Выходы 1	2	3	Следующий адрес			
0	0	0	0	1	1	1	0	0	0	1. Ожидание входного сигнала
1	0	0	0	0	1	1	0	0	1	2. Отработка выходной последовательности при условии, что входной сигнал постоянно находится в состоянии логической единицы
1	0	0	1	0	0	1	0	1	0	
1	0	1	0	0	0	0	0	1	1	
1	0	1	1	0	0	1	1	0	0	
1	1	0	0	0	1	1	1	0	1	
1	1	0	1	1	1	1	1	0	1	3. Ожидание снятия входного сигнала
0	0	0	1	0	0	0	0	1	0	4. Отработка выходной последовательности при снятии входного сигнала до ее окончания
0	0	1	0	0	0	0	0	1	1	
0	0	1	1	0	0	1	1	0	0	
0	1	0	0	0	1	0	0	0	0	5. Переход на ожидание входного сигнала
0	1	0	1	1	1	0	0	0	0	

Табл. 2.2. Микропрограмма для автомата рис. 2.36.

Как видно из временной диаграммы, выходная последовательность состоит из шести тактов (включая все единичные уровни). Поэтому требуется 3 адресных входа ППЗУ (8 возможных состояний). Но помимо этого есть один входной сигнал, значит значит количество адресных разрядов ППЗУ будет 4, а количество разрядов регистра должно быть равно 7 (еще добавится три выходных сигнала). Схема примет вид показанный на рис. 2.36. Здесь может быть использовано ППЗУ К155РЕ3 с организацией 32 х 8 и регистр КР1533ИР27. Теперь составим прошивку ППЗУ для нашего автомата (табл. 2.2). Нетрудно заметить, что здесь реализованы следующие режимы: последовательный перебор адресов (при отработке выходной последовательности), останов с ожиданием, отключение реакции на входной сигнал (путем дублирования последовательного перебора в зоне адресов, соответствующей изменению входного сигнала).

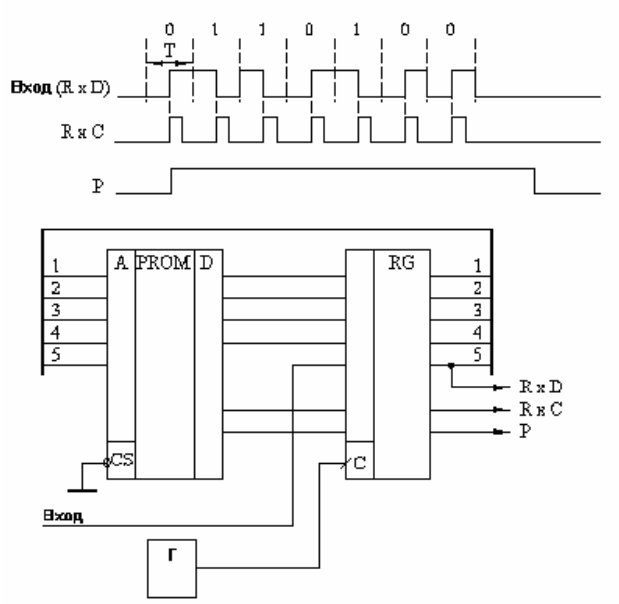


Рис. 2.37. Дешифратор манчестерского кода на микропрограммном автомате

В заключение этой темы — еще один пример микропрограммного автомата, реализующего дешифрацию одного из наиболее распространенных кодов — манчестерского кода, широко применяемого при последовательной передаче данных, в частности, в локальных сетях. Напомним, что в этом коде единица представляется переходом в центре битового интервала из высокого уровня в низкий, а нуль — переходом из низкого уровня в высокий. Дешифрация сводится к выделению из входного сигнала импульсов синхронизации RxC (рис. 2.37). Предлагаемый автомат помимо собственно дешифрации позволяет также сформировать огибающую передаваемого пакета данных, то есть детектировать наличие передачи (сигнал P). Отметим, что в данном случае считается, что пакет всегда начинается с нулевого информационного бита.

Для формирования сигнала RxC после выявления очередного перехода (фронта) в середине битового интервала автомат не реагирует на входной сигнал в течение трех четвертей длительности одного бита (T). Для формирования сигнала P после обработки последнего бита проверяется наличие фронта входного сигнала в течение 1,5 T, и в случае его отсутствия снимается сигнал P. В качестве выходного сигнала данных RxD используется пропущенный через регистр входной сигнал. Тактовая частота

микропрограммного автомата в восемь раз превышает частоту поступления входных данных. Прошивка ППЗУ приведена в табл. 2.3.

2.1.9. Универсальный контроллер параллельного обмена

Предыдущие разделы были посвящены методам построения наиболее типичных узлов УС. Теперь мы переходим к рассмотрению нескольких конкретных примеров УС распространенных типов. Из-за ограниченного объема книги этих примеров будет не слишком много, и описание УС будет довольно кратким.

Первой рассмотренной нами схемой будет простейшее, но, тем не менее, очень полезное УС — универсальный многоразрядный контроллер параллельного обмена информацией. С его помощью компьютер может общаться с любыми цифровыми устройствами в статическом режиме (то есть в темпе, не превышающем быстродействие компьютера). Также его можно использовать для статической отладки УС (о методе статической отладки и его реализации на базе персонального компьютера будет рассказано в разделе 2.3).

Адрес ППЗУ					Данные ППЗУ						Комментарий
Вход	Адрес				C	P	След. адрес				
0	0	0	0	0	0	1	0	0	0	1	Задержка и ожидание положительного фронта входного сигнала
0	0	0	0	1	0	1	0	0	1	0	
0	0	0	1	0	0	1	0	0	1	1	
0	0	0	1	1	0	1	0	1	0	0	
0	0	1	0	0	0	0	0	1	0	0	Снятие P и ожидание входного сигнала
1	0	0	0	0	1	1	0	1	0	1	Выставление RxC и переход на обработку положительного фронта входного сигнала
1	0	0	0	1	1	1	0	1	0	1	
1	0	0	1	0	1	1	0	1	0	1	
1	0	0	1	1	1	1	0	1	0	1	
1	0	1	0	0	1	1	0	1	0	1	
0	0	1	0	1	1	1	0	1	1	0	Снятие RxC и задержка с отключением входа
0	0	1	1	0	0	1	0	1	1	1	
0	0	1	1	1	0	1	1	0	0	0	
0	1	0	0	0	0	1	1	0	0	1	
0	1	0	0	1	0	1	0	0	0	0	Переход на ожидание положительного фронта
1	0	1	0	1	1	1	0	1	1	0	Снятие RxC и задержка с отключением входа
1	0	1	1	0	0	1	0	1	1	1	
1	0	1	1	1	0	1	1	0	0	0	
1	1	0	0	0	0	1	1	0	0	1	
1	1	0	0	1	0	1	1	0	1	0	Переход на ожидание отрицательного фронта
1	1	0	1	0	0	1	1	0	1	1	Задержка и ожидание отрицательного фронта входного сигнала
1	1	0	1	1	0	1	1	1	0	0	
1	1	1	0	0	0	1	1	1	0	1	
1	1	1	0	1	0	1	1	1	1	0	
1	1	1	1	0	0	0	1	1	1	0	Снятие P и ожидание входного сигнала
0	1	0	1	0	1	1	0	1	0	1	Выставление RxC и переход на обработку отрицательного фронта входного сигнала
0	1	0	1	1	1	1	0	1	0	1	
0	1	1	0	0	1	1	0	1	0	1	
0	1	1	0	1	1	1	0	1	0	1	
0	1	1	1	0	0	0	0	1	0	0	Переход на ожидание положительного фронта

Табл.2.3. Микропрограмма дешифратора манчестерского кода (сигнал R x C обозначен в таблице C)

Необходимость данного УС становится понятной тогда, когда требуется подключать к компьютеру большое количество внешних устройств (по очереди или одновременно), особенно, нестандартных. Если для этого использовать индивидуальные, специализированные УС, то их потребуется очень много, из-за чего стоимость системы в

целом будет чрезмерно высокой, к тому же будет требоваться изменение ее конфигурации для подключения каждого нового внешнего устройства. Все используемые УС в этом случае будут включать в себя интерфейсную часть стандартного вида, что приведет к аппаратурной избыточности системы. Надежность системы в целом также будет невысока.

Совсем другое дело — использование многоразрядного контроллера параллельного обмена с двунаправленными внешними линиями, который путем простого изменения программного обеспечения может быть приспособлен для сопряжения с самыми разнообразными внешними устройствами или с несколькими внешними устройствами одновременно. Такое УС легко может эмулировать любые стандартные интерфейсы, его можно также применять в контрольно-диагностических системах, словом, спектр его применений очень широк.

Сформулируем требования к разрабатываемому контроллеру. Во-первых, он должен иметь большое количество линий связи с внешними устройствами (естественно, чем больше, тем лучше, но ограничением сверху здесь выступает количество контактов и размер используемого внешнего разъема, а также размер платы ISA). Во-вторых, для большей универсальности контроллера должна быть предусмотрена возможность простого изменения количества входных и выходных линий (в идеале все линии — двунаправленные, и направление передачи задается для каждой из них отдельно, но здесь также есть ограничение — это допустимое количество микросхем и размер печатной платы).

Исходя из этих соображений, мы будем разрабатывать 56-разрядный контроллер с побайтным управлением направления передачи. Как показывает практика, этого хватает для большинства возникающих задач.

При проектировании любой схемы УС можно рекомендовать следующий порядок. Сначала мы должны хотя бы приблизительно оценить функции интерфейсной части УС исходя из назначения УС (то есть какие режимы обмена с магистралью будут использоваться, какое требуется количество адресов, нужны ли прерывания, прямой доступ и т.д.). Это позволит наложить на операционную часть УС определенные ограничения.

Вторым этапом проектирования должна быть детальная разработка операционной части с учетом особенностей обмена с компьютером и с внешним устройством. И в заключение (третий этап) надо опять же вернуться к интерфейсной части УС, детализировав ее, приспособив для обмена с уже готовой операционной частью. Обычно этих трех этапов хватает, хотя в ряде случаев их может понадобиться и больше (то есть придется опять же вернуться к операционной части и т.д.).

Итак, в соответствии с этим алгоритмом, сначала сформулируем требования к интерфейсной части. При побайтном управлении внешними линиями имеет смысл ограничиться 8-разрядным обменом (скорость передачи информации здесь не очень важна), и тогда для обмена данными потребуется 7 адресов в адресном пространстве ввода/вывода. Помимо этого в УС надо записывать 7 бит управляющего слова, которое будет определять направление передачи информации для каждого из семи внешних 8-разрядных портов. Поэтому нашему УС необходимо в целом 8 адресов. Нужны ли здесь прерывания? Желательны, но не обязательны: можно ограничиться и опросом флага готовности. Прямой доступ не нужен совершенно.

Теперь переходим к операционной части. Начнем с конца, то есть с внешнего разъема (кстати, очень распространенный прием проектирования). Так как все внешние линии нашего УС двунаправленные, не исключена вероятность того, что на одну линию с разных концов (от контроллера и от внешнего устройства) будут поступать сигналы различных уровней. Например, выходной каскад нашего УС выдает сигнал логической единицы, а от внешнего устройства приходит сигнал логического нуля, или наоборот. В худшем случае это может привести к выходу из строя микросхем УС или внешнего

устройства. В лучшем случае на линии будет не тот уровень, который мы хотим, или этот уровень будет нестабильным. Особенно типична такая ситуация при использовании данного контроллера для диагностики и отдадки цифровых устройств. Поэтому здесь необходима какая-то защита, которая может быть реализована тремя взаимодополняющими методами:

— при включении питания компьютера все внешние линии контроллера должны переходить в состояние приема и только потом программно устанавливаться в нужный режим;

— необходимо программно контролировать правильность выдаваемой информации, причем желательно читать именно сигнал, приходящий от внешнего устройства, сравнивая который с выдаваемым сигналом, можно обнаружить конфликт и принять меры по его ликвидации (например, переключить линию на прием, подать нужные сигналы на внешнее устройство, проинформировать оператора и т.д.);

— требуется ограничить уровни выходных токов внешних линий контроллера (наиболее простое решение здесь — последовательно включенные во внешние линии резисторы с малым сопротивлением).

Выходные сигналы разрабатываемого УС должны быть достаточно мощными, чтобы работать с внешними устройствами, находящимися на расстоянии нескольких метров (хотя бы до 2 м) от компьютера. В режиме приема внешних сигналов линии должны обеспечивать малые входные токи, чтобы не перегружать внешние устройства. Вот собственно и все требования к операционной части УС (случай мы рассматриваем простейший). Исходя из них, операционная часть может выглядить следующим образом (рис. 2.38).

Выходные сигналы формируются регистрами с тремя состояниями выхода КР1533ИР37, каждый из которых может находиться в активном или пассивном состоянии в зависимости от значения соответствующего бита в управляющем слове, хранящемся в регистре на базе КР1533ТМ8. На внешний разъем выходные сигналы подаются через защитные резисторы с сопротивлением около 100 Ом. Для чтения состояний внешних линий используются однонаправленные буфера КР1533АП5, выходы которых объединяются для мультиплексирования читаемых данных. Таким образом, в случае конфликта при чтении по шине выходных данных мы получаем именно сигналы, приходящие от внешнего устройства. Сравнивая их с выдаваемыми сигналами, мы можем детектировать конфликтную ситуацию. Конечно же, использование защитных резисторов несколько снижает помехоустойчивость и нагрузочную способность, но настолько незначительно, что, как показывает практика, обычно этим можно пренебречь.

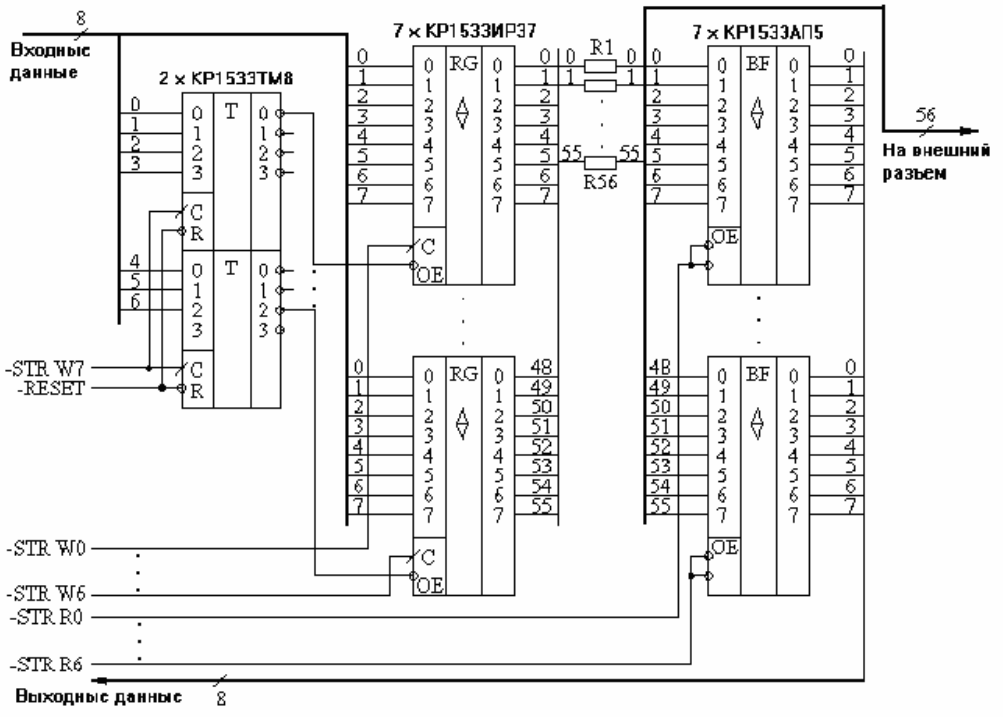


Рис. 2.38. Операционная часто универсального контроллера параллельного обмена информацией

Управляющие сигналы операционной части используются для записи выходных данных (-STR W0 ... -STR W6), для чтения входных данных (-STR R0 ... -STR R6), для записи управляющего слова (-STR W7), определяющего направление передачи каждого из семи 8-разрядных портов, а также для перевода всех внешних линий в состояние приема при включении питания (по сигналу -RESET). Именно эти сигналы должна формировать интерфейсная часть, к которой мы теперь возвращаемся.

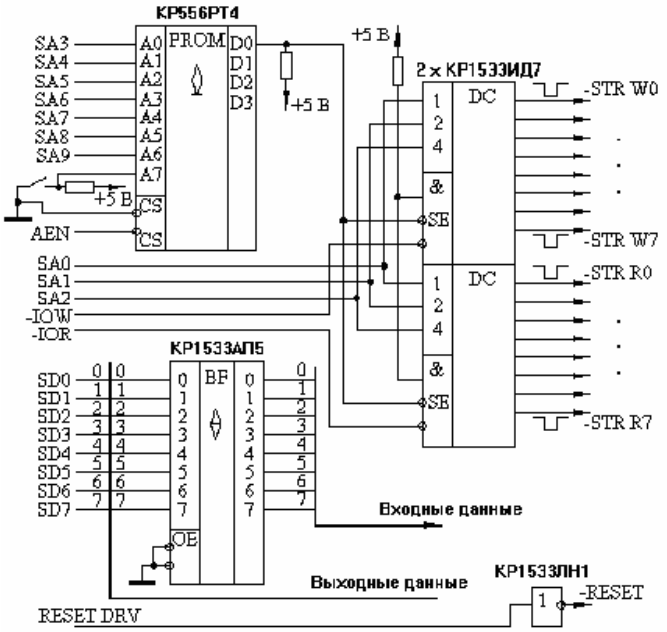


Рис. 2.39. Интерфейсная часть универсального контроллера параллельного обмена информацией

Одно из возможных решений интерфейсной части показано на рис. 2.39. Селектор адреса выполнен на ППЗУ КР556РТ4 и дешифраторах КР1533ИД7. При этом свободный адресный вход ППЗУ используется для переключения селектируемых адресов. Входные данные УС буферируются с помощью КР1533АП5. В адресном пространстве устройств ввода/вывода компьютера наше УС занимает зону в 8 адресов, семь из которых используются для чтения/записи 56 внешних линий, а один — для записи слова состояния. Проинвертированный магистральный сигнал RESET DRV задает начальную конфигурацию внешних линий (все — на прием). Нетрудно подсчитать, что в данном варианте наше УС включает в себя 21 микросхему.

Схема УС может быть дополнена узлом обработки прерываний от внешних устройств. Пусть УС должно принимать два сигнала прерываний и формировать один сигнал требования прерывания IRQ. При этом надо предусмотреть возможность для компьютера определить, какое именно прерывание из двух внешних поступило на УС, для чего удобно использовать незадействованный сигнал -STR R7. Возможное решение такой схемы обработки прерываний — на рис. 2.40. Внешними прерываниями являются положительные фронты сигналов INT1 или INT2, перебрасывающие триггеры, выходы которых формируют IRQ. Для чтения номера пришедшего прерывания использован буфер КР1533ЛП8, открывающийся сигналом -STR R7. Этим же сигналом триггера сбрасываются в исходное состояние.

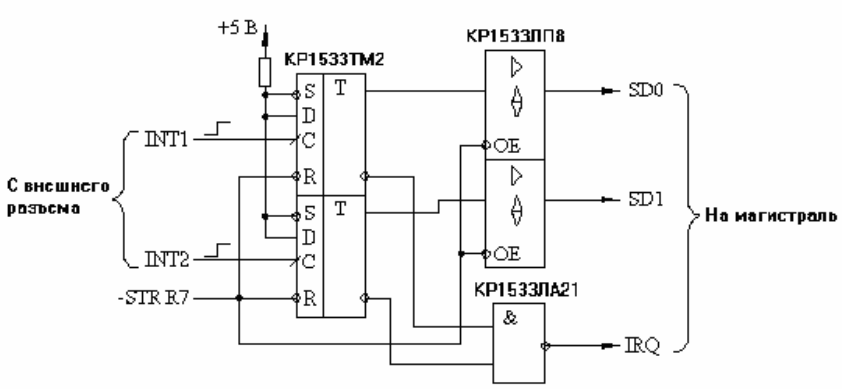


Рис.2.40. Обработка внешних прерываний в универсальном контроллере параллельного обмена информацией

Здесь же стоит упомянуть о еще одном возможном дополнении схемы нашего УС. Если на внешний разъем вывести стробы записи и чтения (STR R и STR W), то в ряде случаев это позволит значительно ускорить обмен с внешним устройством, которое сможет, например, записывать в свои внутренние регистры приходящие данные по стробу STR W.

Как уже отмечалось, с помощью предлагаемого УС и соответствующего программного обеспечения можно реализовать практически любой стандартный интерфейс или даже несколько интерфейсов одновременно (Centronics, IEEE 488, RS — 232 и т.д.). Необходимо только учитывать ограничение на быстродействие эмулируемого интерфейса, связанное с быстродействием компьютера. Точно так же можно организовать свой интерфейс, реализующий протокол, наиболее соответствующий решаемой задаче. При этом все УС можно разместить во внешнем конструктиве, а в компьютер установить только разработанный контроллер. Такой подход обеспечивает все преимущества вынесения УС из компьютера: снятие ограничений на сложность и количество УС, снижение наводок и помех и т.д., хотя и увеличивает стоимость системы в целом. По сравнению со стандартными интерфейсами компьютера в данном случае гораздо проще достигается сопряжение с большим количеством УС (все форматы и протоколы мы выбираем самостоятельно).

2.1.10. Одноплатный логический анализатор

Вторым примером схемы УС, который мы рассмотрим, будет котроллер, применяемый при отладке цифровой аппаратуры — логический анализатор, имеющий в своем составе многоразрядное буферное ОЗУ с узлами управления и синхронизации. Эту схему мы не будем описывать так же подробно, как первую.

Несколько вводных слов. Логический анализатор по своему назначению близок к осциллографу, так как он позволяет наблюдать на экране временные диаграммы сигналов. Но в отличие от обычного (не цифрового) осциллографа логический анализатор работает только с цифровыми двухуровневыми (реже трехуровневыми) сигналами, имеет большое количество входных линий (обычно от 16 до 64), работает в режиме однократного запоминания временных диаграмм (как запоминающий осциллограф) и имеет возможность предпусковой регистрации. Последнее требует некоторых пояснений (рис. 2.41).

В отличие от обычных осциллографов, в которых развертывание формы входного сигнала начинается в момент запуска (то есть прихода внешнего сигнала запуска или превышения входным сигналом заданного уровня напряжения), здесь точка запуска может быть и в начале, и в середине, и в конце окна регистрации. Под запуском здесь понимается

временная привязка процесса регистрации к исследуемому процессу. Запуском может служить например, появление в потоке данных заданного кода или переход (фронт) на одной из входных линий. В этом случае оператор может видеть не только то, что происходило после запуска (как в случае с обычным осциллографом), но и то, что происходило до него.

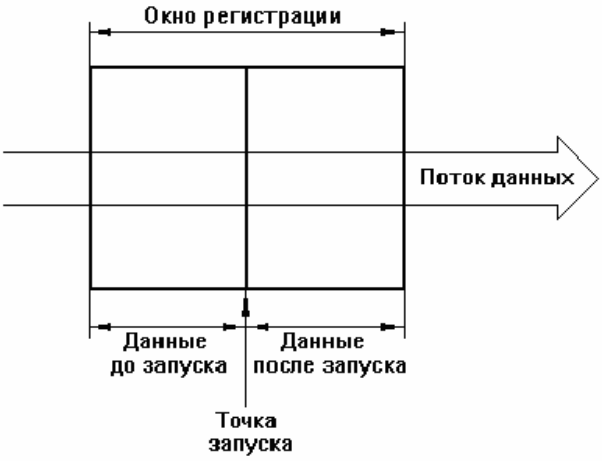


Рис. 2.41. Предпусковая регистрация

Логические анализаторы делятся на синхронные (или анализаторы логических состояний) и асинхронные (или анализаторы временных диаграмм). Синхронные анализаторы работают от тактового генератора исследуемой схемы и фиксируют только временные сдвиги, кратные его периоду, а следовательно, выявляют только нарушения в логике работы схемы. Асинхронные анализаторы работают от собственного внутреннего тактового генератора, поэтому они измеряют абсолютные значения временных сдвигов и могут выявлять ошибки из-за неправильно рассчитанных задержек, из-за емкостных эффектов и т.д. Они обычно делаются гораздо более быстрыми, чем синхронные анализаторы (рассчитываются на предельно возможную частоту регистрации)

Мы в качестве примера будем разрабатывать схему логического анализатора, не отличающуюся рекордными характеристиками ни в плане быстродействия, ни в плане количества разрядов, ни в плане развитости системы запуска. Достоинство ее в другом: она выполняется в виде платы расширения персонального компьютера, и, следовательно, при ее использовании оператор получает в свое распоряжение всю мощь этого компьютера: интеллект, средства ввода и отображения информации, дисковую память и т.д. В результате ценой незначительных дополнительных затрат (цена платы) мы можем превратить компьютер (на время или навсегда) в эффективный и очень удобный логический анализатор. Отметим, что это далеко не все преимущетсва данного подхода.

Исходные данные для проектирования примем следующие: количество входных линий (каналов регистрациия) — 32, количество регистрируемых состояний — 4096, максимальная тактовая частота — 10 МГц, тактовый генератор — внутренний с изменяемой частотой или внешний, запуск — по положительному или отрицательному переходу на одной из 8 входных линий, глубина предпусковой регистрации — задается программно.

Первый этап проектирования в соответствии с описанным алгоритмом — предварительная оценка интерфейсной части. Прежде всего, посмотрим, какие режимы обмена с магистралью нужны в данном случае. Для обеспечения нужного темпа приема данных (до 10 МГц) совершенно необходимо буферное ОЗУ, обмен с которым должен быть периодическим: при регистрации оно заполняется в темпе тактового генератора, по

окончании регистрации его содержимое считывается компьютером. Нужно ли максимально ускорять этот процесс считывания? Зарегистрированная информация должна обрабатываться и отображаться на экране с целью анализа ее оператором. Этот процесс неизмеримо более длительный, чем перекачка информации из буферного ОЗУ в системное ОЗУ компьютера. Поэтому в данном случае особой скорости обмена, по-видимому, не требуется. Конечно же, можно организовать 16-разрядный обмен с нашим УС, дающий большой выигрыш во времени по сравнению с 8-разрядным обменом, но зато он требует дополнительных аппаратурные затрат (вдвое больше буферов данных, формирование сигнала -I/O CS 16). К тому же в этом случае усложняется проектирование печатной платы (нужен второй магистральный разъем). Исходя из всех этих соображений, имеет смысл остановиться на 8-разрядном обмене и отказаться от использования ПДП.

Что касается прерываний, то для логического анализатора их использование очень желательно, если не необходимо. Ведь между началом регистрации и ее окончанием, связанным исключительно с приходом запуска, может пройти довольно большое время. Целесообразно предусмотреть возможность выполнения в этот период компьютером других задач. Поэтому прерывание по окончании регистрации мы будем формировать, что, впрочем, не исключает необходимости чтения флага готовности.

Таким образом, интерфейсная часть нашего одноплатного анализатора должна обеспечивать следующие параметры. Количество адресов УС в адресном пространстве устройств ввода/вывода — 5, из которых четыре используются для чтения зарегистрированных данных, а один — для чтения флага готовности. Для записи управляющего слова будем использовать два из этих пяти адресов (надо записать глубину предпусковой регистрации, код выбора тактовой частоты, код выбора запуска). Используем одно прерывание по окончании регистрации. Как видим, ничего сложного здесь нет, поэтому к интерфейсной части мы не возвращаемся.

Переходим к операционной части. Основные ее узлы: буферное ОЗУ объемом 128 Кбит с организацией 4К х 32, счетчик для перебора адресов , внутренний тактовый генератор с программно изменяемой частотой, схема запуска и входные буфера для регистрируемых сигналов.

ОЗУ целесообразно выполнить на многоразрядных микросхемах (для снижения количества корпусов). Требования к его быстродействию в данном случае невысоки (при максимальной тактовой частоте 10 МГц в течение 100 нс необходимо переключить счетчик адресов и записать входную информацию в ОЗУ). Таких микросхем, особенно зарубежного производства, достаточно много.

От счетчика требуется максимальное быстродействие (можно взять, например, микросхемы КР531ИЕ17, которые достаточно легко каскадируются без потери быстродействия). Кроме простого перебора адресов счетчик должен также обеспечивать предпусковую регистрацию. Остановимся на этом несколько подробнее. Для того, чтобы реализовать предпусковую регистрацию, необходимо до прихода запуска непрерывно переписывать содержимое буферного ОЗУ по кругу (рис. 2.42.). Если мы выбираем глубину предпусковой регистрации N тактов, то надо остановить регистрацию через (4096 — N) тактов после прихода запуска. Затем надо считывать содержимое ОЗУ, начиная с точки остановки с перебором адресов в том же направлении, что и при регистрации. Проведя 4096 операций чтения содержимого ОЗУ, мы получим N тактов до запуска и (4096 — N) тактов после запуска, то есть моменту прихода запуска будет соответствовать адрес ОЗУ, считанный N-ым.

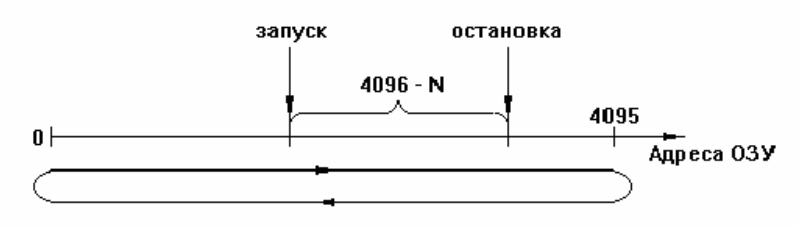


Рис. 2.42. Реализация предпусковой регистрации

Однако все произойдет именно так только в том случае, если от момента начала регистрации до момента прихода запуска наш анализатор успеет зафиксировать N тактов. Иначе мы не перепишем все ОЗУ, и в части его адресов будет находиться предыдущая информация. Чтобы избежать этого, надо запретить реакцию на запуск в течение N тактов после начала регистрации (выдержать своеобразное "мертвое" время). А что будет, если запуск придет в течение этого "мертвого" времени? Если исследуемый процесс периодический, то анализатор среагирует на следующий запуск. Если же процесс однократный, то надо начать процесс регистрации заведомо раньше (на "мертвое" время или больше), чем начнется изучаемый процесс (например, если мы исследуем старт компьютера при включении питания).

В результате счетчики анализатора должны обеспечивать временную диаграмму, показанную на рис. 2.43. Адреса ОЗУ начинают перебираться с началом регистрации. В течение N тактов после начала регистрации запуск запрещен. Через (4096 — N) тактов после прихода запуска регистрация прекращается.

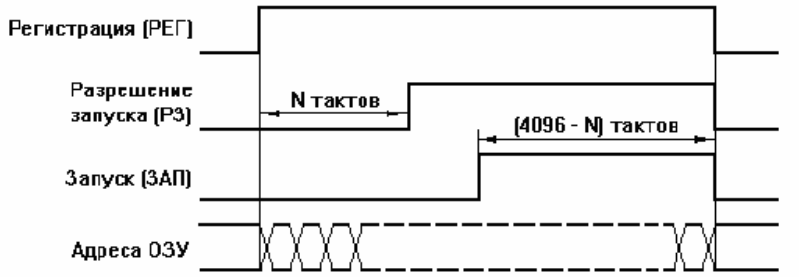


Рис. 2.43. Временная диаграмма работы счетчиков логического анализатора

Отметим, что точно так же может быть реализована предпусковая регистрация в цифровом осциллографе, который, кстати, тоже можно выполнить в виде одноплатного УС, сопрягаемого с системной магистралью. По сравнению с логическим анализатором в схему надо будет добавить одно или несколько АЦП и некоторые другие цифро-аналоговые узлы.

Что касается остальных узлов логического анализатора, то они не представляют особого интереса, поэтому сразу обратимся к функциональной схеме всего УС (рис. 2.44). Здесь мы уже не разрисовываем подробно интерфейсную часть, как мы делали при рассмотрении предыдущего УС, так как ничего принципиально нового она не содержит.

Тактовый генератор выполнен на счетчике Сч.1 и мультиплексоре М1. Он может выдавать ряд частот, различающихся в 2 раза (период 100, 200, 400, 800, 1600, 3200, 6400 нс) или внешний тактовый сигнал ВТС. То есть здесь реализуются как синхронный, так и асинхронный режим работы. В качестве запуска используется положительный или отрицательный переход на одной из восьми входных линий, выбираемых мультиплексором М2 (полярность перехода задается управляемым инвертором на элементе "Исключающее ИЛИ"). 7-разрядное управляющее слово записывается в регистр управляющего слова РУС по сигналу ЗУС (STR W0).

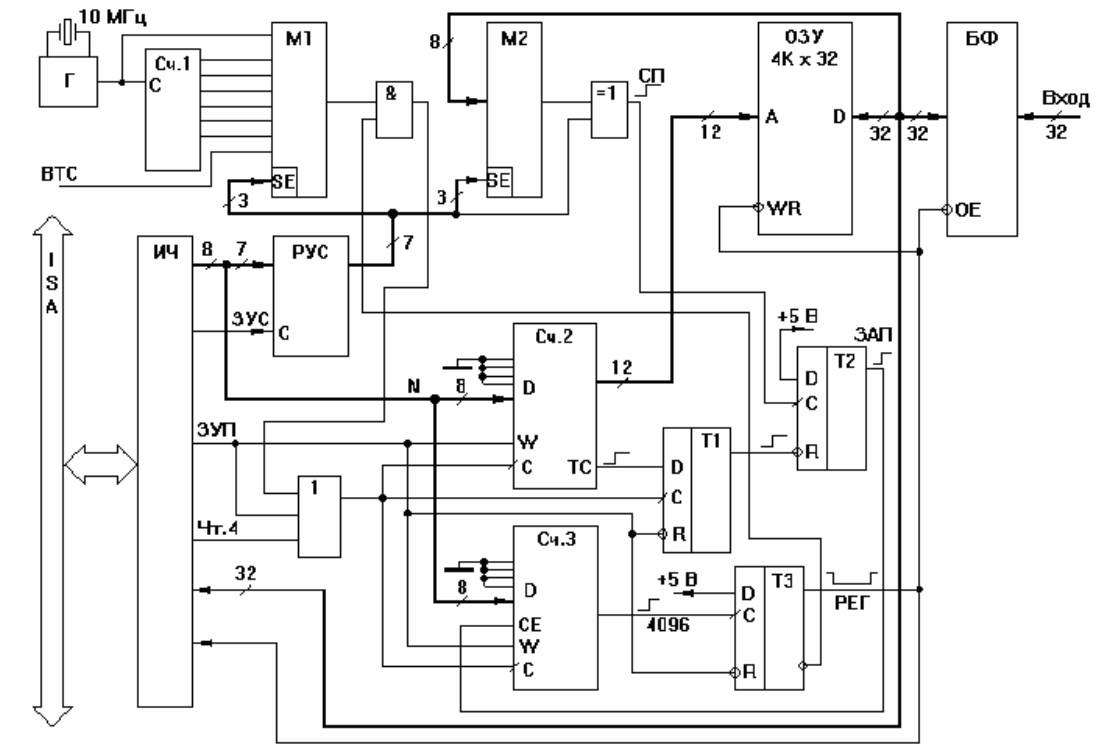


Рис. 2.44. Функциональная схема логического анализатора

Перед началом работы необходимо записать в наш анализатор управляющее слово и упреждение (глубину предпусковой регистрации). 8-разрядный код упреждения N записывается в два 12-разрядных счетчика Сч.2 и Сч.3 по сигналу ЗУП (STR W1). При этом на четыре младших входных разряда счетчиков подается сигнал логического нуля (то есть упреждение задается с точностью до четырех тактов и выбирается из ряда: 0, 16, 32, 48, 64, ..., 4080 тактов).

Запись упреждения служит стартом регистрации (сбрасываются триггера T1, T2, T3, начинает считать Сч.2, ОЗУ переходит в режим записи, а входной буфер БФ — в режим пропускания). Счетчик Сч.2 работает в режиме вычитания (обратного счета). После того, как он отсчитал N тактов, перебрасывается триггер T1 (заканчивается "мертвое" время). После этого T2 начинает реагировать на синхропереход СП. После поступления синхроперехода триггер T2 перебрасывается и разрешает работу Сч.3, работающего в режиме прямого счета. Затем Сч.3 отсчитывает (4096 — N) тактов и перебрасывает T3. На этом регистрация закончена.

Сигнал РЕГ с выхода T3 служит флагом готовности и используется для выработки прерывания. По окончании регистрации по четырем адресам компьютер считывает записанную в ОЗУ 32-разрядную информацию. При этом сигнал чтения из четвертого адреса Чт.4 (STR R3) перебрасывает на единицу Сч.2, то есть изменяет адрес ОЗУ. Этот процесс чтения повторяется 4096 раз. В результате полный цикл перекачки зарегистрированной информации включает в себя 16384 обращения к логическому анализатору.

Таким образом, мы рассмотрели проектирование довольно типичного УС с буферным ОЗУ с периодическим режимом обмена. Данный модуль логического анализатора может быть использован в системе динамической отладки УС, особенности которой мы рассмотрим позже.

2.1.11. Генератор сигналов произвольной формы

Еще один пример УС с буферным ОЗУ — это цифровой программно-управляемый генератор сигналов произвольной формы, позволяющий формировать аналоговые сигналы с заданными параметрами: формой, периодом, амплитудой. Такой генератор может использоваться в контрольно-измерительной системе на базе персонального компьютера или в составе промышленных установок, в которых требуется формирование различных сигналов (разовых или периодических), например, в системах виброиспытаний.

С точки зрения разработчика УС, такой генератор — это УС, имеющее в своем составе буферное ОЗУ с периодическим режимом обмена с внешним устройством. Перед началом работы компьютер записывает в буферное ОЗУ коды выборок генерируемого сигнала, задавая его форму, определяет период сигнала и его амплитуду, а также режим запуска генерации: разовый или автоматический. Затем дается старт генерации, во время которой выходные коды буферного ОЗУ поступают на ЦАП, преобразующий их в уровни выходного аналогового сигнала. Программные средства генератора при этом могут обеспечивать различные методы задания формы сигнала: выбор стандартного сигнала (синусоидальный, прямоугольный, треугольный, пилообразный, шумовой и т.д.), формирование сигнала по математической формуле, задание и коррекция формы на экране компьютера.

Мы будем разрабатывать генератор со следующимим характеристиками: частота выходного сигнала — 2 Гц ... 125 кГц, амплитуда выходного сигнала — 50 мВ ... 10 В. Следует учесть, что сигналы сложной формы нужны обычно низкочастотные, поэтому выбранные параметры обеспечат довольно высокую универсальность генератора. В схеме должны быть предусмотрены режим разового запуска (остановка генерации после одного периода сигнала) и режим автоматического запуска (непрерывная генерация до ее программной остановки).

Оценим предварительно требования к интерфейсной части УС. Если ориентироваться на 8-разрядный обмен (скорость обмена здесь, как и в случае с логическим анализатором, некритична), то потребуется один адрес для обмена с буферным ОЗУ (запись и чтение, которое нужно для самотестирования), два адреса (16 разрядов) для записи кода частоты (периода), один адрес для записи кода амплитуды и один адрес для записи управляющего слова и чтения слова состояния. Итого пять адресов. Отметим, что увеличивать количество разрядов выходного ЦАП больше, чем до восьми, особого смысла в данном случае не имеет. В управляющее слово входят два бита: разрешение/запрет генерации и разовый/автоматический запуск. В слово состояния входит всего один бит (генерация идет), который нужен только при разовом запуске. Прямой доступ здесь не нужен, прерывание — тоже, так как даже по окончании разового запуска никаких срочных действий предприниматься не должно. То есть ничего особенного интерфейсная часть собой не представляет.

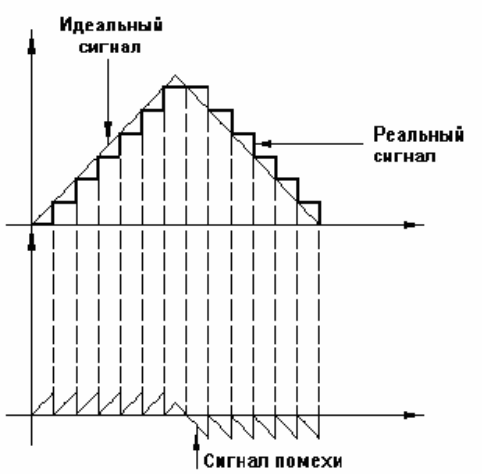


Рис. 2.45. Синтез аналогового сигнала по выборкам

Для задания частоты в цифровых генераторах (или синтезаторах, как их еще называют) сигналов произвольной формы наиболее часто используются два метода. Согласно первому из них, адреса буферного ОЗУ перебираются обычным двоичным счетчиком, а для изменения частоты выходного сигнала меняется частота, с которой эти адреса перебираются. В этом случае всегда опрашиваются все адреса ОЗУ, то есть количество выборок на период выходного сигнала не изменяется при изменении частоты, а значит, не изменяется и точность воспроизведения формы сигнала. Недостатком такого метода является то, что он хорошо работает только в области низких и инфранизких частот выходного сигнала, так как большие частоты требуют в данном случае очень высокого быстродействия ЦАП. Еще один существенный недостаток такого подхода состоит в том, что частота сигнала помехи, возникающей из-за квантования уровней выходного сигнала, здесь прямо пропорциональна частоте выходного сигнала (рис. 2.45). Поэтому фильтрация этого сигнала помехи (обычно необходимая) довольно трудоемка и требует специальных перестраиваемых фильтров.

Второй метод задания частоты выходного сигнала несколько сложнее (рис. 2.46). В этом случае для перебора адресов буферного ОЗУ используется не счетчик, а накапливающий сумматор, состоящий из двоичного сумматора и регистра, охваченных обратной связью. При этом с каждым следующим импульсом тактового генератора к выходному коду регистра прибавляется входной управляющий код и полученная сумма снова записывается в регистр. В результате в каждом такте приращение адреса ОЗУ будет определяться входным управляющим кодом накапливающего сумматора, изменяя который, мы можем изменять скорость прохождения всех адресов ОЗУ и, следовательно, частоту выходного аналогового сигнала. Если обозначить входной код N, частоту тактового генератора f, а количество адресов ОЗУ — M, то выходная частота F = fN/M, то есть пропорциональна входному коду.

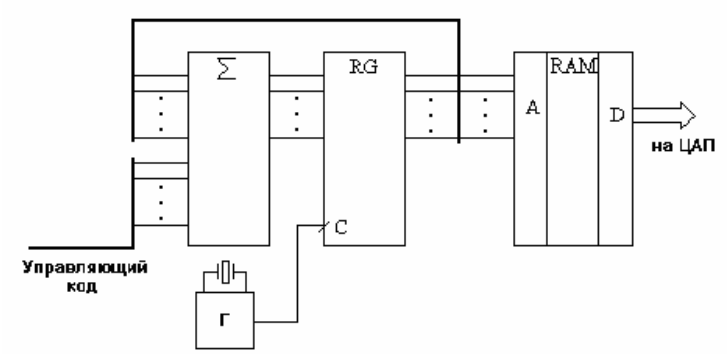


Рис. 2.46. Перебор адресов ОЗУ с помощью накапливающего сумматора

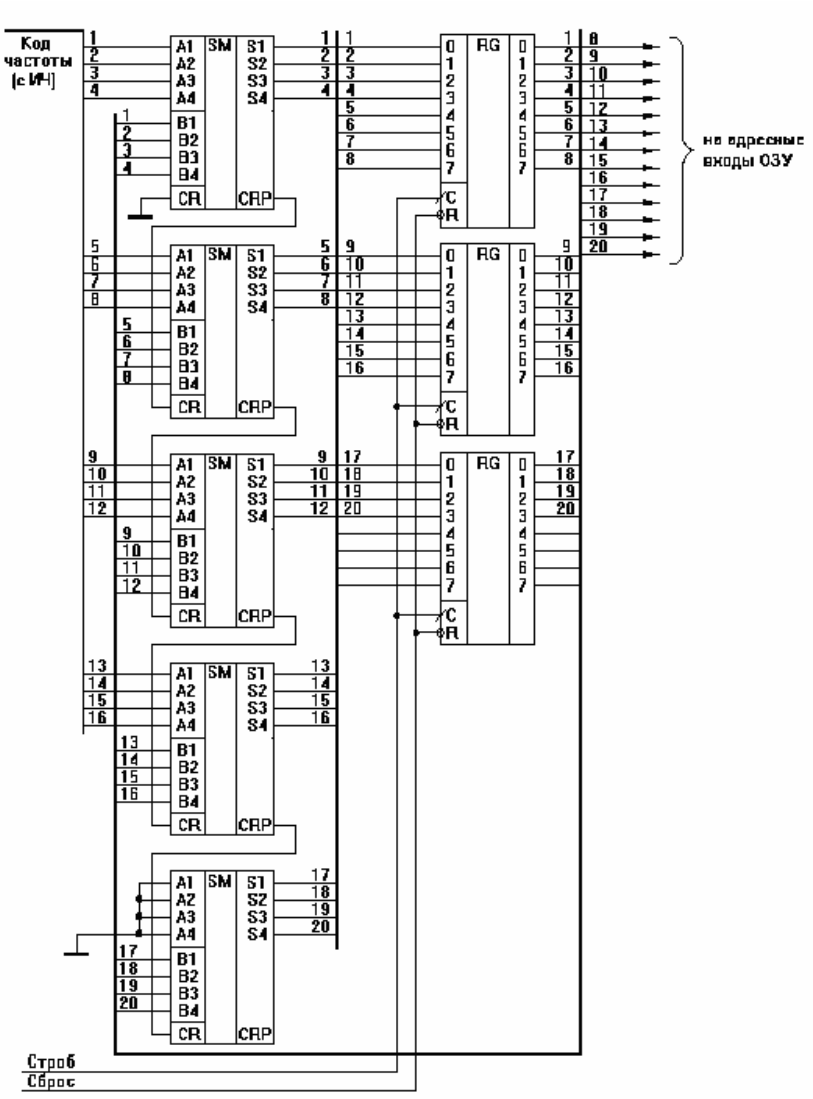


Рис. 2.47. Схема накапливающего сумматора

Недостаток этого метода — изменение количества выборок на период выходного сигнала обратно пропорционально его частоте, то есть форма сигнала воспроизводится с

разной точностью на разных частотах. Но его большое преимущество состоит в том, что частота сигнала помехи будет постоянна (она равна f) и, следова-тельно, отфильтровать этот сигнал очень просто с помощью самого простого (не перестраиваемого) фильтра нижних частот. Частота выходного сигнала при такой схеме задается с постоянным шагом во всем частотном диапазоне, поэтому относительная погрешность ее установки минимальна в верхней части частотного диапазона.

Мы будем разрабатывать генератор на основании именно этого второго метода задания частоты. Для обеспечения требуемого частотного диапазона при минимальном количестве выборок на период выходного сигнала, равном 16, накапливающий сумматор должен иметь 20 разрядов, а тактовая частота должна быть 2 МГц. С такой частотой могут успешно работать многие микросхемы ЦАП, например, К1108ПА1А (время преобразования не более 400 нс). В схему накапливающего сумматора (рис. 2.47) входят 5 микросхем 4-разрядных полных сумматоров К155ИМ3 (задержка не более 50 нс) и 3 микросхемы 8-разрядных регистров со сбросом КР1533ИР35 (задержка не более 15 нс). То есть предельная рабочая частота схемы около 15 МГц. Старшие 4 разряда кода задания частоты здесь не используются. На адресные входы ОЗУ с кодами выборок выходного сигнала подается требуемое число старших разрядов выходного кода накапливающего сумматора.

Полная структурная схема рассматриваемого УС (рис. 2.48) помимо накапливающего сумматора (НС) включает в себя управляющие регистры, схему запуска, буферное ОЗУ объемом 8К х 8, выходной регистр (ВР), ЦАП1, преобразующий коды выборок из ОЗУ в напряжение выходного сигнала и умножающий ЦАП2, задающий амплитуду выходного сигнала. В регистр частоты (РЧ) по сигналу записи кода частоты заносится шаг накапливающего сумматора (НС). В регистр амплитуды (РА) записывается код амплитуды по сигналу ЗКА. Управляющее слово из двух бит заносится в триггеры Т1 и Т2 по сигналу ЗУС. Выходной сигнал Т1 определяет режим генерации или останова, а выходной сигнал Т2 — режим запуска (разовый или автоматический).

Перед началом работы схема переводится в режим останова, задается шаг накапливающего сумматора, соответствующий единице младшего разряда адреса ОЗУ, и производится сброс накапливающего сумматора. Затем проводится запись кодов выборок в ОЗУ. При этом по сигналу записи в ОЗУ или чтения из ОЗУ(зап/чт) производится наращивание адреса ОЗУ. После окончания заполнения ОЗУ задается частота и амплитуда сигнала, и схема переводится в режим генерации, в котором на тактовый вход накапливающего сумматора поступает сигнал с кварцевого генератора. Выходной регистр (ВР) служит для обеспечения одновременности изменения всех разрядов входного кода ЦАП1 с целью уменьшения коммутационных помех ЦАП. ЦАП2 умножающего типа (К572ПА1) умножает выходной сигнал ЦАП1 на код амплитуды с РА.

Фильтр низкой частоты (ФНЧ) с частотой среза, расположенной между верхней частотой выходного аналогового сигнала и частотой кварцевого генератора (Г), служит для сглаживания выходного сигнала. Он отсекает помеховый сигнал (рис. 2.45). Выход 1 используется в том случае, когда требуется хороший фронт выходного сигнала, не сглаженный ФНЧ.

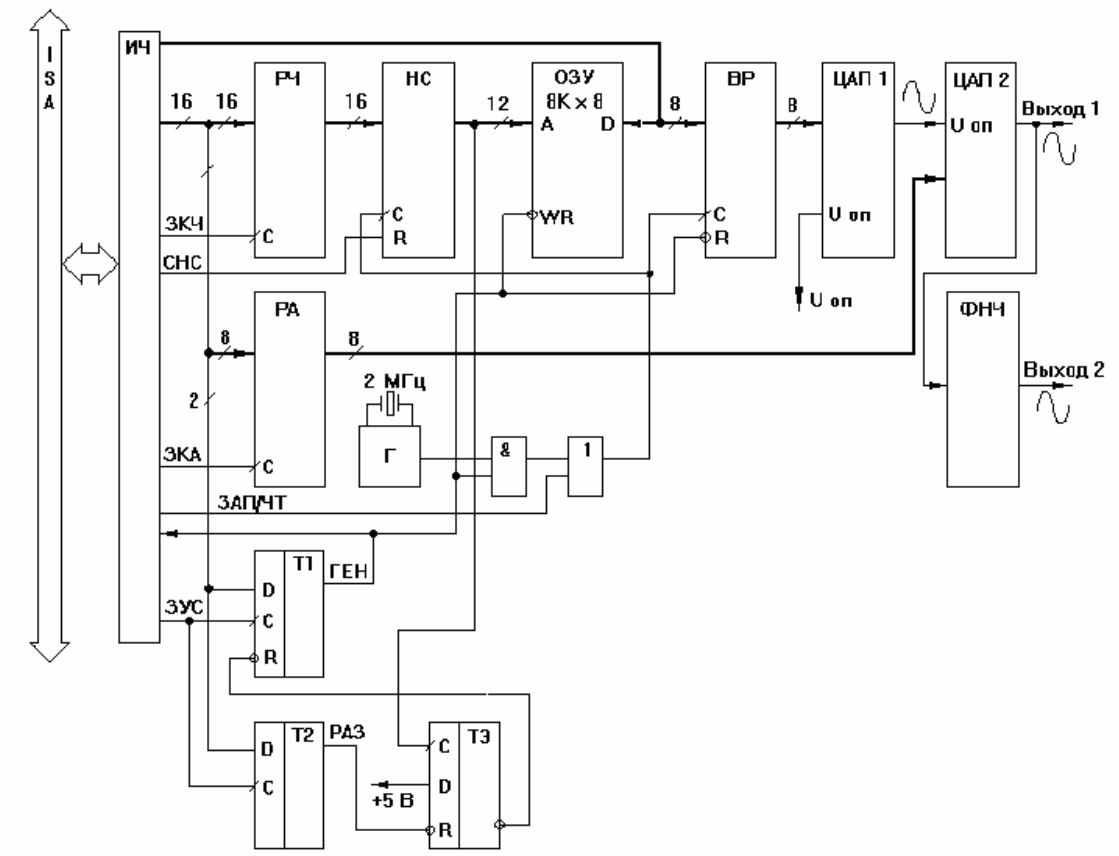


Рис. 2.48. Функциональная схема генератора сигналов произвольной формы

В режиме разового запуска после одного периода выходного сигнала схемы старший разряд кода адреса ОЗУ перебрасывает триггер Т3 и останавливает генерацию. Флагом окончания генерации (единственным битом слова состояния) при этом служит выход триггера Т1.

Интерфейсная часть (ИЧ) рассматриваемого генератора, как уже отмечалось, никаких осбенностей не имеет и поэтому не будет рассматриваться подробно.

2.1.12. Измеритель частоты следования импульсов

Еще один тип УС — это контоллеры ввода/вывода частотно-временных параметров. Сюда относятся всевозможные генераторы прямоугольных импульсов с программно-управляемой частотой и скважностью, измерители и формирователи временных интервалов, а также измерители частоты. В основе всех подобных схем — счетчики, считающие импульсы образцовой или измеряемой частоты.

В рассматриваемом нами случае входным параметром будет частота следования прямоугольных импульсов. Простейший пример практического использования подобного УС — это измерение напряжения с помощью удаленного от компьютера датчика, в качестве которого выступает преобразователь напряжение-частота. Это довольно типичная ситуация в системах управления производственными процессами. Если применять в качестве линии передачи оптоволоконный кабель и оптоволоконные передатчик и приемник на входе и выходе, то датчик можно располагать на расстоянии в несколько километров от компьютера. Другое использование данного УС — в составе контрольно-

измерительных систем, при отладке различных электронных устройств.

Цифровое измерение частоты обычно сводится к подсчету количества импульсов входного сигнала в течение заданного интервала времени (временного окна). Этот метод иллюстрируется рис. 2.49. Однако он имеет существенные недостатки. Если длительность временного окна выбрана постоянной, то диапазон измеряемых частот будет невелик, и, что еще более важно, точность измерения частоты будет существенно зависеть от самой частоты. Очевидно, что при низкой входной частоте эта погрешность будет очень большой, так как количество сосчитанных импульсов будет мало.

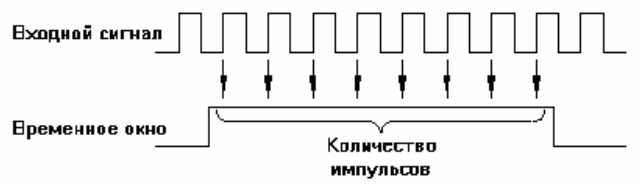


Рис. 2.49. Измерение частоты входного сигнала

Другой метод измерения частоты — косвенный: измеряется период входного сигнала, для чего подсчитывается количество импульсов образцовой частоты в течение периода (рис. 2.50), и затем вычисляется обратная ему величина. И этот метод имеет свои недостатки.Здесь противоположная ситуация: если частота входного сигнала велика, то точность измерения периода, а значит, и частоты, будет низкой, так как количество сосчитанных импульсов будет мало.

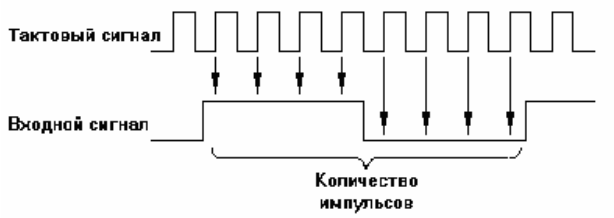


Рис. 2.50. Измерение периода входного сигнала

Чтобы обеспечить требуемую точность измерения частоты (периода) входного сигнала во всем частотном диапазоне, предлагается измерять длительность заданного количества периодов входного сигнала, а затем вычислять частоту по формуле: F = fM/N, где f — частота опорного тактового генератора, M — количество периодов входного сигнала, N — количество периодов тактового генератора (рис. 2.51). Задавая величину M и получая величину N, мы вычисляем F. При этом так как относительная погрешность измерения обратно пропорциональна N, то, выбирая M и, следовательно, N, мы можем обеспечить заданную точность измерения. Алгоритм выбора здесь не очень сложен, и за несколько циклов измерения можно определить частоту с нужной погрешностью. Для этого мы вполне можем ипользовать интеллект нашего компьютера, оставив на долю аппаратуры УС только подсчет импульсов тактового генератора в течение времени поступления заданного количества импульсов входного сигнала.

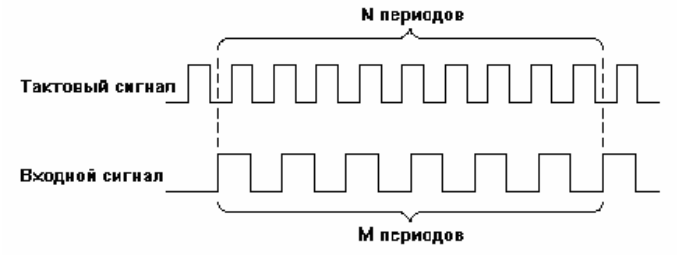


Рис. 2.51. Измерение M периодов входного сигнала

Один из вариантов алгоритма сводится к тому, что сначала производится измерение длительности одного периода входного сигнала, а затем (если необходимо) количество измеряемых периодов входного сигнала увеличивается во столько раз, чтобы погрешность подсчета импульсов тактовой частоты была ниже заданной. При этом потребуется 1, 2 или максимум 3 цикла измерения. Другой вариант алгоритма (адаптивный) состоит в том, что в первом цикле данного измерения берется то число периодов входного сигнала, которое было получено в предыдущем измерении (то есть здесь мы считаем, что частота входного сигнала не изменяется очень быстро, что справедливо в большинстве случаев). При этом довольно часто бывает достаточно только одного цикла измерения.

Переходим к проектированию схемы УС. Зададим количество разрядов нашего измерителя равным 16. Нам потребуется один адрес УС, доступный по записи и чтению (для записи M и чтения N). Можно использовать и 8-разрядный обмен, но тогда надо будет выделить нашему УС два адреса. Еще один адрес нужен для чтения флага готовности по окончании одного цикла измерения. Прямой доступ здесь, естественно, не нужен. Прерывание может быть удобно, особенно если на компьютер будут возложены и другие задачи. Таким образом, к интерфейсной части УС особых требований не предъявляется, и мы ее подробно рассматривать не будем.

Операционная часть должна включать в себя тактовый генератор, два счетчика (для тактового сигнала и для входного сигнала) и схему управления. Для уменьшения времени измерения необходимо выбирать самую высокую частоту тактового генератора и, следовательно, максимально быстродействующие счетчики. Возьмем, например, частоту 20 МГц и микросхемы КР1554ИЕ18 (4-разрядные синхронные, легко каскадируемые двоичные счетчики), которых потребуется 8 штук (по 4 на каждый из двух 16-разрядных счетчиков).

Функциональная схема УС показана на рис. 2.52. Работа измерителя частоты начинается с записи числа (65536 — M) в счетчик Сч.1. При этом устанавливается триггер T1, разрешая запись единицы в триггер T2, и сбрасывается счетчик Сч.2. После прихода первого входного импульса устанавливается триггер T2, разрешая работу счетчиков Сч.1 и Сч.2. Этим мы обеспечиваем временную привязку процесса измерения к входному сигналу. Отсчитав M входных импульсов, Сч.1 перебрасывает триггер T3 и сбрасывает T1 и T2. Компьютер узнает об окончании цикла измерения по сигналу ИЗМ и читает выходной код Сч.2, а затем вычисляет частоту входного сигнала. При необходимости цикл измерения повторяется.

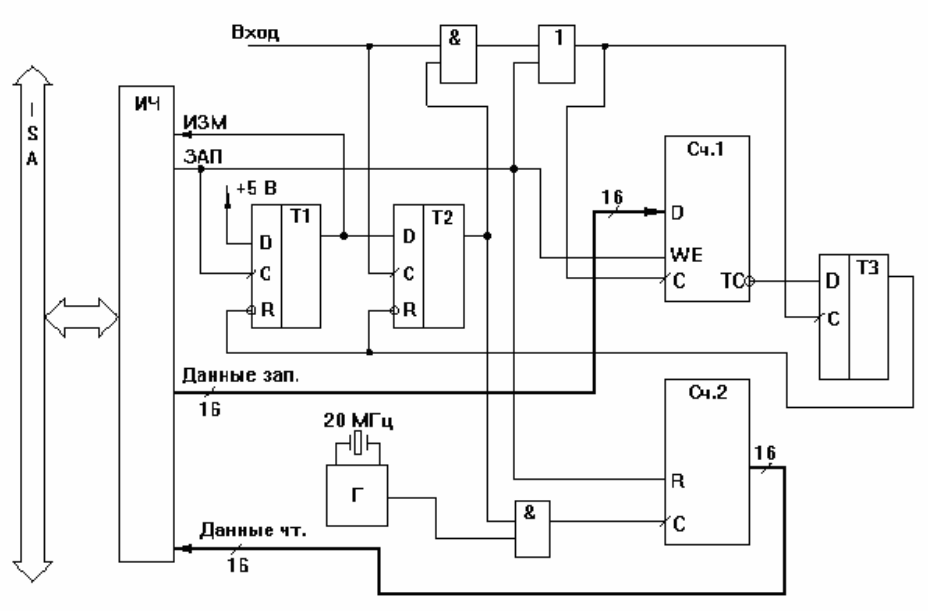


Рис. 2.52. Функциональная схема измерителя частоты

2.1.13. Узлы контроллера локальной сети

Последний из рассматриваемых в данной главе примеров — это котроллер локальной сети, который представляет собой УС для обмена информацией между отдельными компьютерами и организации их совместной работы.

Как известно, локальные сети (в отличие от глобальных) применяются для связи сравнительно близко расположенных компьютеров (характерные расстояния — до нескольких километров). При этом используется последовательная передача данных, которая позволяет сократить количество соединительных проводов, обеспечить высокую помехозащищенность линии передачи и существенно упростить приемо-передающие узлы контроллеров по сравнению с параллельной передачей. Скорости передачи в локальных сетях обычно достигают 1 — 10 Мбит/с и даже выше, поэтому стандартный интерфейс RS-232C для них не подходит. В качестве линий связи наиболее часто используют коаксиальный кабель, витую пару проводов и оптоволоконный кабель.

Типичными конфигурациями (топологиями) локальных сетей являются звезда, кольцо и шина (рис. 2.53). При топологии типа звезда существует центральный абонент (компьютер), управляющий всем обменом в сети, к которому подключаются остальные абоненты. Кольцевая топология предполагает последовательное соединение абонентов в замкнутую цепочку. При шинной топологии все абоненты параллельно подключаются к линии связи.

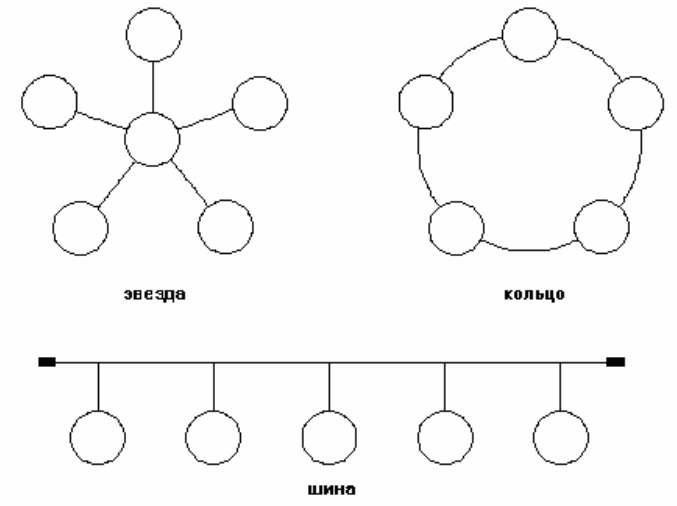


Рис. 2.53. Типичные топологии локальных сетей

С точки зрения аппаратуры котроллера локальной сети, при звездной топологии один контроллер (центральный) должен быть очень сложным, а остальные (периферийные) — довольно простыми. На каждом отрезке линии связи при этом существуют только два абонента: приемник и передатчик. Это же справедливо и для кольцевой топологии, но все контроллеры должны быть одинаковыми. При шинной топологии все контроллеры одинаковы и все они работают на одну линию связи, то есть в любой момент в сети могут работать несколько передатчиков одновременно, что приводит к наложению передаваемых сигналов. Поэтому контроллеры сети данного типа, как правило, наиболее сложные.

Вообще контроллер локальной сети должен выполнять множество функций, среди которых буферирование передаваемых и принимаемых данных, преобразование параллельной информации в последовательную и обратно, кодирование и декодирование данных, управление доступом к сети, контроль за ошибками передачи и т.д. Рассмотрение всех подходов к реализации этих функций потребовало бы отдельной книги, поэтому мы остановимся здесь только на отдельных узлах УС данного типа. Кстати, пример устройства дешифратора одного из наиболее распространенных кодов — манчестерского — приведен в разделе 2.1.8.

Особенностью контроллеров локальных сетей как особого типа УС является то, что здесь как раз очень желательно достижение близких к предельным скоростей обмена данными между УС и компьютером и между УС и линией связи. Это позволяет, с одной стороны, обеспечить эффективный обмен информацией между включенными в сеть компьютерами, а с другой стороны, что не менее важно — снизить нагрузку на линию связи и уменьшить вероятность перегрузки сети.

Рассмотрим следующую задачу. Необходимо разработать схему пословной (16 бит) передачи последовательной информации в сеть без буферирования со скоростью 10 Мбит/с. Эта скорость соответствует 1,25 Мбайт/с, что обеспечивается быстродействием ISA. На передачу 16 бит информации при этом потребуется 1,6 мкс. Пример реализации узла, решающего данную задачу, показан на рис. 2.54.

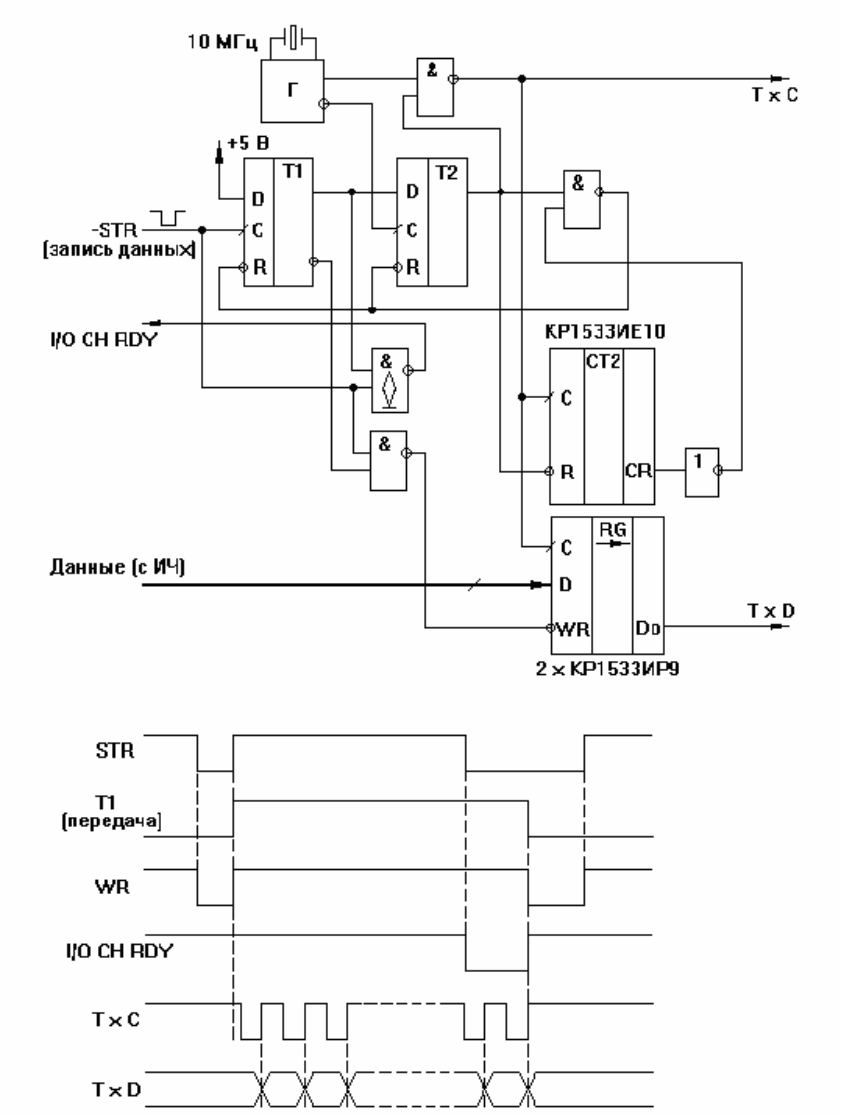


Рис.2.54. Функциональная схема узла последовательной передачи данных и временные диаграммы его работы

По сигналу записи данных STR начинается процесс передачи данных. Данные записываются в сдвиговый регистр и затем сдвигаются 16 раз (отсчитывается счетчиком). Цепочка триггеров T1 и T2 обеспечивает привязку начала процесса передачи к ближайшему
импульсу генератора тактовых сигналов Г. Если очередной сигнал STR приходит раньше, чем закончена передача предыдущего слова, то формируется задерживающий процесс обмена по ISA сигнал I/O CH RDY. Его максимальная длительность не будет превышать в данном случае 1,6 мкс, что удовлетворяет требованиям стандарта.

Обмен по сети без буферного ОЗУ имеет довольно ограниченное применение. Он чаще используется тогда, когда допускается побайтная или пословная передача, например, в оптоволоконных сетях. Однако обычно требуется неразрывная передача целого массива (называемого также пакетом или кадром) со своей структурой и, соответственно, прием этого массива. В этом случае необходимо применять буферное ОЗУ и, конечно же,

обеспечить максимальную скорость обмена компьютера с ним. Как уже отмечалось в 2.1.7, наибольшее быстродействие достигается при применении разделяемой памяти или (что то же самое) при параллельном доступе к буферному ОЗУ (то есть каждому адресу буферного ОЗУ соответствует свой адрес в адресном пространстве памяти компьютера). Рассмотрим практическую реализацию этого подхода при проектировании контроллера локальной сети.

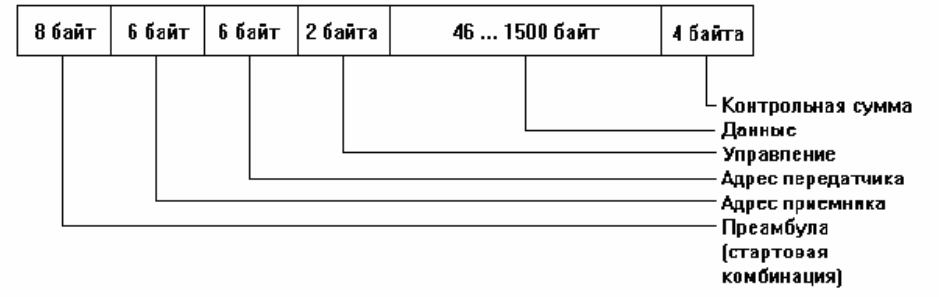


Рис. 2.55. Структура кадра локальной сели Ethernet

Прежде всего отметим, что особенностью обмена по сети является использование пакетов (кадров) самой различной длины (обмен пакетами стандартной длины тоже встречается, но гораздо реже). Помимо информации о сетевых адресах приемника и передатчика пакета, начальной и конечной комбинаций пакета, а также управляющей информации в пакет могут входить или не входить данные, причем объем этих данных может быть самым различным. Примером может служить структура пакета (кадра) широко распространенной локальной сети Ethernet (рис. 2.55). Поэтому важно обеспечить, чтобы буферное ОЗУ контроллера сети могло выдавать в сеть и принимать из сети пакеты различной длины.

Особых требований по быстродействию микросхем буферного ОЗУ здесь не предъявляется, так как при скорости обмена по сети 10 Мбит/с и 16-разрядной организации буферного ОЗУ период смены его адресов будет 1,6 мкс (при 8-разрядной организации — 0,8 мкс). Такие характеристики обеспечивают многие микросхемы памяти.

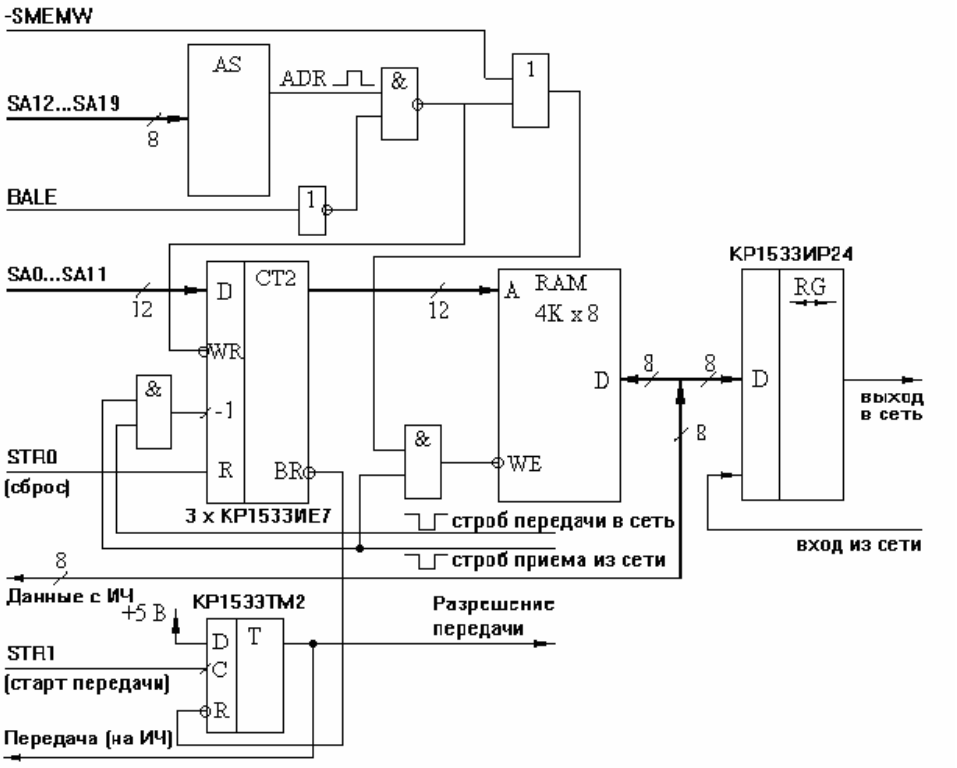


Рис. 2.56. Функциональная схема включения буферного ОЗУ в контроллере локальной сети

Пример функциональной схемы включения буферного ОЗУ в контроллере сети показан на рис. 2.56. Часть интерфейсной части УС для наглядности приведена здесь подробнее. Селектор адреса AS вырабатывает сигнал ADR в случае обращения компьютера в заданную 4К-байтную зону адресов памяти. В случае такого обращения счетчик переходит в режим параллельной записи входных данных и передает 12 младших разрядов адреса на адресные входы буферного ОЗУ. По стробу обмена -SMEMW данные записываются в память. Аналогично обрабатывается строб чтения из памяти -SMEMR, управляющий буфером данных для ОЗУ (на схеме не показано).

Передаваемый пакет данных формируется в ОЗУ, начиная с нулевого адреса. После окончания записи в ОЗУ дается команда старта передачи, перебрасывающая триггер разрешения передачи. В случае возможности передачи (сеть свободна) информация из ОЗУ через регистр сдвига выдается в сеть. При этом каждый строб передачи в сеть уменьшает состояние счетчика на единицу. После выдачи в сеть всего сформированного пакета вырабатывается сигнал переноса счетчика, перебрасывающий триггер разрешения передачи в исходное состояние.

При приеме пакета из сети каждый строб приема из сети также уменьшает состояние счетчика на единицу. Поэтому хотя в сеть пакет поступает как бы задом наперед (байт, записанный последним, выдается первым, что, кстати, надо учитывать при записи стартовой комбинации), но в ОЗУ приемника он располагается так же, как и в ОЗУ передатчика (в порядке возрастания адресов ОЗУ). Это иллюстрируется рис. 2.57. Таким образом производится автоматический учет длины передаваемого пакета, и все байты, записанные в ОЗУ компьютером, передаются в сеть.

Схема рис. 2.56 восьмиразрядная, что позволяет несколько упростить аппаратуру. Но для достижения большего быстродействия можно перейти на 16-разрядную схему. И в

том, и в другом случае компьютеру не требуется дополнительного времени для перекачки содержимого системного ОЗУ в буферное ОЗУ контроллера и наоборот. В то же время следует отметить такой недостаток предложенного подхода, как невозможность приема пакета из сети во время обмена буферного ОЗУ с компьютером, что ограничивает применение приведенной схемы. Устранение этого недостатка тре бует существенного усложнения аппаратуры и здесь не рассматривается.

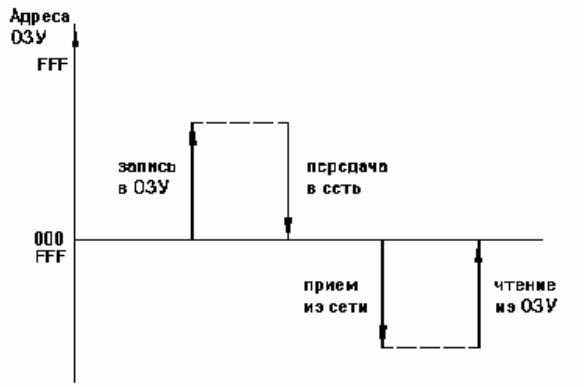


Рис. 2.57. Порядок обмена с буферным ОЗУ контроллера локальной сети

2.2. Разработка программного обеспечения устройств сопряжения для ISA

2.2.1. Особенности проектирования программного обеспечения для устройств сопряжения.

Специфика программирования аппаратуры и, в частности, устройств сопряжения для компьютера, заключается в повышенных требованиях к быстродействию программного обеспечения и, в ряде случаев, в необходимости получения программы минимального размера. Для этого приходится выбирать подходящие языки программирования, а также использовать специальные методы и алгоритмы.

Наиболее очевидный выбор языка программирования — ассемблер. Действительно, на ассемблере можно написать любую программу, и, при условии минимальной грамотности программиста, она получится самой быстрой, а код — самым коротким (причем это относится ко всем областям программирования). Однако трудоемкость программирования на ассемблере и требования к квалификации программиста несравненно выше, чем для языков высокого уровня.

Кроме того, редко программа управления устройством нужна сама по себе: как правило, она является частью системы (с интерфейсом пользователя, математикой и другими блоками, писать которые на ассемблере, мягко говоря, нецелесообразно). И хотя вопросы сопряжения написанных на разных языках программ во многих случаях решены, это еще одна возможность потратить массу усилий.

В то же время большинство современных реализаций языков программирования высокого уровня (Бейсик, Си, Паскаль и другие) имеют более или менее эффективные средства программирования устройств сопряжения. Рассматривая пригодность того или иного языка, следует проанализировать с точки зрения полноты, оптимальности и удобства программирования такие его возможности, как:

— программный доступ к устройствам ввода/вывода и к памяти,
— обработка прерываний,
— битовые логические операции,
— управление системным таймером

и, возможно, какие-то другие в зависимости от конкретной задачи.

Написанные на языках высокого уровня программы как правило оказываются не намного медленнее и длиннее аналогичных программ, написанных на ассемблере. Многие эксперты считают, что наиболее удобен для программирования аппаратуры язык Си. Многие с ними не согласны. Во многом это дело вкуса и привычки программиста. Именно этим, по-видимому, и определен выбор авторами книги языка Си для реализации примеров программ.

Есть, тем не менее, достаточно узкие области, в которых ассемблер оказывается действительно самым эффективным, а иногда и единственным средством программирования. Эти области лежат на самых крайних пределах требований к размерам и быстродействию — там, где важен каждый байт (резидентные программы; программы, записываемые в ПЗУ) и каждая микросекунда (драйверы некоторых быстрых устройств реального времени).

Наряду с использованием стандартных методов и правил программирования, при программировании аппаратуры приходится учитывать особенности конкретной задачи и применять специфические приемы, часто идущие вразрез с принципами "хорошего стиля". Например, часто неэффективным оказывается принцип модульного программирования, так как сами процедуры могут быть очень короткими и быстрыми, а на вызов функций и передачу параметров может тратиться половина времени.

Как правило, собственно взаимодействие с устройством связано с подачей на него и приемом от него определенных сигналов в определенном порядке и представляет собой последовательность операций ввода/вывода, обильно усыпанную битовыми логическими операциями над принимаемыми и передаваемыми данными. Большинство предлагаемых в данной книге примеров программирования устройств сопряжения написано в едином стиле, опирающемся на такие правила как:

— устройства сопряжения выполняют определенные функции, каждая из которых реализуется в виде отдельного драйвера; на более низком уровне (управление отдельными разрядами отдельных регистров) разделения на программные модули не производится;

— большинство устройств имеют регистры управления или состояния, в которых каждый бит или группа битов соответствуют определенным режимам работы устройства; маски этих битов (позиции в байте или слове) определяются перед драйверами и используются в битовых операциях для установки или проверки;

— в качестве глобальной переменной определяется первый ("базовый") адрес устройства в адресном пространстве, а его остальные адреса вычисляются добавлением к базовому адресу смещения.

Следует помнить, что приведенные здесь программы служат только для иллюстрации особенностей программирования устройств, поэтому не следует механически переносить куски этих программ в свои системы — работоспособность таких систем авторами не гарантируется.

2.2.2. Программирование универсального контроллера параллельного обмена

Описанный в п.2.1.9 универсальный контроллер параллельного обмена (УКПО) содержит 7 программно-доступных байтовых двунаправленных портов и регистр управляющего слова. Таким образом, на нижнем уровне программирования УКПО необходимо реализовать три функции: запись управляющего слова, запись данных в

любой порт и чтение данных из любого порта. На языке Си они выглядят так:

```
// *** Функция записи управляющего слова
// CW_ADDR — адрес регистра управляющего слова
// cw — байт управляющего слова
  outportb (CW_ADDR, cw);

// *** Функция записи данных в порт
// BASE_ADDR — базовый адрес УКПО (адрес порта 0)
// port — номер порта (0...6)
// data — байт данных
  outportb (BASE_ADDR + port, data);

// *** Функция чтения данных из порта
// BASE_ADDR — базовый адрес УКПО (адрес порта 0)
// port — номер порта (0...6)
// data — байт данных
  data = inportb (BASE_ADDR + port);
```

Напишем для примера простейшую программу тестирования УКПО. Будем последовательно записывать во все порты код 55H и контролировать правильность записи. При этом все порты, кроме тестируемого, будем переводить в режим чтения.

```
// *** Программа тестирования УКПО ***

#include <STDIO.H>
#include <DOS.H>

#define NPORT 7                         // Число портов УКПО

// Глобальные переменные
unsigned BASE_ADDR;                     // Базовый адрес УКПО
unsigned CW_ADDR;             // Адрес регистра управляющего слова

void main (void)
{
unsigned i;
unsigned char data = 0x55;

  for (i=0;i<NPORT;i++)
    {
          outportb (CW_ADDR, 1 << i);  // Установка конфигурации:
                                                                  // i—ый порт на вывод,
                                                                  // остальные на ввод
          outportb (BASE + i, data);                  // Запись данных
          if ( inportb (BASE + i) != data )
            printf ("\n\tПорт %d неисправен");
          else
            printf ("\n\tПорт %d исправен");
    }
}
```

```
//*** Конец программы
```

Как уже отмечалось, каждому порту УКПО соответствует один бит управляющего слова. Единица в этом бите устанавливает порт в режим передачи данных, а ноль — в режим приема. Естественно, что когда порт находится в режиме передачи, ничто не мешает прочитать его содержимое, контролируя таким образом правильность вывода. Именно это свойство УКПО использовано в программе его тестирования. Поэтому, хотя мы и будем называть состояния портов "ввод" и "вывод", будем иметь ввиду, что "вывод" на самом деле является "вводом/выводом". В нашем примере для последовательной установки портов в режим вывода использовалась операция сдвига единицы на число разрядов, равное номеру порта. Очевидно, что при этом данный порт устанавливается на вывод, а все остальные — на ввод.

Используя описанные выше функции программирования УКПО, можно реализовать самые разные режимы его работы и алгоритмы взаимодействия с внешними устройствами.

Например, на основе УКПО может быть построен простейший тактовый генератор. Для этого будет использован один из битов порта 0 УКПО. Ниже приведен пример программы тактового генератора, работающего с частотой 1 кГц.

```
// *** Программа тактового генератора на 1 кГц на базе УКПО.
// *** Выход генератора — бит 0 порта 0.

#include <DOS.H>
#include <CONIO.H>

#define MASK_BIT 1   // Маска бита 0

// Глобальные переменные
unsigned BASE_ADDR;                  // Базовый адрес УКПО
unsigned CW_ADDR;             // Адрес регистра управляющего слова

void main (void)
{
unsigned char data = 0;             // Начальное состояние — 0

  outportb (CW_ADDR, 1);        // Установка конфигурации:
                                                  // порт 0 на вывод,
                                                  // остальные на ввод
  while (! kbhit())                        // Пока не нажата любая клавиша
    {
          outportb (BASE, data); // Запись данных
          data ^= MASK_BIT;     // Инверсия бита
          delay (1);                              // Задержка на 1мс
    }
}
//*** Конец программы
```

В этой программе изменение состояния выхода УКПО производится инверсией бита с помощью операции "ИСКЛЮЧАЮЩЕЕ ИЛИ" с позиционной маской бита. Для выдержки временного интервала использована системная функция delay языка Си, в качестве аргумента принимающая время в миллисекундах. Очевидно, что максимальная

частота такого тактового генератора как раз и равна 1 кГц, чего в большинстве случаев недостаточно. Кроме того, точность установки частоты снижается из-за суммирования задержек выполнения остальных команд цикла.

Для повышения частоты тактового генератора можно использовать другой способ формирования программной задержки — цикл с заранее определенным временем исполнения. Например, если известно, что операция

```
for (i=0;i<1;i++)                    // Пустой цикл, выполняемый
  ;                                  // один раз
```

выполняется N микросекунд, то, устанавливая предел выполнения цикла, можно получить любую задержку в диапазоне от N до 65536·N с шагом N. Однако такой способ требует предварительного определения быстродействия компьютера, что обычно нелегко сделать с достаточной точностью.

Наконец, для выдержки временного интервала можно использовать системный таймер компьютера. При этом программные прерывания 08H и 1CH позволяют устанавливать интервалы, не меньшие чем 55 мс (частота примерно 18.2 Гц). Для получения меньших временных интервалов необходимо самому запрограммировать таймер.

С помощью УКПО просто реализуется асинхронный протокол взаимодействия с "медленными" внешними устройствами. Основа такого протокола — цикл типа "команда—ожидание реакции—действие". В качестве примера рассмотрим сопряжение БИС АЦП 1113ПВ1 с компьютером с помощью УКПО. Условное обозначение БИС, ее подключение к портам УКПО и временные диаграммы работы показаны на рис. 2.58. Видно, что процедура чтения кода с АЦП реализуется последовательностью действий: запуск — ожидание готовности — чтение данных.

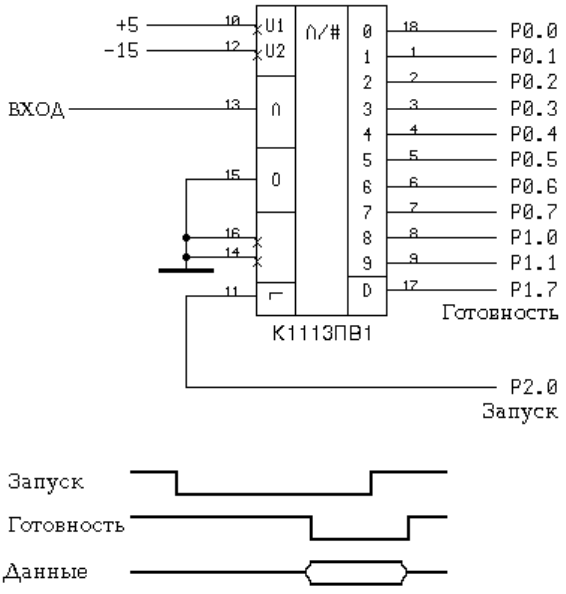


Рис. 2.58. Подключение БИС АЦП 1113ПВ1

```
// *** Программа чтения данных из БИС АЦП 1113ПВ1 ***

#include <DOS.H>
#include <STDIO.H>

#define START 1                 // Маска бита запуска (бит 0 порта 2)
#define READY 0x80   // Маска бита готовности (бит 7 порта 1)
#define DH 3                    // Маска 8-го и 9-го битов данных АЦП
                                        // (биты 0 и 1 порта 1)

// Глобальные переменные
unsigned BASE_ADDR;                     // Базовый адрес УКПО
unsigned CW_ADDR;               // Адрес регистра управляющего слова

void main (void)
{
unsigned data_ADC = 0;                  // Данные с АЦП

  outportb (CW_ADDR, 4);        // Установка конфигурации:
                                                  // порт 2 на вывод,
                                                  // остальные на ввод
  outportb (BASE+2, START^1)            // Запуск АЦП — запись 0
  while ( ( (data_ADC = inportb (BASE+1)) & READY ) != 0 )
    ;                                             // Опрос готовности
  data_ADC = (data_ADC & DH)<<8 + inportb (BASE);// Чтение
                                                  // младшего байта данных
  printf ("\nПрочитан код — %u", data_ADC);
}
//*** Конец программы
```

В следующем примере рассмотрим реализацию протоколов обмена с "быстрым" устройством — модулем ОЗУ емкостью 16 КБайт с байтовой организацией. Подключим модуль ОЗУ к портам УКПО следующим образом:

P0.0 ... P0.7 — младший байт шины адреса: A0 ... A7,
P1.0 ... P1.5 — старшие линии шины адреса: A8...A13,
P1.6 — сигнал разрешения записи -WE,
P1.7 — сигнал разрешения выхода -OE,
P2.0 ... P2.7 — шина данных: D0 ... D7.

Ниже приведены драйверы модуля ОЗУ, реализующие процедуры записи и чтения байта данных по произвольному адресу. Перед вызовом драйверов должно быть предварительно установлено управляющее слово УКПО — 03H (порты 0 и 1 — на вывод, порт 2 — на ввод).

```
// *** Драйверы модуля ОЗУ ***

#define WE 0x40         // Маска бита WE (бит 6 порта 1)
#define OE 0x80         // Маска бита OE (бит 7 порта 1)
#define AH 0x3F         // Маска линий A8...A13 шины адреса
                                // (биты 0...5 порта 1)

// Глобальные переменные
```

```
unsigned BASE_ADDR;                    // Базовый адрес УКПО
unsigned CW_ADDR;              // Адрес регистра управляющего слова

// Прототипы функций
void Write_RAM (unsigned addr, unsigned char data);
unsigned char Read_RAM (unsigned addr);

// Функция записи байта данных
void Write_RAM (unsigned addr, unsigned char data)
{
  outportb (CW_ADDR, 7);                       // Порты 0...2 — на вывод
  outportb (BASE_ADDR, addr);                  // Младший байт адреса
  outportb (BASE_ADDR+1, (addr>>8) & AH | OE & (~WE));
                                                          // Старшие линии адреса,
                                                          // OE=1, WE=0
  outportb (BASE_ADDR+2, data);        // Установка данных
  outportb (BASE_ADDR+1, (addr>>8) & AH | OE | WE)); // WE=1
  outportb (CW_ADDR, 3);                       // Порты 0,1 — на вывод,
                                                          // порт 2 — на ввод
}



// Функция чтения байта данных
unsigned char Read_RAM (unsigned addr)
{
unsigned char data;

  outportb (BASE_ADDR, addr);                  // Младший байт адреса
  outportb (BASE_ADDR+1, (addr>>8) & AH & (~OE) | WE);
                                                          // Старшие линии адреса,
                                                          // OE=0, WE=1
  data = inportb (BASE_ADDR+2);       // Чтение данных
  outportb (BASE_ADDR+1, (addr>>8) & AH | OE | WE)); // OE=1
  return (data);
}
//*** Конец драйверов
```

Важно обратить внимание на управление отдельными битами порта 1. Для этого использованы логические операции с прямыми и инверсными масками соответствующих битов. Кроме того, следует отметить, что подключение к порту 2 двунаправленной шины данных требует изменения направления передачи по этому порту, то есть перепрограммирования управляющего слова (в отличие от случаев подключения однонаправленных линий, когда управляющее слово устанавливается в начале прогаммы и больше не модифицируется).

Другой задачей, где в процессе работы требуется изменение направления передачи порта, является использование части его линий для ввода, а другой части — для вывода информации (как уже неоднократно говорилось, направление передачи информации можно установить только для всего порта целиком, а не для отдельных его битов). Рассмотрим сопряжение с компьютером простейшего устройства, состоящего из четырех ключей и четырех лампочек и предположим, что ввиду большой загруженности УКПО

другими задачами для подключения нашего устройства выделен только один порт 5 (рис. 2.59). Драйверы, приведенные ниже, реализуют две функции — чтение состояния ключей и "поджигание" лампочек.

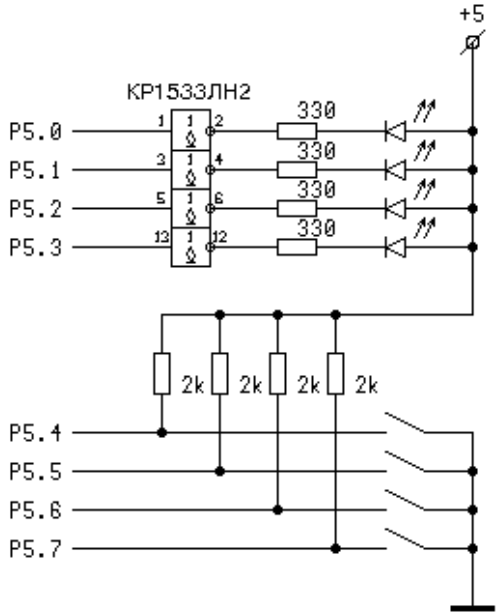


Рис. 2.59. Подключение простейшего устройства из ключей и лампочек

```
// *** Драйверы устройства "Набор лампочек и кнопочек" ***

#define LAMP 0x0F     // Маска битов светодиодов
                                  // (биты 0...3 порта 5)
#define P5 0x20           // Маска бита порта 5 в управляющем слове

// Глобальные переменные
unsigned BASE_ADDR;                  // Базовый адрес УКПО
unsigned CW_ADDR;             // Адрес регистра управляющего слова
unsigned char CW;                    // Управляющее слово

// Прототипы функций
unsigned char Get_Switch (void);
void Set_Lamp (unsigned char data);

// Функция чтения состояния ключей
unsigned char Get_Switch (void)
// В возвращаемом байте четыре старших бита (4...7)
// соответствуют четырем ключам (0 — замкнут, 1 — разомкнут).
{
unsigned char data;

  outportb (CW_ADDR, CW & (~P5));  // Порт 5 — на ввод
  data = inportb (BASE_ADDR+5);       // Чтение состояния ключей
  outportb (CW_ADDR, CW | P5);        // Порт 5 — на вывод
```

```
  return (data);
}

// Функция "поджигания" лампочек
void Set_Lamp (unsigned char data)
// Четыре младших бита из data соответствуют четырем
// лампочкам (1 — горит, 0 — погашен).
{
  outportb (BASE_ADDR+5, (data & LAMP) | Get_Switch());
}
//*** Конец драйверов
```

Понятно, что если для "поджигания" лампочек просто вывести соответствующую тетраду в порт 5, это может привести к конфликту на четырех старших линиях между выводимой информацией и состоянием ключей. Поэтому в старшую тетраду надо записывать те данные, которые установлены ключами. Для этого приходится считывать их состояние при каждой записи в порт 5. Для считывания состояния ключей необходимо переводить порт 5 на ввод, а затем, после чтения — опять на вывод. Такое кратковременное отключение порта от лампочек (время зависит от быстродействия компьютера, но во всяком случае не превосходит нескольких десятков микросекунд) не будет заметно на глаз.

2.2.3. Программирование логического анализатора

При написании драйверов логического анализатора (см. п.2.1.10) необходимо реализовать три основные процедуры:

— установка параметров и запуск регистрации;
— проверка окончания регистрации;
— чтение массива данных.

```
// *** Драйверы логического анализатора ***

#define READY 0x01   // Маска бита готовности (бит 0)

#define FREQ 0x07              // Маска битов кода частоты
                                    // регистрации (биты 0...2)
#define SYNC 0x38              // Маска битов кода синхролинии
                                    // (биты 3...5)
#define POLAR 0x40   // Маска бита полярности
                                    // синхроперехода (бит 6)

#define LENGTH 4096// Глубина регистрации — 4096 кадров

// Глобальные переменные
unsigned CW_ADDR;              // Адрес регистра управляющего слова
unsigned PRE_REG_ADDR;    // Адрес регистра глубины
                                                  // предпусковой регистрации
unsigned R0,R1,R2,R3;          // Адреса чтения байтов 0...3 данных
unsigned READY_ADDR;       // Адрес флага готовности
```

```
// Структура параметров регистрации
typedef struct LA_Param {
        unsigned char freq;     // Код частоты (0...7)
        unsigned char sync;     // Номер синхролинии (0...7)
        unsigned char polar;    // Код полярности синхроперехода
                                //      0 — положительный,
                                //      1 — отрицательный.
        unsigned pre_reg;       // Глубина предпусковой
                                // регистрации (0 — (LENGTH-1))
        };

// Прототипы функций
int Init_LA (LA_PARAM *la);
unsigned Check_Ready (void);
void Read_Data (unsigned long *data);

// Функция установки параметров и запуска регистрации
int Init_LA (LA_PARAM *la)
// Возвращает: 0 — запуск произведен,
//             -1 — ошибка параметров регистрации
{
unsigned char cw. pr;

// Проверка правильности входных параметров
  if (la->freq > 7 || la->sync > 7 || la->polar > 1 ||
     la->pre_reg >= LENGTH)
    return (-1);                        // Выход по ошибке
// Формирование управляющего слова
// и кода предпусковой регистрации
  cw = (la->freq & FREQ) + ((la->sync << 3) & SYNC)
      + ((la->polar << 6) & POLAR);
  pw = la->pre_reg >> 4;                // Точность — 16 кадров

  outportb (CW_ADDR, cw);               // Запись управляющего слова
  outportb (PRE_REG_ADDR, pr);          // Запись кода предпусковой
                                        // регистрации и запуск анализатора
  return (0);                           // Выход по нормальному запуску
}

// Функция проверки окончания регистрации
unsigned Check_Ready (void)
// Опрашивает флаг готовности и возвращает:
// 0 — готов, иначе — не готов
{
  return (inportb (READY_ADDR) & READY);
}

// Функция чтения данных из буферного ОЗУ логического
// анализатора в память компьютера
void Read_Data (unsigned long *data)
```

```
// data — указатель на массив, в который будут переписаны
//        данные из буферного ОЗУ.
{
unsigned i;

  for (i=0;i<LENGTH;i++)        // Повторяем чтение 4096 раз
    data[i] = (unsigned long) inportb (R0) +
            (unsigned long) inportb (R1) << 8 +
                    (unsigned long) inportb (R2) << 16 +
                    (unsigned long) inportb (R3) << 24;

}
//*** Конец драйверов
```

С использованием этих драйвером в качестве примера напишем простейшую программу для подсчета соотношения длительности единицы и нуля по всем из подключенных ко входам логического анализатора линий за время 100 мкс после прихода первой единицы по линии 3. Для получения максимальной точности будем проводить регистрацию на предельной частоте 10 МГц.

```
// Программа для подсчета соотношения единиц и нулей

#include <STDIO.H>

void main (void)
{
struct LA_Param la;
unsigned long data[LENGTH];
unsigned num0[32], num1[32];
float ratio[32];
unsigned i, j;

// Инициализация параметров регистрации
  la.freq = 0;      // Код частоты 10 МГц (период 100 нс)
  la.sync = 3;      // Номер синхролинии
  la.polar = 0;     // Запуск по положительному перепаду
  la.pre_reg = 0;// Предпусковая регистрация не нужна

// Запуск регистрации
  if (Init_LA (&la) == -1)
    {
      printf ("\nОшибка.");
            exit (-1);
    }

// Проверка окончания регистрации
  while (Check_Ready() != 0)
    ;
```

```
// Чтение данных
  Read_Data (&data);

// Определение соотношений единиц и нулей по всем линиям
// 100мкс — это 1000 кадров регистрации на частоте 10 МГц
  for (j=0;i<32;j++)                  // Обнуление счетчиков
    {                                          // единиц и нулей
          num0[j] = 0;                  // по всем линиям

          num1[j] = 0;
    }
  for (i=0;i<1000;i++)                // Подсчет числа нулей и единиц
    for (j=0;j<32;j++)                // по всем линиям, (1L << j) -
      if ( (data[i] & (1L << j)) == 0 )        // маска j-го бита
        num0[j]++;
      else
            num1[j]++;
  for (j=0;j<32;j++)                  // Вывод на экран соотношений
    {
      printf ("\nЛиния №%d. ", j);
      if (num0[i] != 0)               // Был ли ноль вообще ?
        printf (Отношение 1/0 - %f.", num1[j]/num0[j]);
          else                                 // Не было
        printf ("Нуля не было.");
    }
}
// Конец программы
```

В предложенных драйверах окончание регистрации определялось опросом флага готовности логического анализатора. Однако во многих случаях удобнее использовать аппаратное прерывание, которое анализатор вырабатывает по окончании регистрации (см. п. 2.1.10). Например, если синхропереход приходит редко, можно построить систему таким образом, чтобы после запуска регистрации программа переходила на выполнение другой задачи (например, обработка ранее полученных данных), а по прерыванию от логического анализатора переходила на его обработку. В этом случае необходимо написать программу обработки прерывания, в которую включить все необходимые действия. Драйвер Check_Ready становится не нужен.

```
// *** Программа обработки прерывания от логического анализатора
void interrupt LA_handler (unsigned long *data)
{
  disable();                  // Запрет прерываний на время обработки
  . . .                       // Какие-то действия
  Read_Data (data);           // Чтение данных из буфера
  . . .                       // Какие-то действия
  enable();                          // Разрешение прерываний
}
// Конец программы
```

В тело основной программы следует включить установку вектора прерывания от логического анализатора:

```
setvect (INT, LA_handler);          // INT — номер прерывания
```

2.2.4. К вопросу о программировании сетевого контроллера

Описанные в п. 2.1.12 варианты реализации узлов контроллера локальной сети требуют пересылки информационных пакетов из ОЗУ компьютера в буферное ОЗУ контроллера и наоборот. При этом скорость пересылки должна быть максимальной для повышения информационного быстродействия сети. Написанная "в лоб" даже на ассемблере (цикл команд MOV), эта процедура оказывается слишком медленной. Однако не следует спешить использовать прямой доступ к памяти. В системе команд имеется инструкция для пересылки массива из одного места памяти в другое — MOVS. Например, если буфер контроллера сети занимает адреса B0000H — B0FFFH, то процедура записи в него пакета длиной 4Кбайта из адресов 80000H—80FFFH выглядит следующим образом:

```
MOV   CX, 1000H;          длина пакета
MOV   SI, 80000H;         начало источника
MOV   DI, B0000H;         начало приемника
MOVS;                     пересылка
```

В библиотеке языка Си фирмы Borland имеется несколько функций, реализующих быструю пересылку массивов памяти. Эти функции построены на базе инструкции MOVS. Так, для решения той же задачи можно использовать функцию movedata:

```
movedata (0x8000, 0, 0xB000, 0, 4096);
```

где первый и второй параметры — сегмент и смещение источника, третий и четвертый — сегмент и смещение приемника, пятый — длина массива.

2.3. Особенности отладки устройств сопряжения для ISA

Как уже отмечалось, особенностью разработки УС является опасность выхода из строя компьютера, к которому подключается изготовленное УС. В первую очередь это, конечно, относится к УС, ориентированным на ISA, то есть подключаемым к "внутренностям" компьютера. Большое число сигналов интерфейса ISA, подключение к нему системных устройств компьютера, сложность алгоритмов взаимодействия по ISA, использование внутреннего источника питания — все это приводит к тому, что вероятность поломки компьютера (и, как следствие этого, потеря интереса у разработчика к проетированию любых УС) довольно велика. Причем эта опасность исходит не только от вновь созданных УС, но и от УС, изготовленных недостаточно надежным производителем. Что касается использования двух других внешних интерфейсов компьютера (Centronics и RS-232C), то УС, подключаемые к ним, хотя и могут вывести из строя соответствующие порты, но очень редко ломают весь компьютер целиком. Поэтому для них рассматриваемая проблема стоит не столь остро.

Какие же методы могут быть предложены для предотвращения этой неприятной ситуации? Прежде всего стоит отметить, что полной гарантии от нарушений в работе компьютера, вызванных подключением к нему любой дополнительной платы (а отнюдь не только УС), не может дать ни один метод. Ни одна система контроля или отладки просто не в силах смоделировать все возможные сочетания комбинаций и последовательностей

комбинаций сигналов интерфейса. Этот процесс занял бы огромное время.

Однако нарушения нарушениям рознь. Наиболее важно выявить те недостатки УС, которые ведут к необратимым поломкам компьютера, например, к выходу из строя драйверов системных узлов, к поломке источника питания и т.д. К другой группе дефектов УС относятся нарушения стандарта интерфейса, устойчиво проявляющиеся в циклах обращения к данному УС и мешающие нормальной работе компьютера, но не ломающие его окончательно. Задачу их выявления усложняет то, что некоторые из них наблюдаются только в циклах обмена, осуществляемых в реальном масштабе времени, в том темпе, который задается самим компьютером. Все эти дефекты УС могут быть достаточно достоверно выявлены, локализованы и устранены с помощью систем отладки, которые мы рассмотрим в этом разделе.

Почему возникает необходимость создания специальных средств отладки для УС? Ведь существует же большое число приборов, применяемых для контроля и отладки цифровой и цифроаналоговой техники: всевозможные генераторы, осциллографы, вольтметры и т.д. Но дело в том, что особенностью УС является шинная организация с большим количеством входных, выходных и двунаправленных сигналов, которые должны одновременно формироваться и одновременно контролироваться. Применение стандартных приборов в этом случае превратится в мучение для инженера или техника, занимающегося отладкой, и потребует неоправданно большого времени. Поэтому изготавливаются специальные многоразрядные формирователи входных воздействий и регистраторы ответных реакций, существенно упрощающие этот процесс. Естественно, гораздо большую эффективность имеют системы, в которых предусмотрена та или иная степень автоматизации процесса отладки. При этом часть трудоемких операций можно возложить на персональный компьютер, который часто служит основой такой системы.

2.3.1. Комплекс средств статической отладки

Метод статической отладки цифровых устройств, как следует из его названия, позволяет контролировать работу этих устройств или их отдельных узлов в статическом режиме, то есть в режиме неизменных входных и выходных сигналов. Этот метод основан на том положении, что большинство цифровых устройств может находиться в каждом из своих состояний бесконечно большое время. Он позволяет выявлять по различным оценкам до 80% неисправностей. Оставшиеся 20% приходятся на долю неисправностей, связанных с динамикой, с быстрыми изменениями сигналов, с емкостными и индуктивными эффектами, с гонками сигналов, с высокочастотными помехами и наводками, с ограничением быстродействия отдельных элементов и узлов и т.д.

Существуют узлы, которые в принципе не проверяются методом статической отладки, например, блоки динамического ОЗУ, одновибраторы (ждущие мультивибраторы), генераторы, времязадающие RC-цепочки, некоторые микросхемы с заданным нижним пределом тактовой частоты (например, процессоры, внутри которых используются элементы динамической памяти). Однако в системе отладки часто предусматривают так называемый квазидинамический режим, позволяющий иногда проверить и их.

Какие же неисправности выявляются рассматриваемым методом? Прежде всего это внутренние неисправности микросхем отлаживаемого устройства, приводящие к нарушениям в логике их работы, к невыполнению таблицы истинности. Например, выход логического элемента не реагирует на один или несколько входных сигналов, или вход микросхемы закорачивает на землю или питание подводимый к нему сигнал. Кроме того метод статической отладки позволяет обнаружить дефекты печатного монтажа платы: непропаи выводов элементов, закоротки соседних проводников, обрывы проводников, ошибки при изготовлении фотошаблона платы.

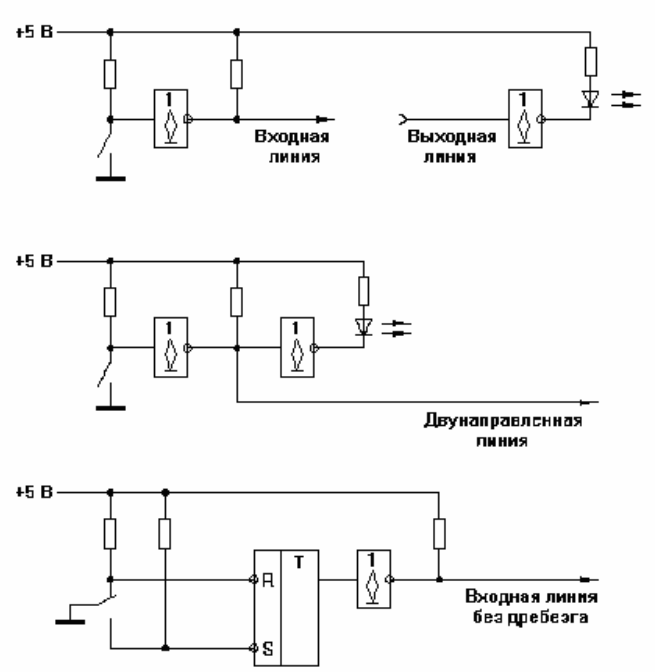


Рис. 2.60. Простейшая реализация метода статической отладки

Простейший путь реализации метода статической отладки — использование обычных тумблеров для задания входных и двунаправленных сигналов, а также светодиодов для индикации состояния выходных и двунаправленных сигналов (рис. 2.60). Однако при этом на оператора ложится огромная работа по переключению тумблеров и контролю светодиодов. К тому же он просто не в силах проверить все возможные ситуации, например, перебрать все состояния многоразрядной шины. В то же время в ряде случаев, например, при большом разнообразии довольно простых отлаживаемых цифровых устройств этот путь оказывается вполне приемлемым.

Особенность УС как объекта отладки — однотипность набора входных, выходных и двунаправленных сигналов со стороны разъема интерфейса компьютера и однотипность последовательностей этих сигналов в соответствии с протоколом обмена по интерфейсу. Это позволяет обеспечить довольно высокую степень автоматизации процесса отладки и облегчить труд оператора, на долю которого остается только выбор режима работы и анализ результатов контроля.

При отладке УС надо добиться, чтобы, во-первых, оно правильно выполняло свои функции (то есть при обращении по его адресам вырабатывало нужные внутренние стробы обмена, правильно принимало и записывало информацию из магистрали и выдавало информацию на магистраль). Во-вторых, УС не должно нарушать работу магистрали, то есть оно не должно выдавать на магистраль ненужных сигналов в ненужное время, не должно закорачивать магистральные сигналы между собой, на общую шину и шины питания. Причинами этого могут, например, быть неправильная работа селектора адреса, неисправность входных, выходных и двунаправленных буферов, ошибки монтажа компонентов УС.

Если первую из указанных двух задач еще можно решить вручную (с использованием тумблеров и светодиодов), то вторую — практически невозможно, так как надо перебирать все возможные комбинации сигналов на магистрали. Но именно решение этой второй задачи очень легко автоматизировать, так как контролировать здесь надо только магистральные (внешние для УС) сигналы, а не внутренние точки УС.

Реализовать систему статической отладки на базе персонального компьютера можно очень просто, если использовать описанный в разделе 2.1.9 универсальный контроллер параллельного обмена (и, конечно же, соответствующие программные средства). В этом случае аппаратура системы включает в себя только компьютер, рассмотренный контроллер и соединительный кабель, оканчивающийся магистральным разъемом ISA, к которому и подключается исследуемая плата УС (рис 2.61). В принципе, в случае данной системы отлаживаемая плата может быть любой, главное, чтобы количество ее внешних сигналов не превышало количество внешних линий контроллера параллельного обмена. Но для каждой платы нужен свой соединительный кабель (или кабель с индивидуальными зажимами для каждой линии) и, самое главное — свое программное обеспечение.

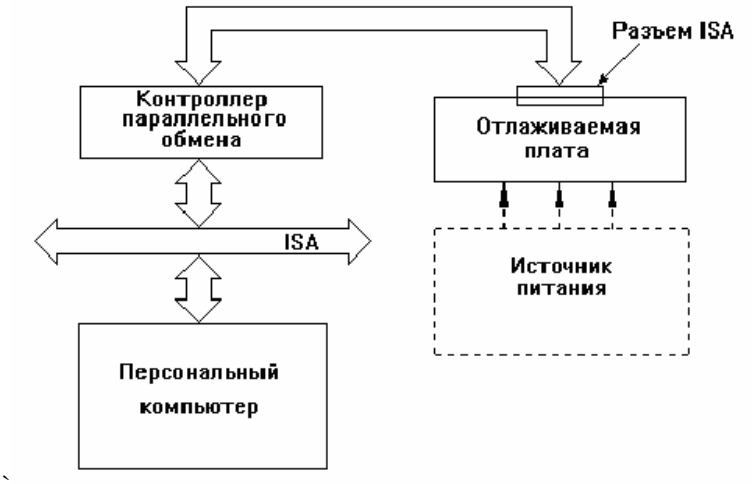


Рис.2.61. Система статической отладки на базе персонального компьютера

Для отладки же УС любого типа кабель и программы могут быть одни и те же, так как все УС должны отрабатывать один и тот же узкий набор последовательностей (машинных циклов) одних и тех же сигналов. При этом мы можем четко выделить входные, выходные, двунаправленные линии и заложить порядок формирования и контроля сигналов на них в программу, освободив от этого оператора.

Рассмотрим несколько упрощенно управление внешними линиями контроллера параллельного обмена при пошаговой эмуляции машинных циклов ISA. Но сначала выделим минимальный набор сигналов ISA, необходимый в системе отладки. Пусть речь идет только об отладке УС, работающих как устройства ввода/вывода. В этом случае достаточно использовать 10 разрядов адреса (SA0 ... SA9), 8 или 16 разрядов данных (SD0 ... SD7 или SD0 ... SD15), два строба обмена (-IOR и -IOW) и два управляющих сигнала (BALE — не обязательно и AEN). Адресные и управляющие сигналы должны быть определены как выходные, данные — как двунаправленные. Однако надо учесть, что в системе отладки обязательно надо предусмотреть контроль правильности информации, выводимой на выходные линии. Все это легко позволяет сделать контроллер параллельного обмена.

Выделим шаги (или точки останова), которые будут использованы при пошаговой отладке (рис. 2.62). Принцип выделения здесь очень простой: каждый новый шаг отличается от предыдущего изменением хотя бы одного сигнала. При этом если задержка между сигналами определяется только временем распространения сигналов по магистрали или только временем срабатывания логических элементов (например, между -IOR и SD), то мы считаем их одновременными, так как при статической отладке обнаружить такой

сдвиг все равно невозможно.

На нулевом шаге (исходное состояние) мы должны проверить отключение платы от магистрали. Первый шаг, в принципе, не обязателен, так как обычно в УС на нем ничего не происходит. На втором шаге проверяется работа селектора адреса. На третьем — контролируется реакция УС на сигнал BALE (он используется довольно редко). Шаги 4 ... 6 (для цикла записи) или четвертый шаг (для цикла чтения) — это проверка выработки внутренних стробов обмена УС и функционирования буферов данных.

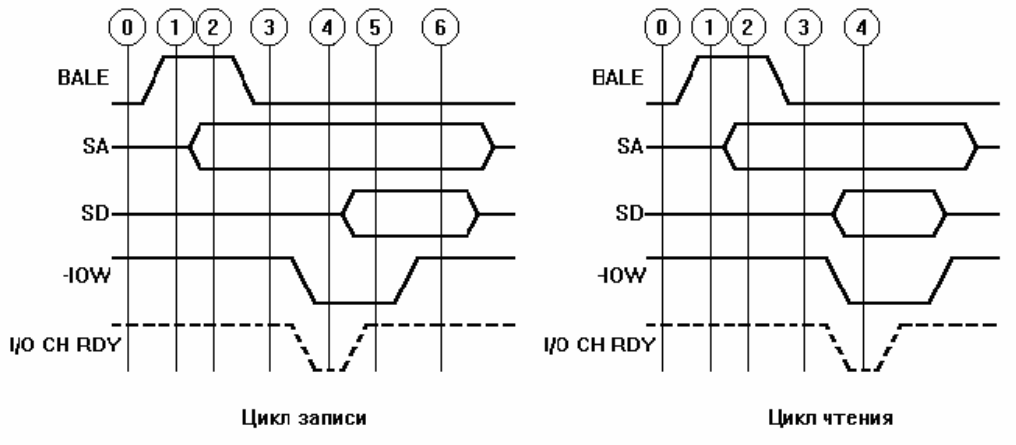


Рис. 2.62. Шаги (точки останова) в циклах записи и чтения ISA

Что касается сигнала I/O CH RDY, то его при статической отладке проконтролировать совершенно невозможно, так как его длительность по стандарту не превышает 15,6 мкс, а в худшем случае может быть равна даже одному периоду SYSCLK. Единственно, что можно здесь проверить — это не остается ли он снятым (нулевым) слишком долго.

Последовательности сигналов (рис. 2.62) могут вырабатываться компьютером не только с остановками на каждом шаге, но и без остановок (с максимально возможной для обмена компьютера с используемым контроллером скоростью). Это, конечно же, не будет истинным режимом реального времени, но быстрое периодическое повторение этих циклов облегчает отладку некоторых узлов с помощью осциллографа (уже упоминавшаяся квазидинамическая отладка).

В процессе отладки необходимо контролировать не только сигналы на магистральном разъеме платы УС, что легко может осуществлять сам компьютер, но и внутренние сигналы платы, не имеющие выхода на внешний разъем. Конечно же, для этого можно использовать обычный осциллограф, этот универсальный прибор электроники, но можно пойти и по другому пути.

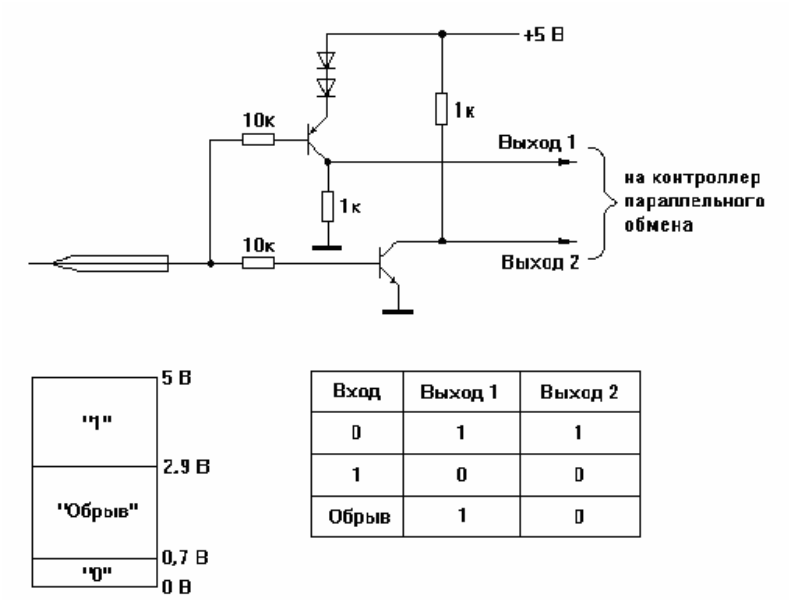
Применяемый нами котроллер параллельного обмена имеет много внешних линий, и мы задействовали для эмуляции ISA не все эти линии. Используя две из оставшихся как входные, можно довольно часто обойтись без осциллографа и существенно увеличить удобство работы оператора. Для этого надо применить схему простейшего логического щупа (пробника), показанную на рис. 2.63. Эта схема различает напряжения трех уровней: логический нуль, логическая единица и так называемый "висячий" потенциал или "обрыв" (напряжение на неподключенном входе ТТЛ-элемента). Информацию о состоянии щупа можно выводить на экран компьютера и сопровождать звуковым сигналом, поставив в соответствие, например, нулю — низкий тон, единице — высокий тон, обрыву — отсутствие звука.

А теперь несколько слов об основных сервисных функциях, которые может

выполнять компьютерная система статической отладки.

1. Перед началом отладки проводится проверка отключения отлаживаемой схемы УС от магистрали ISA. То есть УС должно быть пассивно, если к нему нет обращений. Если же обнаруживаются какие-нибудь активные линии, то на экран выводится информация об ошибке с указанием наименований ошибочных сигналов и характера неисправности.

2. Проверка установки схемы УС в исходное состояние по сигналу магистрального сброса (RESET DRV). Этот сигнал (выходной) формируется контроллером параллельного обмена и требует только два шага статической отладки: сигнал активен, сигнал пассивен. Такая проверка часто является необходимой, так как некоторые схемы УС будут неправильно работать, если их не инициализировать.



Вход	Выход 1	Выход 2
0	1	1
1	0	0
Обрыв	1	0

Рис. 2.63. Принципиальная схема логического щупа, различаемые им уровни и таблица истинности

3. Проверка, на какие адреса отвечает УС. В стандарте ISA не существует магистрального сигнала подтверждения, соответствующего ответу УС на обращенный к нему цикл (как, например, в системных магистралях Q-bus или Unibus). Сигнал I/O CH RDY используется только медленными УС, не успевающими выполнить цикл за положенное время, к тому же он очень короткий для статической отладки. Поэтому, если анализировать только канальные сигналы, мы не сможем определить, на какие адреса отвечает наше УС. Здесь возможно два пути. Простейший состоит в том, что мы проводим циклы чтения из всех возможных адресов и проверяем считанную из них информацию. Если в каких-то адресах она отлична от пассивного состояния (FFFF для 16-разрядного УС), то мы считаем, что на эти адреса данное УС отвечает. Понятно, что этот путь ненадежен, так как действительно считанная из УС информация может быть в данный момент как раз равной FFFF. Другой путь — использование упомянутого логического щупа, подключаемого к контроллеру параллельного обмена (рис. 2.63). Устанавливая его на выход селектора адреса УС и перебирая все возможные адреса, мы можем надежно определить те из них, которые принадлежат отлаживаемому УС.

4. Далее должна следовать собственно отладка, которая проводится или в статическом, пошаговом режиме (обязательный этап) или в квазидинамическом режиме. В режиме отладки мы проверяем формирование внутренних стробов обмена УС, работу буферов, правильность записываемых и читаемых данных. Функционирование

интерфейсной части УС может быть проверено практически всегда полностью. С операционной частью УС дело обстоит сложнее. Однако, если она содержит в себе только входные и выходные порты, как, например, в схеме универсального контроллера параллельного обмена, то и для нее процесс отладки проводится без проблем. Если же в операционной части есть динамические узлы (генераторы, одновибраторы, микропрограммные автоматы и т.д.), то часто может помочь квазидинамический режим с бесконечным повторением циклов обмена и осциллограф.

5. Помимо перечисленных обязательных функций система отладки может моделировать целые серии циклов обращения к УС, позволяющие легко отлаживать, например, УС с буферным ОЗУ, к которому предусмотрен последовательный доступ. Понятно, что при большом объеме такого ОЗУ произвести пошаговую запись всех его ячеек или пошаговое чтение из них очень непросто, если вообще возможно. Для реализации данной функции в системе отладки должен быть предусмотрен некий специальный язык для задания требуемых последовательностей циклов.

2.3.2. Отладка в динамическом режиме

Для выявления неисправностей УС, проявляющихся только в режиме реального времени система статической отладки не подходит. Построение аналогичной по возможностям системы динамической отладки гораздо сложнее и дороже, а эффект от ее использования зачастую оказывается невысоким. Ведь, как уже упоминалось, чисто статическая отладка выявит около 80% неисправностей, а квазидинамическая — еще около 10%. Стоит ли организовывать сложную систему для оставшихся 10% неисправностей? В то же время, в ряде случаев оказывается необходимым не заменять УС, имеющее динамические дефекты, на новое, а починить его. Тем более, что некоторые узлы принципиально работают только в режиме реального времени, и, следовательно, статическая отладка для них не подходит в принципе.

Для целей динамической отладки удобно использовать все тот же персональный компьютер, дополненный модулями динамической генерации входных сигналов отлаживаемого УС и динамической регистрации его выходных сигналов. В качестве такого модуля регистрации можно применить логический анализатор, описанный в разделе 2.1.10. Конечно, быстродействие данного анализатора не всегда достаточно для измерения малых временных задержек, но его можно заметно увеличить за счет снижения количества разрядов (рис. 2.64). Тем более, что такое быстродействие (в нашем случае 80 МГц) по всем входам регистрации, как правило, и не нужно. Достаточно иметь его по одному — двум входам.

Что касается модуля генерации последовательностей цифровых кодов, то его организация очень близка к организации модуля логического анализатора, но только направление потока данных противоположное.

Структура системы динамической отладки в данном случае будет довольно простой (рис. 2.65). Дополнительный буферный модуль (БМ) служит для усиления сигналов с целью передачи их по соединительному кабелю и формирования пришедших по кабелю сигналов. Генератор кодов задает последовательности сигналов интерфейса ISA, а логический анализатор контролирует как канальные сигналы ISA, так и сигналы с внутренних точек отлаживаемого УС. При этом контроль за правильностью функционирования может быть возложен как на оператора, анализирующего фиксируемые временные диаграммы на экране компьютера, так и на компьютер, сравнивающий полученные временные диаграммы с эталонными, хранящимися у него в памяти. Последний режим особенно удобен в том случае, когда нам нужно не чинить плату УС, а только проконтролировать ее исправность по принципу "годен — не годен", что очень полезно при выходном контроле на производстве. При этом мы можем практически

полностью автоматизировать процесс контроля, возложив на оператора только смену контролируемых плат.

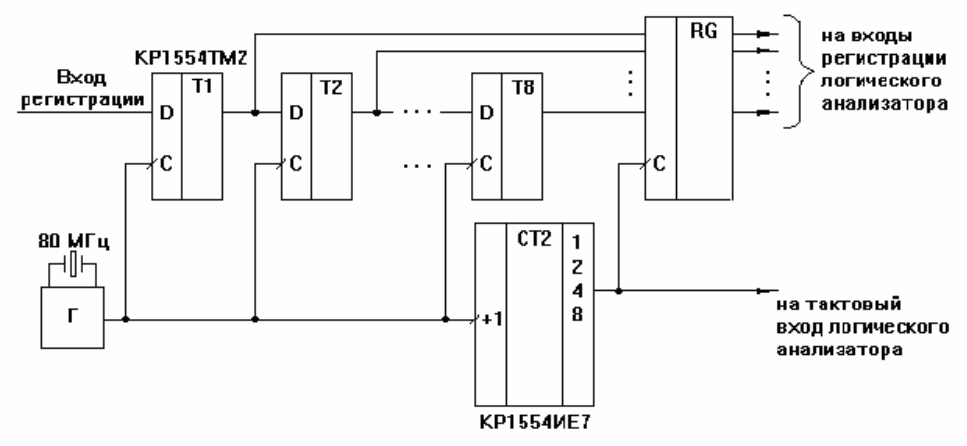


Рис. 2.64. Увеличение в 8 раз быстродействия логического анализатора за счет снижения количества разрядов

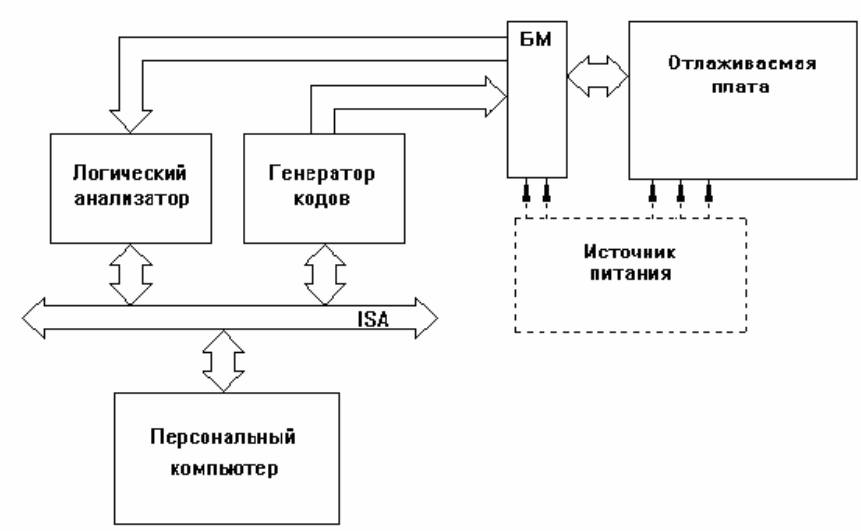


Рис. 2.65. Система динамической отладки

Таким образом, дополнив персональный компьютер несколькими УС, можно получить самые разнообразные системы контроля и отладки различных цифровых (да и аналоговых при использовании ЦАП и АЦП) устройств и систем. Причем компьютер обеспечит нам удобный интерфейс пользователя, сложные алгоритмы обработки, средства хранения результатов работы и их документирования.

ГЛАВА 3

РАЗРАБОТКА УСТРОЙСТВ СОПРЯЖЕНИЯ ДЛЯ CENTRONICS

3.1. Основные принципы проектирования аппаратуры для сопряжения с Centronics

3.1.1. Чем удобен и чем неудобен интерфейс Centronics.

Интерфейс Centronics благодаря простоте сопряжения и удобству программирования широко используется для подключения к компьютеру нестандартных внешних устройств. Однако выбор разработчиком именно этого интерфейса для связи своего устройства с компьютером должен быть осознанным и учитывать ряд уже упоминавшихся ограничений.

Во-первых, возможности реализации различных протоколов информационного обмена с устройством через параллельный порт невелики. Действительно, небольшое количество сигнальных линий интерфейса и возможности его программирования не позволяют реализовать обмен по прерываниям или прямой доступ к памяти. Практически приходится ограничиваться программно-управляемым обменом.

Кроме того, так как интерфейс Centronics является программно-управляемым, скорость информационного обмена не может быть особенно велика и оказывается напрямую связанной с быстродействием компьютера. Поэтому не имеет смысла сопряжение с через параллельный порт устройств, требующих обработки или передачи информации в реальном масштабе времени, таких как устройства ввода изображения, звуковые системы и т.д. Кроме того, зависимость скорости информационного обмена от быстродействия компьютера делает практически нереализуемыми без специальных ухищрений быстродействующие синхронные протоколы связи.

Имеется также ограничение на длину линии связи устройства, подключенного к интерфейсу Centronics. Оно должно располагаться на расстоянии не более 1.5 — 2 метров от компьютера.

Еще одной особенностью интерфейса Centronics является отсутствие на его разъеме шин питания (есть только "земля"). Это означает, что сопрягаемое устройство должно использовать внешний источник питания. Вообще говоря, на взгляд авторов, в ряде случаев это не только не является недостатком интерфейса, но скорее его достоинством. Нет искушения использовать питание от компьютера, что может привести к выходу его из строя.

В 99% компьютеров имеется только один параллельный порт, к которому должен подключаться принтер. Но и это ограничение часто не является существенным. Во-первых, многие компьютеры, ориентированные на работу с внешней аппаратурой, прекрасно обходятся без принтера. Во-вторых, имеется масса простых и дешевых устройств (коммутаторов) для подключения к одному параллельному порту двух устройств.

Основным достоинством интерфейса Centronics является его стандартность — он есть на каждом компьютере и на всех компьютерах работает одинаково (правда, с разной скоростью). Для подключения внешнего устройства к параллельному порту не требуется открывать системный блок компьютера, что для многих пользователей может стать проблемой. Надо только подсоединить кабель к разъему на его задней стенке.

Можно также отметить такое достоинство интерфейса Centronics как простота его программирования на любом уровне. В большинстве языков программирования имеются процедуры взаимодействия с принтером, которые легко использовать и для

программирования нестандартного устройства. А так как с точки зрения программирования Centronics представляет собой три программно доступных регистра, не вызывает затруднений и написание программ нижнего уровня.

Итак, интерфейс Centronics можно рекомендовать в первую очередь для сопряжения с компьютером относительно несложных устройств без предъявления жестких требований по скорости информационного обмена и длине линии связи.

В данной главе рассматриваются основные принципы проектирования интерфейсов сопряжения устройств с параллельным портом.

3.1.2. Подключение простейших нестандартных устройств

Интерфейс Centronics и, соответственно, параллельный порт персонального компьютера, ориентированы на подключение принтера. Подтверждением этому является и название сигналов интерфейса — AUTO FD — автоматический перевод бумаги, PE — конец бумаги и т.д. Однако при разработке нестандартных устройств для подключения к интерфейсу Centronics его сигналы могут быть использованы произвольно.

Итак, какими же ресурсами располагает разработчик устройства, сопрягаемого с интерфейсом Centronics? Все сигналы интерфейса, которые уже были перечислены в главе 1, можно разделить на четыре группы:

1 — восьмиразрядная шина данных для записи из компьютера (сигналы D0...D7);

2 — четырехразрядная шина управления для записи из компьютера (сигналы -STROBE, -AUTO FD, -INIT и -SLCT IN);

3 — пятиразрядная шина состояния для чтения в компьютер (сигналы -ACK, BUSY, PE, SLCT и -ERROR);

4 — шина "земли".

Все сигналы программно доступны, что позволяет реализовать произвольные протоколы информационного обмена в рамках имеющегося их набора и быстродействия компьютера.

Простейший анализ набора сигналов позволяет выделить основную проблему, возникающую при сопряжении устройств с интерфейсом Centronics. Поскольку шина данных является однонаправленной, что позволяет использовать ее только на вывод, для ввода данных необходимо использовать сигналы из пятиразрядной шины состояния. Таким образом, если не предпринимать специальных шагов, разрядность информационного обмена по чтению ограничена пятью линиями.

Рассмотрим следующую задачу. Необходимо разработать устройство типа "набор лампочек и кнопочек", содержащее переключатели и светодиоды. В зависимости от положения переключателей по некоторому алгоритму должны загораться светодиоды. Устройство должно подключаться к персональному компьютеру через параллельный порт.

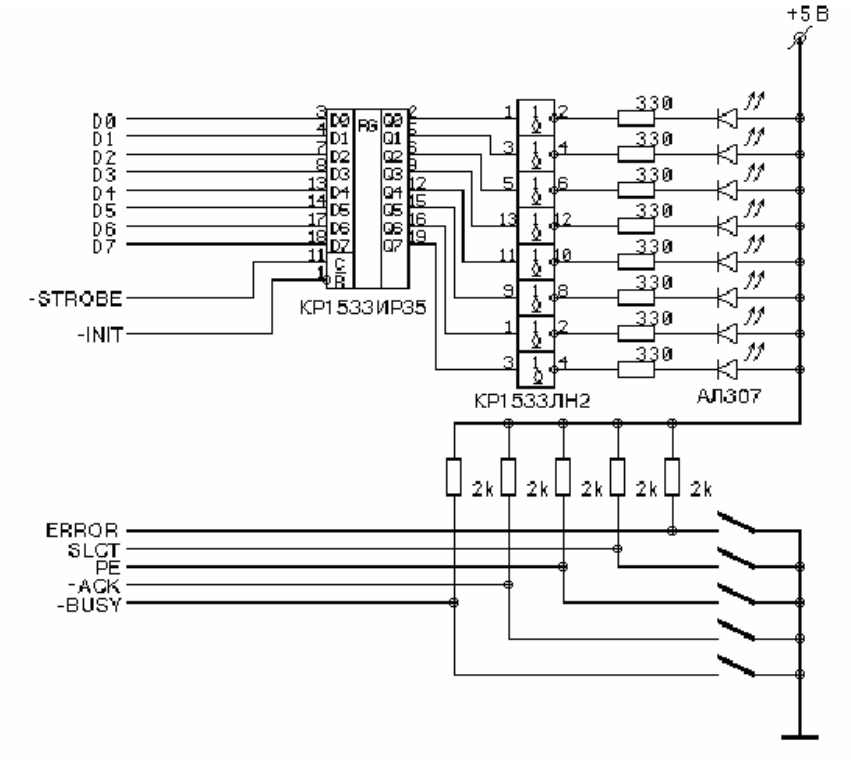


Рис. 3.1. Устройство типа "набор лампочек и кнопочек"

На рис. 3.1. показан вариант простейшей реализации такого устройства. Для управления светодиодами использованы сигналы D0...D7, поэтому их число ограничено восемью. Для опроса положения переключателей используются пять сигналов шины состояния, поэтому максимальное число таких переключателей — пять. Регистр КР1533ИР35 предназначен для фиксации данных и начальной установки (гашения) светодиодов. Сигнал -STROBE используется для стробирования записи данных в регистр, а сигнал -INIT — для сброса регистра и гашения светодиодов (светодиод горит при записи логической "1" в соответствующий разряд регистра). При замыкании переключателя по соответствующей ему линии шины состояния читается логический "0", а при размыкании — логическая "1". Для питания устройства используется внешний источник напряжением +5 В.

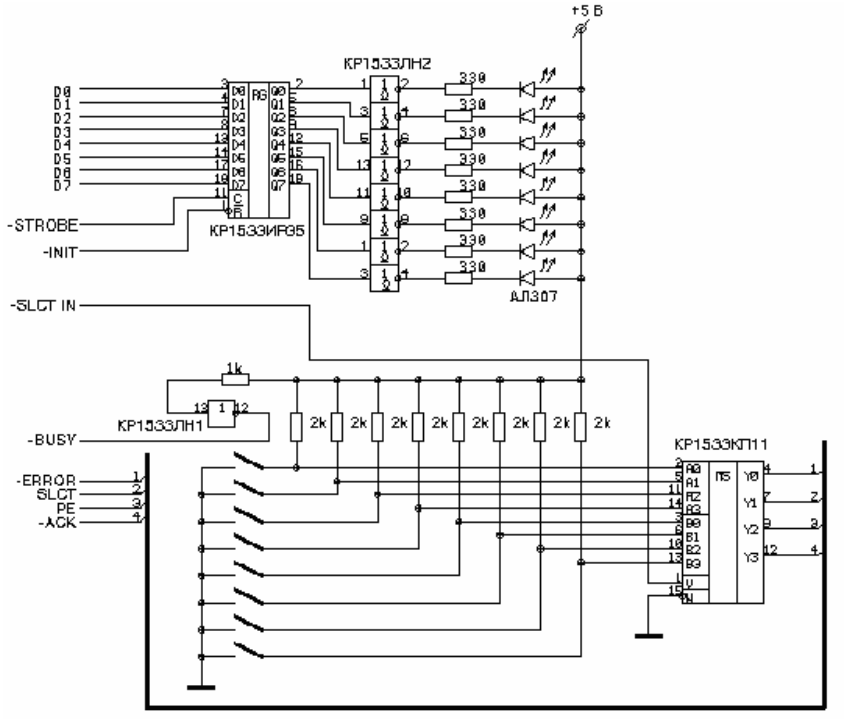


Рис. 3.2. Устройство типа "набор лампочек и кнопочек" с 8 переключателями

В устройстве, схема которого показана на рис. 3.1, количество переключателей и светодиодов определяется разрядностью соответствующих шин интерфейса Centronics. При этом если байтовой разрядности шины данных в некоторых случаях еще хватает, то пяти линий шины состояния для ввода явно недостаточно. Наиболее естественный и простой способ увеличения числа разрядов на ввод — мультиплексирование принимаемых данных.

На рис. 3.2. показана схема устройства типа "набор лампочек и кнопочек" уже с восемью переключателями. Мультиплексор КР1533КП11 реализует преобразование восьми линий в четыре, которое выполняется в две фазы — ввод младшей тетрады и ввод старшей тетрады. Для переключения мультиплексора можно использовать свободный сигнал шины управления, например, -SLCT IN.

Как видно, остается свободным один сигнал шины состояния. Его можно использовать, например, для определения наличия внешнего питания +5В (как показано на рис. 3.2) или детектирования подключения устройства (если поставить перемычку на "землю"). В обоих случаях предполагается, что с оборванной линии будет прочитана "1" (т.к. в адаптере Centronics компьютера стоит ТТЛШ-буфер).

Таким образом, использование мультиплексора позволило организовать двунаправленный байтовый информационный обмен через интерфейс Centronics. Это не так уж плохо, но все же в большинстве случаев байтовых шин недостаточно. В следующем параграфе рассмотрены простейшие способы расширения разрядности интерфейса.

3.1.3. Подключение модулей памяти

Рассмотрим задачу подключения к параллельному порту компьютера модуля памяти. Сама по себе она кажется несколько надуманной, однако модули памяти (ОЗУ или ПЗУ) могут входить в состав самых разных устройств, в том числе и сопрягаемых с

компьютером через параллельный порт.

На рис. 3.3 показана схема модуля ППЗУ емкостью 256 4-разрядных слов. Он построен на одной микросхеме КР556РТ4А. Для ее сопряжения с интерфейсом Centronics вполне достаточно разрядности шин данных и состояния. Сигнал -BUSY использован для детектирования подключения модуля ППЗУ.

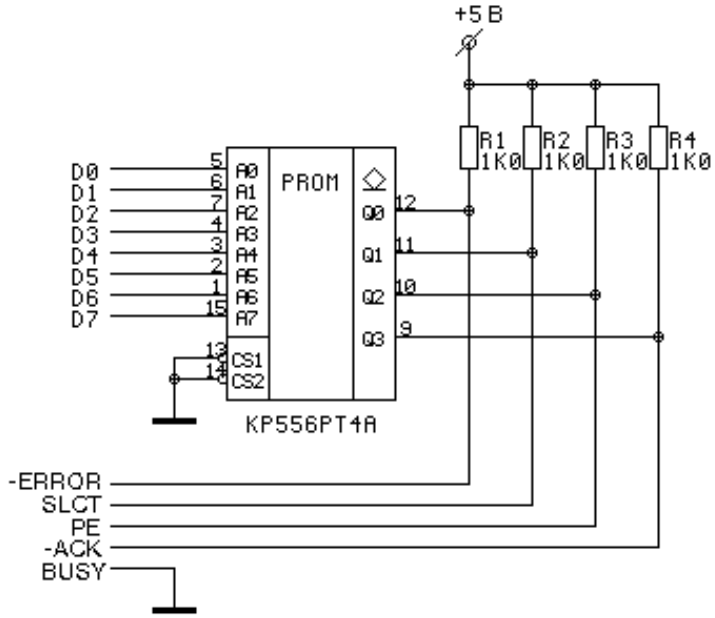


Рис. 3.3. Модуль ППЗУ емкостью 256 4-разрядных слов

На рис. 3.4а показана схема модуля ППЗУ емкостью 1 Кбайт, построенного на двух микросхемах КР556РТ5. В этом случае в качестве старших адресных линий приходится использовать две линии шины управления, а также мультиплексор для преобразования байта данных в две последовательно считываемые тетрады.

Такой же модуль ППЗУ можно подключить к параллельному порту несколько по-другому (рис.3.4б). Для установки адреса используются байтовые регистры КР1533ИР23, выбор которых реализуется дешифратором КР1533ИД7. Свободные биты регистров могут использоваться как внутренние сигналы управления устройства (в данном случае свободны 6 битов одного из регистров, и старший из них используется для переключения тетрад мультиплексора при чтении).

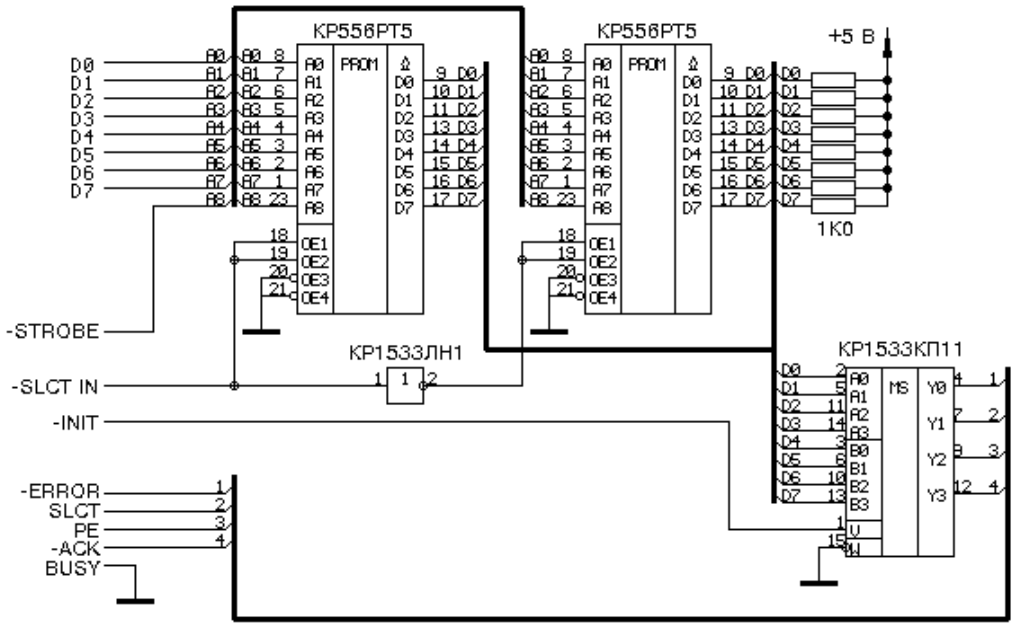


Рис. 3.4а. Модуль ППЗУ емкостью 1К байт

Такой способ позволяет интерпретировать интерфейс Centronics как примитивную магистраль с шинами адреса (сигналы -STROBE, -AUTO FD и -INIT), данных (шина данных интерфейса по записи и шина состояния по чтению) и стробом записи (сигнал -SLCT IN). При этом протоколы записи и чтения по этой магистрали реализуются программно (см. п.3.2). Для данного модуля ППЗУ вариант магистрального сопряжения является избыточным, однако он наиболее универсален.

На рис.3.5 показана схема магистрального сопряжения с интерфейсом Centronics модуля ОЗУ емкостью 16 Кбайт, построенного на двух микросхемах КР537РУ17. Основное отличие этой схемы от предыдущей связано с необходимостью реализации как чтения, так и записи, поэтому приходится использовать буфер КР1533АП5 для развязки выходов ОЗУ от шины данных при чтении. Кроме того, требуется дополнительный управляющий сигнал RE, переключающий режимы чтения (RE=0) и записи (RE=1). Сигнал управления мультиплексором назван здесь "1/2".

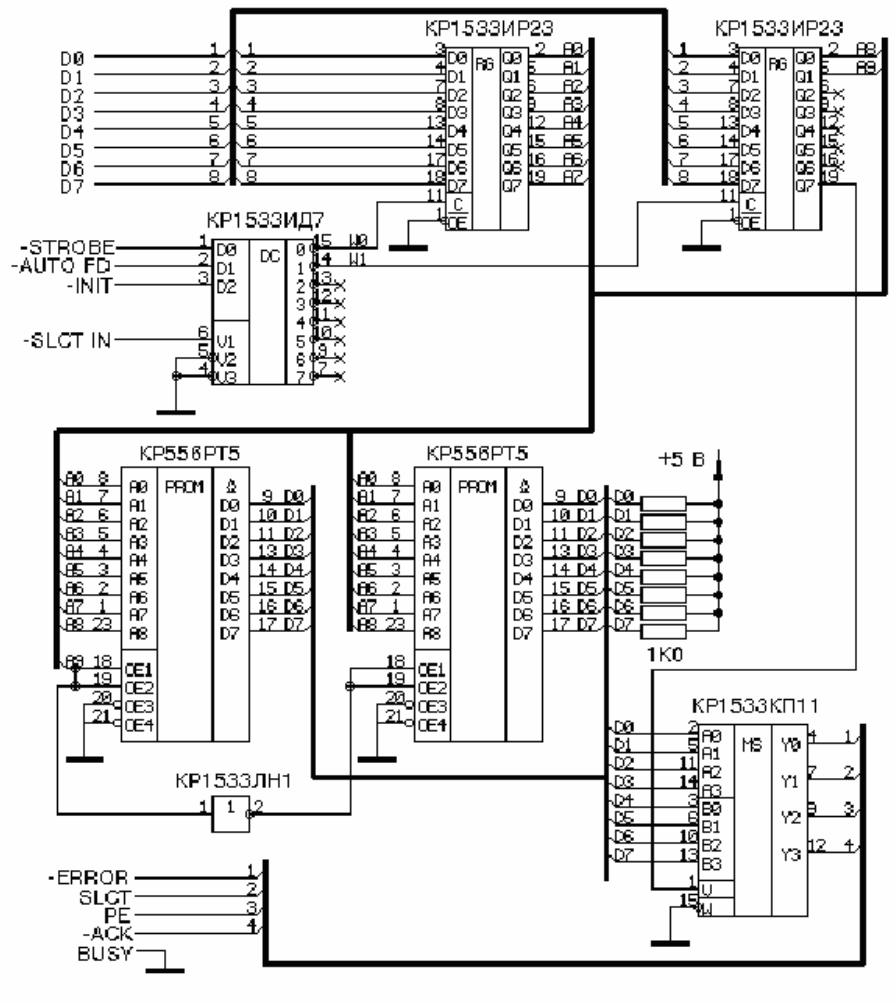


Рис. 3.4б. Модуль ППЗУ емкостью 1К байт (второй вариант)

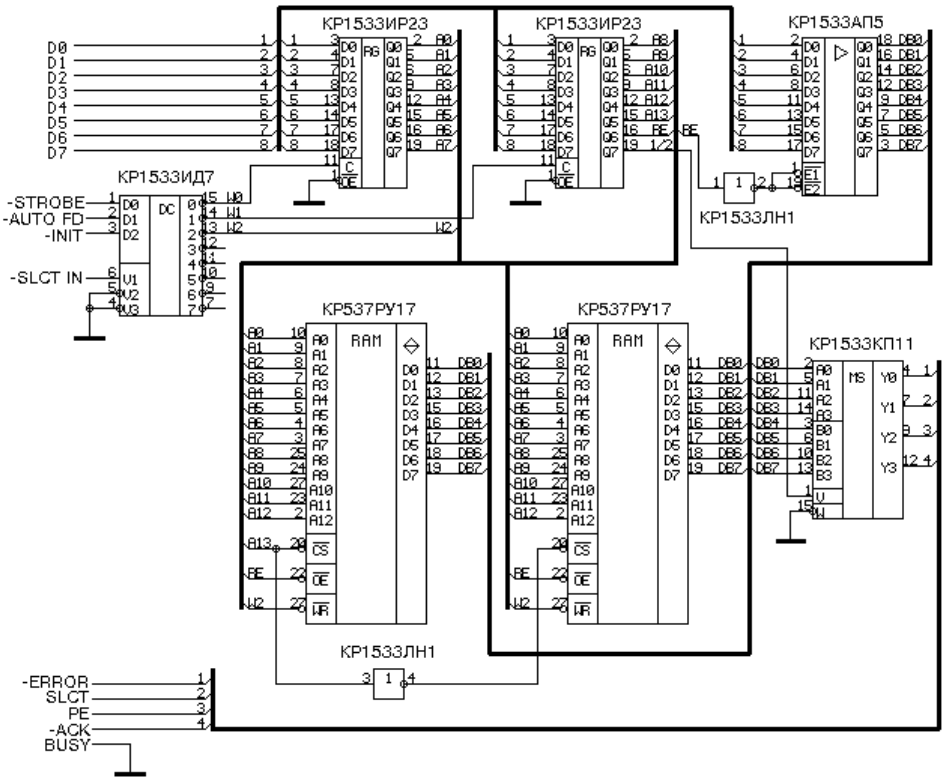


Рис.3.5. Модуль ОЗУ емкостью 16 К байт

3.1.4. Универсальный параллельный адаптер

Примером, достаточно полно иллюстрирующим возможности магистрального сопряжения устройств с интерфейсом Centronics, является схема универсального параллельного адаптера — УПА (рис. 3.6). УПА предназначен для информационного программно-управляемого обмена с периферийными устройствами по семи байтовым двунаправленным портам PORT0 — PORT6 и по своим возможностям аналогичен универсальному контроллеру параллельного обмена, описанному в 2.1.9 (как уже упоминалось, оба названия равнозначны).

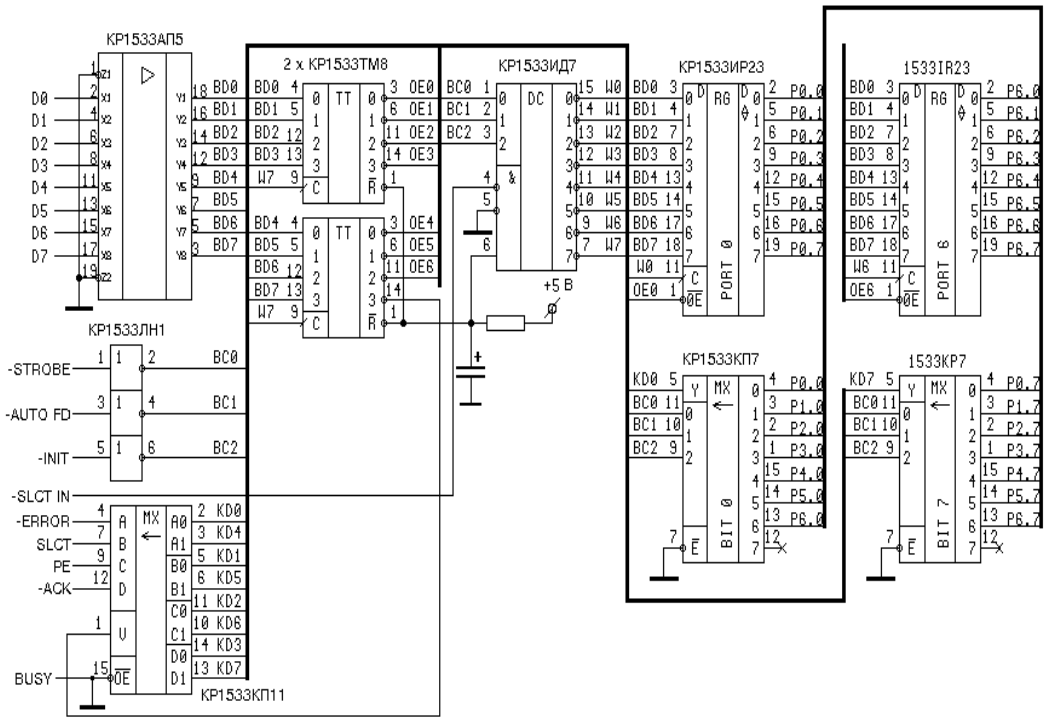


Рис. 3.6. Универсальный параллельный адаптер

Для вывода данных используются семь регистров КР1533ИР23 — по одному на каждый порт, а для ввода — восемь мультиплексоров КР1533КП7 — по одному на каждый бит данных. Семь младших битов регистра конфигурации (два триггера КР1533ТМ8) устанавливают направление передачи каждого из портов — чтение (0) или запись (1). Старший бит переключает тетрады входного мультиплексора КР1533КП11. При включении питания все порты устанавливаются на чтение.

Важно отметить, что так как к шинам данных и управления интерфейса Centronics должно быть подключено несколько нагрузок, то их приходится буферировать. Для этого предназначены микросхемы КР1533АП5 и КР1533ЛН1.

3.2. Проектирование программного обеспечения для информационного обмена через Centronics

Для персонального компьютера семейства IBM PC имеются две возможности программирования параллельного порта — обращение по физическим адресам регистров порта (нижний уровень) или вызов программного прерывания 17H (верхний уровень). При этом в обоих случаях может использоваться как ассемблер, так и язык высокого уровня. В данном разделе, как и ранее, все примеры написаны на языке Си.

3.2.1. Программирование на нижнем уровне

Параллельный порт имеет три адреса в пространстве устройств ввода-вывода компьютера:

BASE — регистр данных,
BASE+1 — регистр состояния,
BASE+2 — регистр управления.

Здесь "BASE" — первый адрес порта. В компьютере может быть до трех параллельных портов — LPT1 ... LPT3. Таблица базовых адресов портов находится в области данных BIOS, начиная с ячейки 408H: LPT1 — 0:408, LPT2 — 0:40A, LPT3 — 0:40C (все коды 16-ричные). Если порт не установлен, то в соответствующей ячейке записан 0.

Таким образом, перед обращением к порту необходимо выполнить процедуру определения его базового адреса. Ниже приведен фрагмент программы, устанавливающий базовый адрес порта LPT1 (переменная Cent_Base).

```
b = (int *) MK_FP (0,0x408);   // Указатель на ячейку 0:408
if ( *b != 0 )                 // Если порт установлен
 Cent_Base = *b;               // Считываем его базовый адрес
else                                   // Иначе — какие-то действия
 .....
```

Программирование подключенной к параллельному порту аппаратуры заключается в установке определенных битов в регистрах данных и управления и чтении определенных битов из регистра состояния. При этом если с регистром данных проблем не возникает (это обычный байтовый регистр), то два других регистра имеют некоторые особенности. Во-первых, некоторые биты являются инверсными. При записи в регистр управления нуля в этих битах устанавливаются единицы, а если на входах регистра состояния установлены нули, то из этих битов считываются единицы. Во-вторых, если четыре бита регистра управления расположены в младших битах байта (биты 0-3), то пять битов регистра состояния — в старших (биты 3-7). Полная информация об отображении сигналов шин управления и состояния интерфейса Centronics на регистры параллельного порта компьютера приведена в табл. 3.1.

ШИНА	СИГНАЛ	БИТ	ИНВЕРСИЯ
Управление	-STROBE	0	инверсный
	-AUTO FD	1	инверсный
	-INIT	2	прямой
	-SLCT IN	3	инверсный
Состояние	-ERROR	3	прямой
	SLCT	4	прямой
	PE	5	прямой
	-ACK	6	прямой
	-BUSY	7	инверсный

Таблица 3.1. Управляющие сигналы Centronics

Для того чтобы при программировании каждый раз не задумываться об особенностях того или иного бита, целесообразно один раз написать функции записи в регистр управления и чтения из регистра состояния, учитывающие инверсии битов и их расположение в байте. Вариант реализации таких функций приведен ниже.

```
void Cent_Control_drv (unsigned char control)
// Запись в регистр управления Centronics
{
```

```
 outportb (Cent_Base+2, (control ^ 0x0B) ^ 0x0E );
}

unsigned char Cent_Status_drv (void)
// Чтение из регистра управления Centronics
{
 return ( ( inportb (Cent_Base+1) >> 3 ) ^ 0x10 );
}
```

Что касается регистра данных, то запись в него производится командой вида:

```
outportb (Cent_Base, data);
```

3.2.2. Программирование на верхнем уровне

Программное прерывание 17H предоставляет некоторые возможности по работе с параллельным портом принтера. Однако сразу же следует отметить, что этих возможностей недостаточно для полноценного программирования подключенных к этому порту внешних устройств. Тем не менее, некоторые функции этого прерывания могут оказаться полезными.

Имеется три функции прерывания 17H, выбираемые значением регистра AH:

Функция №0 — печать символа.

Вход: AH = 0, AL — символ, DX — номер порта (0, 1 или 2).

Выход: AH — статус порта (см. функцию №2).

Функция №1 — инициализация порта.

Вход: AH = 1, DX — номер порта (0, 1 или 2).

Выход: AH — статус порта (см. функцию №2).

Функция №2 — определение статуса порта.

Вход: AH = 2, DX — номер порта (0, 1 или 2).

Выход: AH — статус порта:

Значения "1" в битах: 0 — таймаут,
3 — ошибка,
4 — принтер выбран,
5 — конец бумаги,
6 — подтверждение,
7 — принтер не занят.

Легко видеть, что устанавливаемые всеми функциями биты статуса соответствуют сигналам шины состояния интерфейса Centronics. Поэтому функцию №2 можно использовать для чтения данных с этой шины. Однако, как и в других случаях, использование прерывания существенно замедляет работу программы, поэтому рекомендуется непосредственно считывать данные по соответствующему адресу.

Действительно полезной оказывается функция №1 — инициализация порта. Дело в том, что эту процедуру необходимо выполнять после окончания работы с портами, иначе возможны конфликты с принтером. Поэтому рекомендуется вызывать функцию №1 прерывания 17H при выходе из программы.

3.2.3. Примеры программирования

В данном параграфе приведены примеры драйверов устройств, показанных на рис. 3.2, 3.5 и 3.6.

3.2.3.1. Драйверы устройства "набор лампочек и кнопочек"

Для показанного на рис. 3.2 устройства необходимо реализовать следующие процедуры:

— инициализация,
— запись в порт светодиодов,
— чтение состояния переключателей.

```
//***** Драйверы устройства "набор лампочек и кнопочек" *****

// Глобальная переменная — базовый адрес порта
int Cent_Base;

// Прототипы функций нижнего уровня (см. п.3.2.1)
// Запись в регистр управления Centronics
extern void Cent_Control_drv (unsigned char control);
// Чтение из регистра управления Centronics
extern unsigned char Cent_Status_drv (void);

// Прототипы функций
// Функция инициализации.
int Init_SP (void);
// Функция записи в регистр светодиодов
void Write_SP (unsigned char data);
// Функция чтения состояния переключателей
unsigned char Read_SP (void);

// Маски битов регистров управления и состояния
// Регистр управления
#define STROBE 0x01
#define AUTOFD 0x02
#define INIT 0x04
#define SLCTIN 0x08
// Регистр состояния
#define BUSY 0x10
#define STATUS_DATA 0x0F

// Функция инициализации.
// Определяет адрес порта, к которому подключено устройство
// (по сигналу наличия внешнего питания).
// Гасит все светодиоды.
// Возвращает:1 — устройство подключено, 0 — не подключено
int Init_SP (void)
{
int *b;

 b = (int *) MK_FP (0, 0x408);  // Указатель на ячейку 0:408
 while ( *b != 0 )                                  // Если порт установлен
 {
```

```
        Cent_Base = *b;                         // Считываем его базовый адрес
        if (Cent_Status_drv() & BUSY != 0)
        continue;                                       // Устройство не подключено
                                                        // к этому порту
        Cent_Control_drv (0);           // Сброс регистра светодиодов
        Cent_Control_drv (INIT);        // Разрешение записи
                                                        // в регистр светодиодов
        return 1;                                       // Устройство подключено
 }
 return 0;                                              // Устройство не подключено
}

// Функция записи в регистр светодиодов.
// Вход: data — байт данных, в котором каждый бит
//       соответствует светодиоду: 0 — не горит, 1 — горит
void Write_SP (unsigned char data)
{
 outportb (Cent_Base, data);                    // Установка данных
 Cent_Control_drv (STROBE | INIT);   // Строб записи = 1
 Cent_Control_drv (INIT);                       // Строб записи = 0
}

// Функция чтения состояния переключателей.
// Возвращает: байт данных, в котором каждый бит
//             соответствует переключателю:
//             0 — замкнут, 1 — разомкнут
unsigned char Read_SP (void)
{
unsigned char data;

 data = Cent_Status_drv() & STATUS_DATA;  // Чтение младшей
                                                                        // тетрады
 Cent_Control_drv (SLCTIN | INIT);              // Выбор старшей
                                                                        // тетрады
 data += (Cent_Status_drv() & STATUS_DATA) << 4; // Чтение
                                                        // старшей тетрады
 Cent_Control_drv (INIT);                       // Исходное состояние
 return data;
}
// Конец драйверов
```

Ниже приводится текст простейшей программы, зажигающей светодиоды при замыкании соответствующих переключателей.

```
#define ESC 0x1B        // Код клавиши ESC

#include <CONIO.H>

// Выход из программы — по клавише Esc
void main (void)
{
```

```
char key = 0;

 if (Init_SP() == 0)                    // Инициализация
 exit (1);                                      // Не подключено
 while (key != ESC)                     // Выход по Esc
 {
          while (! kbhit())                     // Проверка нажатия клавиши
          Write_SP ( Read_SP() );       // Чтение состояния
                                                        // переключателей и
                                                        // запись в регистр
                                                        // светодиодов
         key = getch();                         // Чтение кода клавиши
 }
}
```

3.2.3.2. Драйверы модуля ОЗУ

Драйверы показанного на рис. 3.5 модуля ОЗУ емкостью 16 Кбайт реализуют функции инициализации, чтения и записи данных по произвольному адресу.

```
//***** Драйверы модуля ОЗУ *****

// Глобальная переменная — базовый адрес порта
int Cent_Base;

// Прототипы функций нижнего уровня (см. п.3.2.1)
// Запись в регистр управления Centronics
extern void Cent_Control_drv (unsigned char control);
// Чтение из регистра управления Centronics
extern unsigned char Cent_Status_drv (void);

// Прототипы функций
// Функция инициализации.
int Init_RAM (void);
// Функция записи данных
void Write_RAM (unsigned addr, unsigned char data);
// Функция чтения данных
unsigned char Read_RAM (unsigned addr);

// Маски битов регистров данных, управления и состояния
// Регистр данных
#define T1_T2 0x80
#define RE 0x40
// Регистр управления
#define AL_W 0x00
#define AH_W 0x01
#define RAM_W 0x02
#define SLCTIN 0x08
// Регистр состояния
#define BUSY 0x10
```

```
#define STATUS_DATA 0x0F

// Функция инициализации.
// Определяет адрес порта, к которому подключен модуль
// (по сигналу BUSY).
// Возвращает:1 — модуль подключен, 0 — не подключен
int Init_RAM (void)
{
int *b;

 b = (int *) MK_FP (0, 0x408);  // Указатель на ячейку 0:408
 while ( *b != 0 )                              `       // Если порт установлен
 {
          Cent_Base = *b;                       // Считываем его базовый адрес
          if (Cent_Status_drv() & BUSY != 0)
          continue;                             // Модуль не подключен к этому порту
          Cent_Control_drv (0);  // Строб записи = 0
         return 1;                              // Модуль подключен
 }
 return 0;                                      // Модуль не подключен
}

// Функция записи данных в ОЗУ.
// Вход: data — байт данных, addr — адрес
void Write_RAM (unsigned addr, unsigned char data)
{
// Младший байт адреса ОЗУ
 outportb (Cent_Base, addr);            // Установка младшего
                                                // байта адреса
 Cent_Control_drv (AL_W);               // Адрес регистра младшего
                                                // байта адреса
 Cent_Control_drv (AL_W | SLCTIN);  // Строб записи = 1
 Cent_Control_drv (0);                  // Строб записи = 0

// Старший байт адреса ОЗУ
 outportb (Cent_Base, (addr>>8) | RE );// Установка старшего
                                                        // байта адреса
                                                        // и режима запись
 Cent_Control_drv (AH_W);                       // Адрес регистра старшего
                                                        // байта адреса
 Cent_Control_drv (AH_W | SLCTIN); // Строб записи = 1
 Cent_Control_drv (0);                          // Строб записи = 0

// Байт данных
 outportb (Cent_Base, data);                    // Установка байта данных
 Cent_Control_drv (RAM_W);              // Адрес данных
 Cent_Control_drv (RAM_W | SLCTIN);// Строб записи = 1
 Cent_Control_drv (0);                          // Строб записи = 0
}

// Функция чтения данных из ОЗУ
```

```
// Вход: addr — адрес.
// Возвращает: байт данных
unsigned char Read_RAM (unsigned addr)
{
unsigned char data;

// Младший байт адреса ОЗУ
 outportb (Cent_Base, addr);              // Установка младшего
                                                  // байта адреса
 Cent_Control_drv (AL_W);                 // Адрес регистра младшего
                                                  // байта адреса
 Cent_Control_drv (AL_W | SLCTIN); // Строб записи = 1
 Cent_Control_drv (0);                          // Строб записи = 0

// Старший байт адреса ОЗУ и чтение младшей тетрады данных
 outportb (Cent_Base, (addr>>8));         // Установка старшего
                                                  // байта адреса и
                                                  // режима чтения
                                                  // младшей тетрады данных
 Cent_Control_drv (AH_W);                 // Адрес регистра старшего
                                                  // байта адреса
 Cent_Control_drv (AH_W | SLCTIN); // Строб записи = 1
 Cent_Control_drv (0);                          // Строб записи = 0
 data = Cent_Status_drv() & STATUS_DATA; // Чтение младшей
                                                                  // тетрады

 // Старший байт адреса ОЗУ и чтение старшей тетрады данных
 outportb (Cent_Base, (addr>>8) | T1_T2);         // Установка
                                                  // старшего байта адреса и
                                                  // режима чтения
                                                  // старшей тетрады данных
 Cent_Control_drv (AH_W);                 // Адрес регистра старшего
                                                  // байта адреса
 Cent_Control_drv (AH_W | SLCTIN); // Строб записи = 1
 Cent_Control_drv (0);                          // Строб записи = 0
 data += (Cent_Status_drv() & STATUS_DATA) << 4; // Чтение
                                  // старшей тетрады

 Cent_Control_drv (0);                          // Исходное состояние
 return data;
}
// Конец драйверов
```

Ниже приведен текст программы, реализующей функциональный тест "Поле нулей" модуля ОЗУ.

```
#define RAM_SIZE 16384

#include <STDIO.H>
```

```
// Программа выводит на экран сообщение
// о результатах теста — OK или количество ошибок
void main (void)
{
unsigned i;
unsigned char data = 0;                    // Поле нулей
unsigned err;                              // Счетчик ошибок

 if (Init_RAM() == 0)                      // Инициализация
 exit (1);                                 // Не подключено
// Запись поля нулей
 for (i=0;i<RAM_SIZE:i++)
 Write_RAM (i, data);
// Чтение и сравнение с полем нулей
 for (i=0,err=0;i<RAM_SIZE:i++)
 if ( Read_RAM (i) != data )
          err++;
// Вывод на экран результатов теста
 if (err = 0)
 printf ("Нет ошибок.");
 else
 printf ("Ошибок — %u", err);
}
```

3.2.3.3. Драйверы универсального параллельного адаптера

Драйверы УПА реализуют функции инициализации, записи управляющего слова, записи данных в порт и чтения данных из порта.

```
//***** Драйверы УПА *****

// Глобальные переменные
int Cent_Base;  // Базовый адрес порта
unsigned char CW;       // Управляющее слово

// Прототипы функций нижнего уровня (см. п.3.2.1)
// Запись в регистр управления Centronics
extern void Cent_Control_drv (unsigned char control);
// Чтение из регистра управления Centronics
extern unsigned char Cent_Status_drv (void);

// Прототипы функций
// Функция инициализации.
int Init_UPA (void);
// Функция записи управляющего слова
void WriteCW_UPA (void);
// Функция записи данных в порт
void WriteP_UPA (unsigned port, unsigned char data);
// Функция чтения данных из порта
```

```
unsigned char ReadP_UPA (unsigned port);

// Маски битов регистров данных, управления и состояния
// Регистр данных — бит переключения тетрад при чтении
#define T1_T2 0x80
// Регистр управления
#define SLCTIN 0x08
// Регистр состояния
#define BUSY 0x10
#define STATUS_DATA 0x0F

// Функция инициализации.
// Определяет адрес порта, к которому подключен модуль
// (по сигналу BUSY).
// Возвращает:1 — модуль подключен, 0 — не подключен
int Init_UPA (void)
{
int *b;

 b = (int *) MK_FP (0, 0x408);  // Указатель на ячейку 0:408
 while ( *b != 0 )                        // Если порт установлен
 {
          Cent_Base = *b;                         // Считываем его базовый адрес
          if (Cent_Status_drv() & BUSY != 0)
          continue;                               // Модуль не подключен к этому порту
          Cent_Control_drv (0xF);         // Строб записи = 1 -
                                                  // исходное состояние
         return 1;                                // Модуль подключен
 }
 return 0;                                        // Модуль не подключен
}

// Функция записи управляющего слова УПА
void WriteCW_UPA (void)
{
 WriteP_UPA (7, CW);
}

// Функция записи данных в порт УПА
// Вход: data — байт данных, port — номер порта (0...7)
void WriteP_UPA (unsigned port, unsigned char data)
{
 Cent_Control_drv ( (port^0x7) | SLCTIN );      // Номер порта
                                                                  // (с инверсией) и
                                                                  // строб записи = 1
 outportb (Cent_Base, data);                            // Установка байта данных
 Cent_Control_drv (port^0x7);                   // Строб записи = 0
 Cent_Control_drv ( (port^0x7) | SLCTIN );      // Строб записи =1
}
// Функция чтения данных из порта УПА
// Вход: port — номер порта (0...6).
```

```
// Возвращает: байт данных
unsigned char ReadP_UPA (unsigned port)
{
unsigned char data;

// Чтение младшей тетрады
 WriteCW_UPA (CW | T1_T2);            // Режим чтения младшей тетрады
 Cent_Control_drv ( (port^0x7) | SLCTIN );      // Номер порта
                                                                    // (с инверсией) и
                                                                    // строб записи = 1
 data = Cent_Status_drv() & STATUS_DATA; // Чтение младшей
                                                    //тетрады

 // Чтение старшей тетрады
 WriteCW_UPA (CW);                    // Режим чтения старшей тетрады
 Cent_Control_drv ( (port^0x7) | SLCTIN );      // Номер порта
                                                                    // (с инверсией) и
                                                                    // строб записи = 1
 data += (Cent_Status_drv() & STATUS_DATA) << 4; // Чтение
                                                                    // старшей тетрады
 return data;
}
// Конец драйверов
```

ГЛАВА 4

РАЗРАБОТКА УСТРОЙСТВ СОПРЯЖЕНИЯ ДЛЯ RS-232C

Наряду с параллельными методами обмена информацией, к которым относятся применение интерфейса CENTRONICS и подключение к системной магистрали ISA, можно использовать и интерфей последовательного обмена RS-232C. Его применение имеет свои особенности, о которых уже упоминалось в первой главе.

Несмотря на очевидные скоростные преимущества параллельных методов, их применение оказывается затруднительным, а часто и вовсе невозможным, в случаях, когда по ряду причин требуется организовать обмен со сколько-нибудь удаленным внешним устройством. В подобных ситуациях, если интенсивность обмена не слишком высока (предполагается, что разработчик знает основные требования, предъявляемые к системе), применение интерфейса RS-232C вполне оправдано, тем более, что персональный компьютер, не имеющий встроенных последовательных каналов ввода-вывода (портов RS-232C) встречается в наше время крайне редко.

Таким образом, выбор в пользу применения интерфейса RS-232C может быть сделан при наличии следующих требований:

- относительная удаленность объекта обмена информацией (внешнего устройства) от компьютера (стандартом оговорена длина кабеля до 15 м при наличии общего контура заземления, однако во многих практических случаях она может быть существенно увеличена, хотя и с некоторым снижением рабочих скоростей);
- сравнительно (по отношению к параллельным методам и локальным вычислительным сетям) невысокая скорость обмена данными (максимально возможная скорость передачи данных стандартного последовательного порта компьютера составляет 115200 бит/сек, что ограничивает скорость обмена величиной около 10 Кбайт/сек);
- требование применения стандартного интерфейса для подключения к компьютеру без его вскрытия (несмотря на то, что времена, когда установка любой дополнительной платы в компьютер представлялась кощунством и вызывала дрожь его хозяина, прошли, применение RS-232C для подключения внешних устройств существенно упрощает процесс подключения и повышает оперативность в работе).

Далее в настоящей главе приведена информация, пользуясь которой разработчик сможет осуществить сопряжение проектируемого устройства с компьютером при помощи интерфейса RS-232C. Следует, однако, иметь в виду, что при изложении материала авторы нисколько не претендовали на всестороннее освещение вопроса, что потребовало бы отдельной книги, а стремились только донести до читателя некоторые свои проверенные на практике технические решения.

4.1. Постановка задачи сопряжения

При использовании интерфейса RS-232C задача сопряжения объекта обмена информацией с компьютером обычно формулируется следующим образом: требуется обеспечить связь с удаленным контроллером, обслуживающим технологическую или лабораторную установку. Именно этот контроллер играет в данном случае роль УС.

Чаще всего такой контроллер представляет собой микроЭВМ, имеющую собственную магистраль и набор внешних устройств, осуществляющих передачу входных сигналов с разнообразных датчиков (если таковые имеются) и выдачу управляющих воздействий на органы управления (если они присутствуют). Для нас существенным моментом является наличие в контроллере процессора, обрабатывающего информацию,

представленную в параллельной форме, и магистрали, обеспечивающей взаимодействие раздичных его узлов. Если же требуется организовать сопряжение с устройством, не имеющим собственного интеллекта, задача сразу же существенно усложняется и часто становится практически невыполнимой. Поэтому в таком случае стоит подумать о выборе других путей сопряжения.

Этапы преобразования сигналов интерфейса RS-232C на пути от компьютера к микропроцессору удаленного контроллера достаточно очевидны и проиллюстрированы рис. 4.1. Здесь и далее мы считаем, что для сопряжения через RS-232C используется наиболее распространенная простейшая 4-проводная линия связи

Блок преобразователей уровня обеспечивает электрическое согласование уровней сигналов последовательного интерфейса, формируемых контроллером, входящим в состав компьютера (±12 В), с уровнями сигналов, присутствующими в микропроцессорной системе (здесь и далее предполагаем, что в микропроцессорной системе действуют уровни ТТЛ).

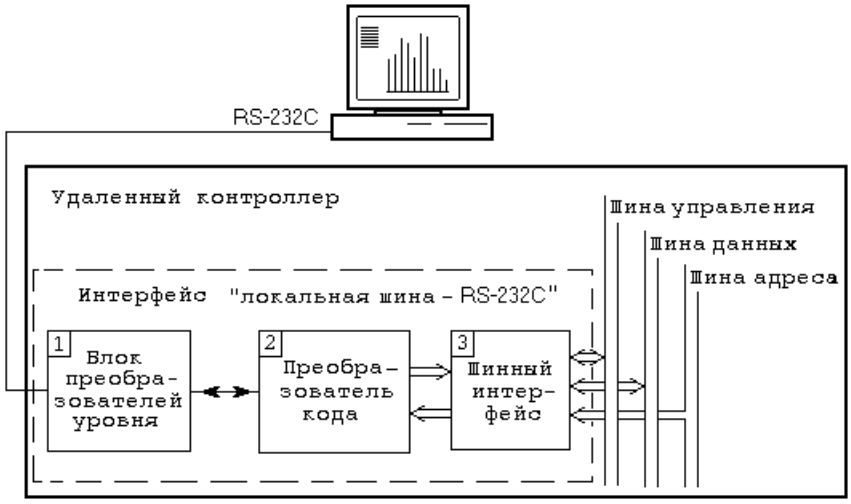


Рис. 4.1. Организация сопряжения через интерфейс RS-232C

Блок преобразователя кода переводит последовательрое представление информации в параллельное (и наоборот), осуществляя распознавание начала и конца посылки, синхронизацию приема-передачи битов кадра, слежение за наличием ошибок, информирование о готовности к выполнению операций и т. п.

Интерфейс шины обеспечивает сопряжение преобразователя кода с локальной магистралью микропроцессорной системы, осуществляя двунаправленную передачу данных в соответствии с алгоритмами и временными соотношениями, принятыми в ней.

Таким образом, даже на этапе постановки задачи ситуация в случае использования RS-232C существенно отличается от рассмотренных ранее (при использовании ISA и Centronics).

4.2. Схемотехника преобразователей уровня

В зависимости от требований, предъявляемых к проектируемой системе, преобразователи уровня могут быть выполнены самыми различными способами.

Так в случае, когда требуется передача и прием всего лишь одной пары сигналов (TxD и RxD), и желательно иметь возможность быстро восстановить работоспособность

системы в случае выхода ее из строя, можно применить передатчик, выполненный на дискретных компонентах (рис 4.2).

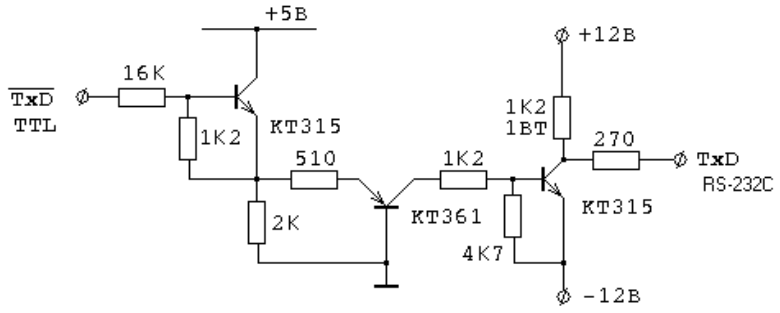


Рис. 4.2. Передатчик на дискретных компонентах

Функционирование схемы вряд ли нуждается в комментариях, а транзисторы КТ315 и КТ361, как правило, являются непременным атрибутом любой лаборатории.

Для простейшего приемника сигналов последовательного порта вообще не требуется никаких дополнительных схем кроме входных ограничителей уровня и инвертора, но все же неплохо, если приемник будет обладать некоторым гистерезисом. Вполне пригодная к использованию схема приведена на рис. 4.3.

Часто однако случается, что применение дискретных элементов является нежелательным, а если к тому же требуется наличие более двух сигналов последовательного интерфейса, то лучшим решением может оказаться использование специализированных интерфейсных интегральных микросхем преобразователей уровня, например, IFC1488 (КР559ИП19) и IFC1489 (КР559ИП20), представленных на рис. 4.4 и рис. 4.5.

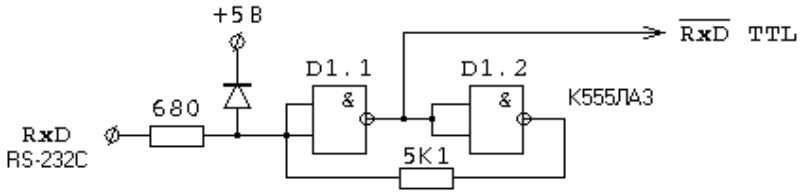


Рис. 4.3. Простейший приемник сигналов

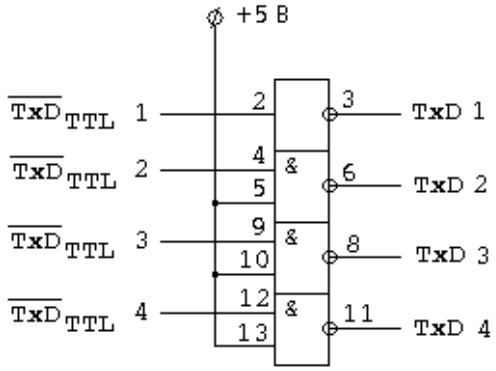


Рис.4.4. Микросхема передатчика IFC1488 (КР559ИП19). Корпус DIP-14. Вывод 1 — -12В, вывод 14 — +12В, вывод 7 — общий.

Выходы микросхемы IFC1488 имеют последовательно включенные резисторы номиналом 300 Ом и обеспечивают биполярные уровни Uвых i ± 6 В.

По информационным входам (RxD) микросхема обладает гистерезисом, положение которого определяется напряжением, поданным через токоограничительный резистор на выводы g. Допускается оставлять эти выводы неподсоединенными.

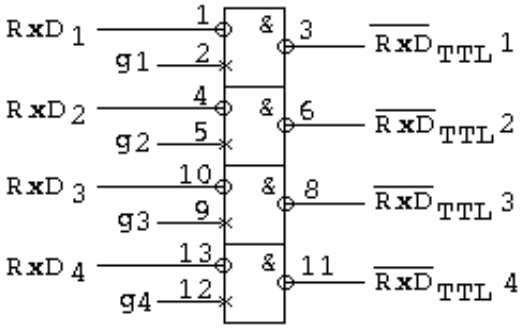


Рис. 4.5. Интегральная микросхема приемника IFC1489 (КР559ИП20). Корпус DIP-14. Вывод 14 — +5 В, вывод 7 — общий

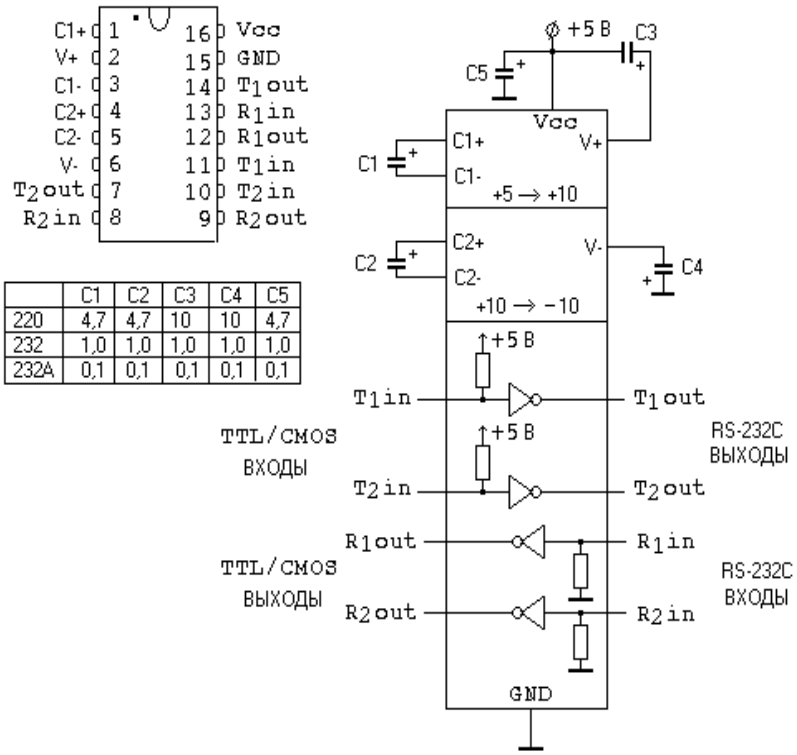
Микросхемы КР559ИП19 и ИП20 выпускаются в настоящее время белорусским объединением "Интеграл" (г. Минск).

С точки зрения разработчика цифровой аппаратуры, очевидным недостатком обоих типов преобразователей уровня является необходимость наличия источника двуполярного напряжения питания ±12 В. Несмотря на то, что в компьютере такие напряжения имеются, на разъем интерфейса RS-232C они не выведены, и использовать их довольно-таки затруднительно.

Обойтись без дополнительных источников питания возможно, применив специально разработанные для этой цели интерфейсные микросхемы. Широкая гамма таких кристаллов выпускается известной фирмой MAXIM.

Они содержат преобразователь напряжения +5 В в напряжение +10 В (генератор + умножитель напряжения), инвертор (преобразующий напряжение +10 В в -10 В) и собственно преобразователи уровней сигналов последовательного интерфейса. Большинство из них требуют дополнительных элементов (необходимы внешние конденсаторы), что не является чрезмерной платой за преимущества их применение.

Полная номенклатура изделий вместе со всей необходимой подробной информацией по их применению имеется в соответствующих справочных материалах фирмы. Ниже приведены некоторые данные лишь по нескольким микросхемам.



	C1	C2	C3	C4	C5
220	4,7	4,7	10	10	4,7
232	1,0	1,0	1,0	1,0	1,0
232A	0,1	0,1	0,1	0,1	0,1

Рис. 4.6. Микросхемы MAX 220, MAX 232, MAX 232A фирмы MAXIM. В таблице приведены номиналы конденсаторов в микрофарадах

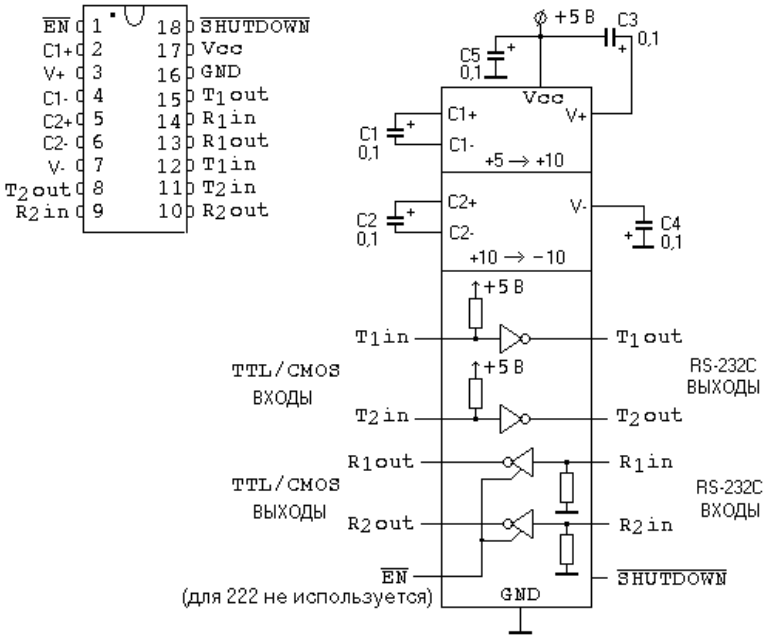


Рис. 4.7. Микросхемы MAX 222 и MAX 242 фирмы MAXIM

В заключение обзора преобразователей уровня представляется полезным упомянуть о еще одной схеме, которая может оказать помощь в случае, если требуется обеспечить связь при значительном удалении обслуживаемого объекта от компьютера с сохранением в качестве порта ввода-вывода стандартрого последовательного интерфейса. Это схема (рис. 4.9) преобразователя RS-232C — "токовая петля 20ma", осуществляющая к тому же гальваническую развязку компьютера от объекта управления.

Отметим, что гальваническая развязка бывает крайне необходима при сколько-нибудь значительном расстоянии между любыми объектами, соединяемыми электрическим кабелем. В подобных ситуациях известны довольно многочисленные случаи выхода из строя компьютеров, которые были заземлены, но находились в разных комнатах. Измерения токов, текущих по нулевому проводу соединительных кабелей, дают такие сумасшедшие величины, как 2-3 А, несмотря на то, что оба конца этого провода вроде бы заземлены.

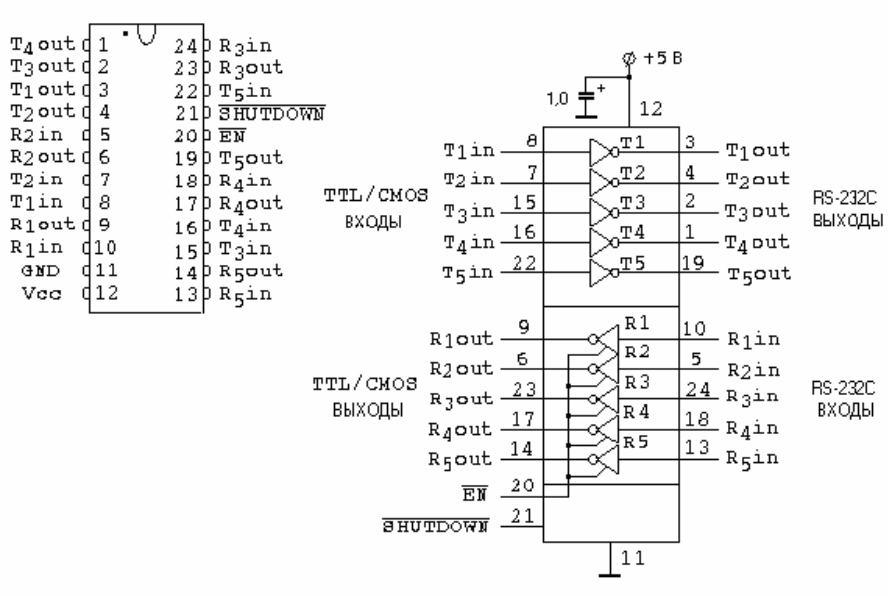


Рис. 4.8. Микросхема MAX 235 фирмы MAXIM

Данная схема (рис. 4.9) обеспечивала связь компьютера с удаленным контроллером при длине кабеля около 2 км на скорости 38400 бит/сек. Очевидно, что для двунаправленной связи приемная и передающая части должны быть подключены к обоим концам кабеля.

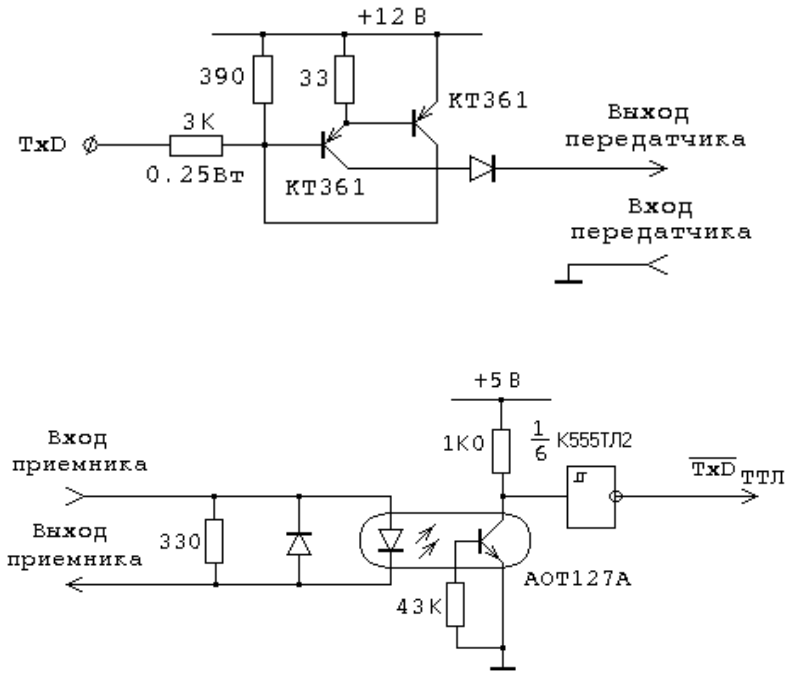


Рис. 4.9. Преобразователь RS-232C — "токовая петля 20 ма", активный передатчик — пассивный приемник.

4.3. Схемотехника преобразователей кода

Преобразование кода из параллельного в последовательный (и наоборот) может осуществляться различными способами в зависимости от конкретной реализации контроллера, с которым производится сопряжение. Выбор того или иного способа во многих случаях достаточно очевиден.

Наиболее просто проблема разрешается в том случае, если в качестве центрального процессора удаленного контроллера применена однокристалльная микроЭВМ, уже содержащая Универсальный асинхронный последовательный приемо-передатчик (УАПП). В качестве примера такой микроЭВМ можно упомянуть микросхему КР1816ВЕ31. Более того, возможно, что выбор подобной микроЭВМ как раз и обусловлен наличием в ней встроенного последовательного порта. Ясно, что построение преобразователя кода в данном случае сводится к задействованию встроенного ресурса в соответствии со спецификациями на примененную микросхему.

Если же центральный процессор контроллера не обладает встроенным УАПП, или его использование по каким-либо причинам невозможно, а также в случае, когда требуется несколько каналов последовательного ввода-вывода, преобразователи кодов выполняются на основе специализированных внешних интегральных схем.

Одной из них является ныне безнадежно устаревшая микросхема программируемого универсального синхронно-асинхронного приемо-передатчика i8251 (КР580ВВ51). Она содержит шинный интерфейс и собственно преобразователь кода. Микросхема требует двух внешних задающих генераторов частоты (один из которых задает скорость передачи, а другой формирует внутренние тактирующие импульсы), что является ее очевидным недостатком. Эта микросхема подробно описана во множестве книг, посвященных построению микропроцессорных структур, но сейчас уже практически нигде не используется.

Дальнейшим развитием идей, реализованных при проектировании указанного УАПП, являются микросхема i8250 (ее аналог — микросхема КР1847ВВ2 Минского объедининия "Интеграл") и совпадающие с нею по цоколевке и назначению основных регистров ИМС TL16C450 и TL16C550 фирмы Texas Instruments (последняя содержит стеки FIFO размером по 16 байт для передающей и приемной частей).

На рис. 4.10 приведены назначение выводов ИМС i8250 для корпуса DIP-40 и одна из возможных типовых схем включения. Подробное описание внутренней структуры этой микросхемы и порядка обмена с ней можно найти во многих книгах или в интерактивном справочнике TechHelp! (с которым жизнь разработчика приложений для IBM-совместимых персональных компьютеров становится чуть-чуть проще). Краткое описание форматов обмена с данной микросхемой, входящей в состав компьютера, приведено в приложении.

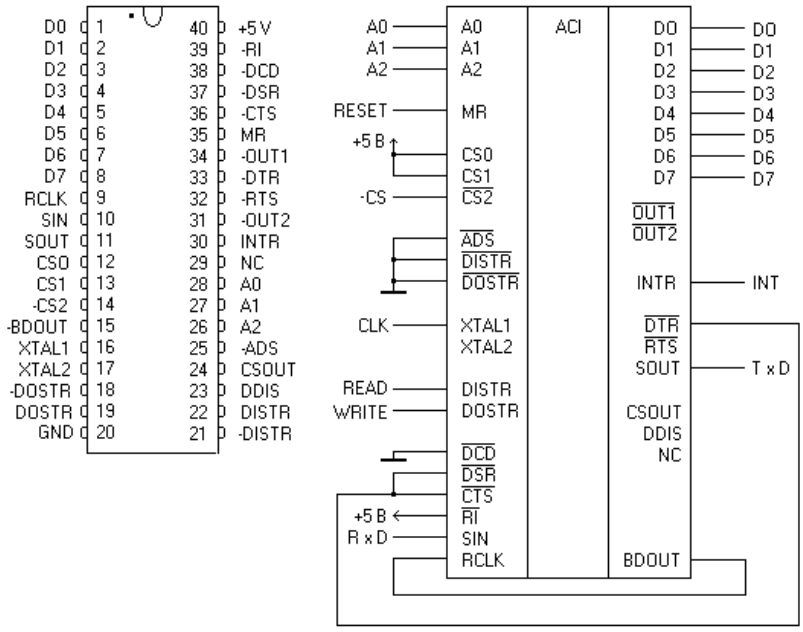


Рис. 4.10. Назначение выводов и одна из типовых схем включения микросхемы i8250

Еще одним, хотя и довольно экзотическим, способом построения преобразователя кода можно воспользоваться, если нежелательно применение дополнительных микросхем, а вычислительные возможности центрального процессора задействованы не полностью (правда еще потребуются как минимум два разряда параллельного интерфейса по одному на ввод и на вывод информации).

Способ заключается в чисто программном преобразовании кодов при считывании (либо при выводе) информации через какой-нибудь из свободных разрядов параллельного интерфейса используемой микроЭВМ.

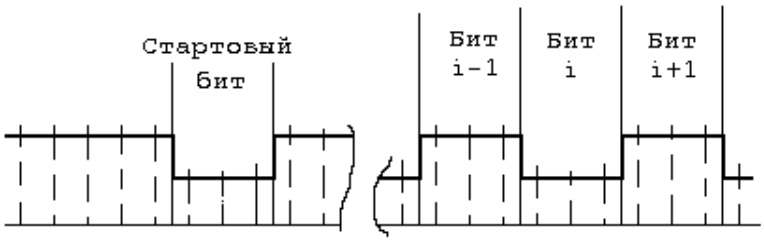


Рис. 4.11. Программный опрос входного сигнала RS-232C

Ниже приведен фрагмент программы для однокристальной микроЭВМ КР1816ВЕ31, осуществляющий ввод одного бита входной последовательности. Предполагается, что регистр DPTR содержит адрес параллельного порта ввода информации, сигнал Rx ТТЛ соответствует разряду D0. Отсчет берется трижды, чтобы уменьшить возможное влияние помех (рис. 4.11).

```
                ..
INBIT: CLR      R0              ;очистка регистра результата
       MOV      R1,#3           ;инициализация счетчика отсчетов
BITLP: MOVX A,@DPTR             ;ввод состояния сигналов
```

```
        RRC             ;выделение D0 (флаг CARRY ¬ D0)
        JC      ZERO    ;анализ состояния сигнала
ONE:    INC     R0      ;единичное состояние
        SJMP    WAIT
ZERO:   DEC     R0      ;нулевое состояние
WAIT:   LCALL   WDTIME  ;пауза до следующего отсчета
        DJNZ    R1,BITLP ;подсчет количества отсчетов
        RET
        ..
```

Решение о состоянии информационного сигнала RxD принимается на основе анализа знака содержимого регистра R0.

Аналогичным образом составляются подпрограммы ожидания стартового и стоповых битов, прием бита четности и т. п.

4.4. Примеры проектирования устройств сопряжения для RS-232C

Рассмотрим кратко два примера реальных УС для связи через RS-232C.

Первое устройство предназначено для управления удаленным от компьютера объектом. Как уже отмечалось, в данном случае УС должно содержать в себе однокристальную микроЭВМ, которая и будет осуществлять как управление объектом (внешним устройством), так и обменом по линии связи RS-232C. Наиболее удобно при этом использовать микросхемы однокристальных микроЭВМ, имеющих вход и выход последовательного канала, например, КР1830ВЕ31. Структурная схема УС на основе этой микросхемы (рис. 4.12) помимо стандартных элементов типовой схемы ее включения (ОЗУ для хранения данных, ППЗУ для хранения микропрограмм, регистров для демультиплексирования адреса и данных, а также микросхемы параллельного порта для связи с внешними устройствами) содержит только уже рассмотренные ранее (рис. 4.2 и 4.3) узлы приемника RS-232C и передатчика RS-232C, осуществляющие требуемое преобразование уровней. Схема управления производит разделение адресного пространства микроЭВМ и выработку управляющих стробов обмена. Схема выработки сигнала сброса на рисунке не показана.

Мы не будем подробно останавливаться на структуре и особенностях работы КР1830ВЕ31 и других однокристальных микроЭВМ. Информация об этом имеется в многочисленных справочниках. Отметим только, что, изменяя программное обеспечение, зашитое в ППЗУ, можно обеспечить выполнение самых различных алгоритмов управления широким спектром внешних устройств. Что касается аппаратуры, то здесь обычно используется стандартная схема включения, и разработчик УС должен только аккуратно ее повторить. Поэтому говорить об особенностях проектирования отдельных узлов УС данного типа не приходится.

Отметим, что универсальность схемы включения существенно облегчает труд разработчика, но нисколько не снижает важности процесса отладки УС. Отладка подобных универсальных контроллеров на основе однокристальных микроЭВМ имеет свои особенности, некоторые из которых будут рассмотрены несколько позже, и существенно отличается от отладки УС на жесткой логике. В частности, здесь наиболее существенной частью процесса отладки становится составление нужной программы для зашивки в ППЗУ.

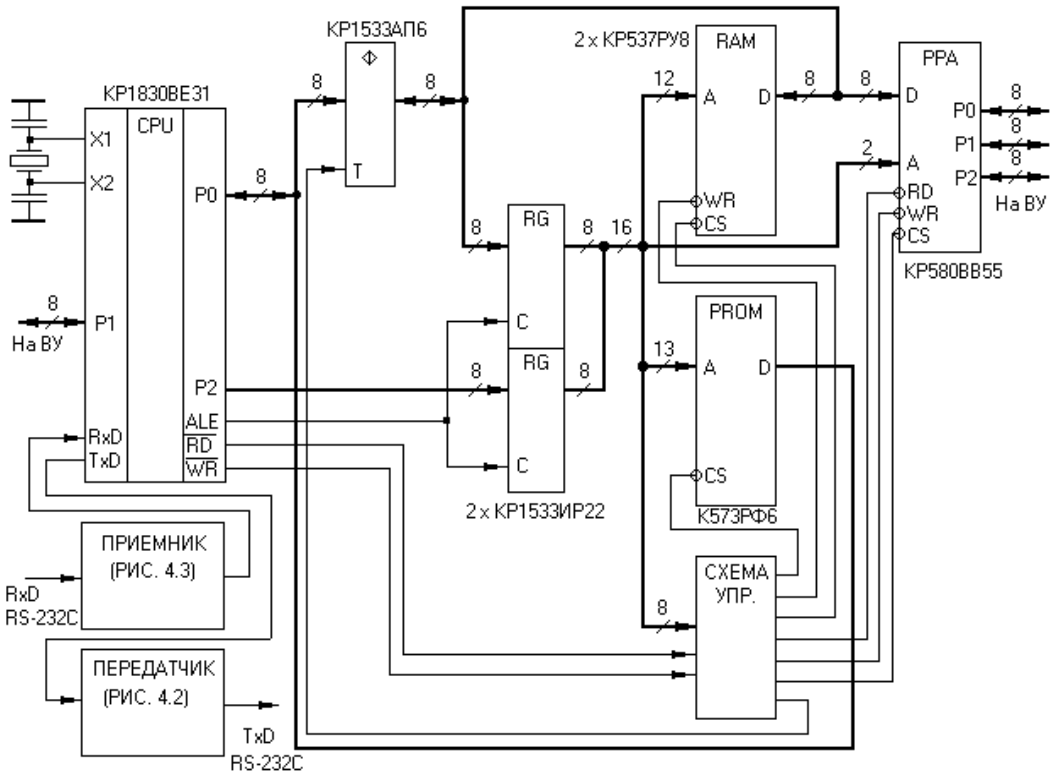


Рис. 4.12. Структурная схема УС на базе однокристальной микроЭВМ КР1830ВЕ31

Вторая схема УС, которую мы рассмотрим, выполняет несколько иную функцию. Дело в том, что имеющихся обычно в составе компьютера двух последовательных портов (COM1 и COM2) в ряде случаев может оказаться недостаточно. Простейший пример — необходимость организации связи данного компьютера с несколькими другими компьютерами или с несколькими терминалами. Конечно, в такой ситуации можно приобрести и подключить нужное количество стандартных плат расширения, имеющих два последовательных порта и один параллельный, но подобное решение часто оказывается далеко не оптимальным, так как требует наличия свободных разъемов ISA и приводит к большой аппаратурной избыточности.

Предлагаемое УС (рис. 4.13) обеспечивает обмен компьютера с восемью портами интерфейса RS-232C и занимает всего одну стандартную плату расширения ISA. Для преобразования уровней в схеме УС использованы микрохемы передатчиков IFC1488 (рис. 4.4) и микросхемы приемников IFC1489 (рис. 4.5). В качестве преобразователей кодов применены микросхемы i8250 (схема включения приведена на рис. 4.10).

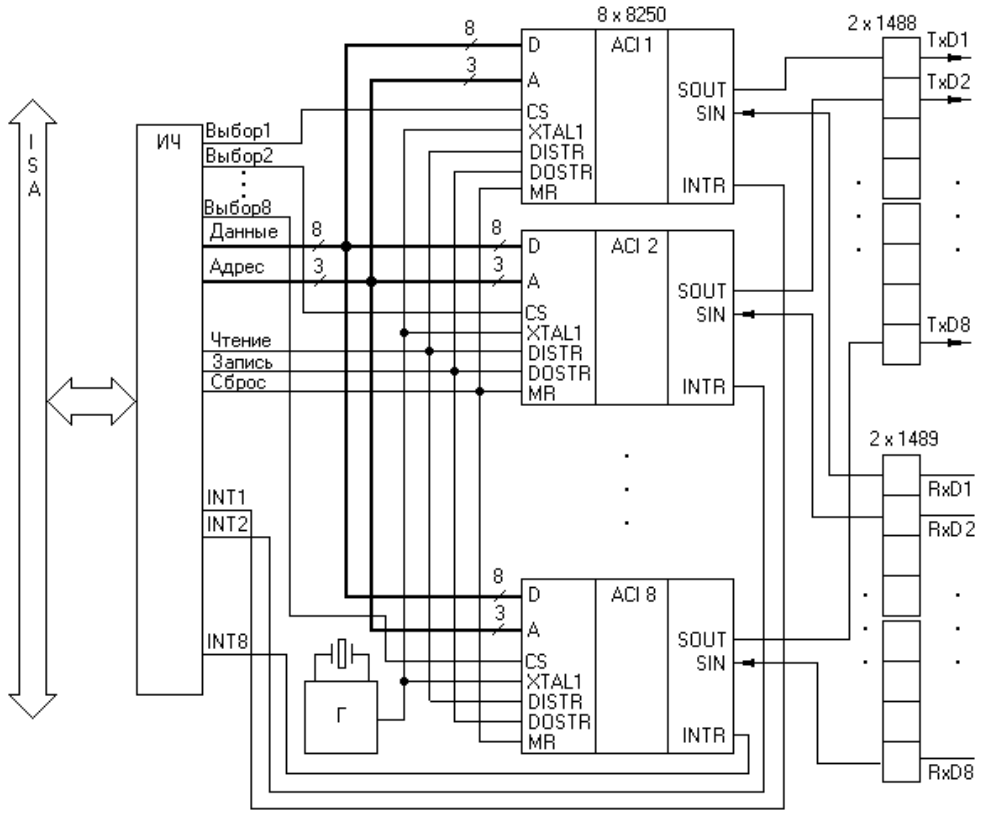


Рис. 4.13. 8-канальное УС для связи по интерфейсу RS-232C, ориентированное на сопряжение с ISA

Выбор одного из восьми приемопередатчиков производится по сигналам Выбор 1 ... Выбор 8. Адресация внутренних регистров приемопередатчиков осуществляется трехразрядной адресной шиной. Данные передаются по 8-разрядной шине. Таким образом, в адресном пространстве устройств ввода/вывода персонального компьютера данное УС занимает 64 адреса (8 окон по 8 адресов). Выходные сигналы прерываний всех приемопередатчиков подаются на интерфейсную часть УС и формируют единственный сигнал IRQ. Определение источника прерывания производится с помощью программного опроса сигналов INT1 ... INT 8.

4.5. Разработка программного обеспечения для RS-232C

В качестве примера программирования последовательного порта приведем подпрограммы его инициализации и приема-передачи данных. Для начала определим некоторые постоянные, которые мы будем использовать в дальнейшем. Файл описаний (назовем его SERIAL.H) может выглядеть следующим образом:

```
//Определение адресов регистров порта
int     BASE; //Базовый (первый) адрес блока регистров порта
#define OUT_REG     BASE  //Регистр передачи данных
#define IN_REG              BASE  //Регистр приема данных
#define LOW_DIV     BASE  //Младший байт делителя
#define HIGH_DIV    BASE+1        //Старший байт делителя
#define INT_REG     BASE+1        //Маска разрешенных прерыва-
```

```
                                        //ний
#define INT_ID_REG  BASE+2      //Причина прерывания
#define CONTROL     BASE+3      //Управляющее слово УАПП
#define MODEM               BASE+4      //Регистр управления состоянием
                                        //статических сигналов
#define STATUS              BASE+5      //Состояние УАПП
#define M_STATUS    BASE+6      //Состояние статических сигналов

//Определение основных констант

//Делители для различных скоростей передачи данных
//Скорость передачи — число после символов "B_":
#define B_110           1040
#define B_150           768
#define B_300           384
#define B_600           192
#define B_1200          96
#define B_2400          48
#define B_4800          24
#define B_9600          12
#define B_19200             6
#define B_38400             3
#define B_57600             2
#define B_115200            1

//Маски разрешенных прерываний:
#define DATA_REC    0x01        //Данные приняты
#define EMPTY_BUF   0x02        //Буфер передачи пуст
#define ERROR               0x04        //Ошибка при приеме данных
#define MODEM_INT   0x08        //Изменилось состояние
                                //статических сигналов

//Управляющие константы:
#define DIVISOR             0x80        //Бит управления доступом к
                                //регистрам записи делителя
#define BIT_5           0x00        //Длина посылки 5 бит
#define     BIT_6           0x01        //Длина посылки 6 бит
#define     BIT_7           0x02        //Длина посылки 7 бит
#define     BIT_8           0x03        //Длина посылки 8 бит

#define STOP_1              0x00        //1 стоповый бит
#define STOP_2              0x04        //2 стоповых бита
#define NOPARITY    0x00        //Нет контроля четности
#define EVEN            0x18        //Контроль четности
#define ODD                 0x08        //Контроль нечетности
#define FIXPARITY   0x20        //Фиксация бита четности
#define DTR                 0x01        //Управление линией DTR
#define RTS                 0x02        //Управление линией RTS
#define OUT1            0x04        //Управление линией OUT1
#define OUT2            0x08        //Управление линией OUT2
#define LOOPBACK    0x10        //Управление внутренним тестом
```

```
//Маски готовности:
#define DATA_IN      0x01        //Данные приняты
#define OVERRUN      0x02        //Переполнение
#define PARITY_ERR   0x04        //Ошибка четности
#define FRAME_ERR    0x08        //Ошибка формата посылки
#define DATA_OUT     0x20        //Готов к передаче

//Маски состояния статических сигналов:
#define CTS                  0x10        //Состояние линии CTS
#define DSR                  0x20        //Состояние линии DSR
#define RI                   0x40        //Состояние линии RI
#define DCD                  0x80        //Состояние линии DCD

//Определение функций:
void        init_ser(void);
int         out_sym(int symbol);
int         in_sym(int *symbol, timeout, *error_code);

//Конец файла SERIAL.H
```

Функция инициализации последовательного порта init_ser() в данном примере не требует параметров, поскольку (так чаще всего и бывает) заранее известен режим работы микросхемы.

Функция передачи байта out_sym() в случае успешного завершения возвращает 0, в противном случае — 1.

Функция приема байта in_sym() в случае успешного приема возвращает 0, в случае же неуспеха — код ошибки равный 1 если в течение ожидаемого времени данные не были приняты, 2 если байт данных был принят с ошибкой (байт состояния в этом случае присваивается переменной error_code).

Инициализируем последовательный порт для работы со скоростью 9600 бит/с, по опросу (все прерывания запрещены), статические сигналы управления не используются (применена трехпроводная линия связи), длина посылки 8 бит, стоповых битов 2, контроля четности нет. Следует обратить внимание на то, чтобы переменной BASE, было присвоено верное значение ДО вызова любой из описанных выше функций.

```
void init_ser(void) {
//Подготовка к записи делителя скорости передачи:
  outportb(CONTROL, DIVISOR);
//Запись младшего и старшего байт делителя:
  outportb(LOW_DIV, B_9600);
  outportb(HIGH_DIV, B_9600 >> 8);
//Запись управляющего слова, соответствующего выбранному //режиму:
  outportb(CONTROL, BIT_8+STOP_2+NOPARITY);
//Запрет прерываний:
  outportb(INT_REG, 0);
}
```

Теперь приведем текст функции передачи байта:

```
int out_sym(int symbol) {
```

```
long      timeout;
  timeout=clock();
//Ожидаем окончания передачи предыдущего байта,
//следя за временем передачи:
  do {
    if((clock() - timeout) == 3) return -1; //Время истекло
  } while((inportb(STATUS) & DATA_OUT) == 0);
//Предыдущий байт передан успешно, передаем дальше:
  outportb(OUT_REG, symbol);
  return 0;
}
```

Функция cllock(), описанная в файле TIME.H, возвращает значение системной переменной, которая увеличивается на 1 каждые 55 мс (по прерываниям от системного таймера). Таким образом, возврат по условию ((clock()-timeout) == 3) означает невозможность передать предыдущий символ за время, большее 110 мс, что с избытком превосходит время, необходимое для передачи байта даже при работе на скорости 110 бит/с. Такая ситуация может возникнуть если, например, отвалился провод, по которому на микросхему УАПП поступает тактовая частота, или она (микросхема) сгорела на работе да так, что бит DATA_OUT постоянно находится в состоянии 0.

Немногим сложнее выглядит функция приема байта:

```
int in_sym(int *symbol, timeout, *error_code) {
long                  tmp;
  tmp=clock();
//Ожидаем приема байта, следя за временем:
  do {
    if((clock() - tmp) >= timeout) return 1;     //Время истекло
    *error_code=inportb(STATUS);
  } while ((*error_code & DATA_IN) == 0);
//Может была ошибка при приеме ?
if((*error_code & (FRAME_ERR+OVERRUN)) != 0) {
//Тогда возвращаем соответствующий код:
    return 2;
  } else {
    *symbol=inportb(IN_REG);
     return 0;
  }
}
```

Для надежного приема байта отнюдь не следует безмерно увеличивать значение timeout (например, до одной недели), поскольку если всего лишь не подсоединен разъем RS-232C, выйти из программы окажется невозможно.

Напишем небольшую программу, которая будет проводить внутренний тест передачи-приема вводимых с клавиатуры символов. Параметром запуска будет служить номер последовательного порта 1 - COM1, 2-COM2. По умолчанию будем работать с портом COM1.

```
#include <stdio.h>
#include <time.h>
#include <stdlib.h>
#include "serial.h"

#define ESC  27

main(int argc, char *argv[]) {
int tmpint,                    //Переменная для всяких нужд
```

```
   symbol,                  //Символ для передачи
   symbol2,                 //Символ для приема
   error_code;              //Код ошибки для функции in_sym()

  if(argc == 1) {   //Есди нет параметров
   BASE=0x3f8;
  } else {                  //Если есть:
   sscanf(argv[1], "%d", tmpint);
   if(tmpint == 1) {
    BASE=0x3f8;
   } else {
    BASE=0x2f8;
   }
  }
//Инициализируем порт
  init_ser();
//Переводим его в режим внутреннего теста:
  outportb(MODEM, LOOPBACK);
//"Вечный" цикл:
  while(1) {
   symbol=getch();
   if(symbol == ESC) exit(0);
   if(out_sym(symbol) != 0) {
    printf("\nОшибка передачи !");
    exit(1);
   } else {
    if(in_sym(&symbol2, 10, &error_code) != 0) {
     printf("\nОшибка приема !");
     exit(2);
    } else {        //Т.е. нет ошибки приема
     printf("\nПередано: %02X Принято %02X", symbol, symbol2);
    }
   }
  } /* конец вечного цикла while */
} /* конец функции main */
```

4.6. Отладка контроллеров на базе однокристальной микроЭВМ

Как уже отмечалось в первой главе, особенность контроллеров, содержащих микроЭВМ, состоит в том, что, с одной стороны, их схемотехническое проектирование довольно просто (обычно используются стандартные схемы включения), а, с другой стороны, они требуют специальных средств для разработки и отладки программного обеспечения. Достоинством таких контроллеров является их высокая гибкость и универсальность, а недостатком (по сравнению с УС на жесткой логике) — малое быстродействие.

В процессе отладки рассматриваемых контроллеров можно выделить два этапа. Первый из них предполагает проверку функционирования аппаратной части. То есть в основном требуется проверить правильность разводки печатной платы и исправность использованных микросхем (схема соединения - стандартная). На втором этапе требуется проверить реальное функционирование всего контроллера в целом, включая и программное обеспечение однокристальной микроЭВМ. Здесь надо уже быть уверенным, что аппаратура работает нормально.

Следует отметить, что при отладке аппаратуры (на первом этапе) нельзя полностью

проверить исправность самой однокристальной микроЭВМ, так как проконтролировать правильность выполнения всех команд и всех возможных последовательностей команд чисто физически невозможно (как и для любого микропроцессора). Но обычно этого и не требуется. Если выяснится, что данная микросхема не выполняет должным образом требуемые программы, ее можно просто заменить.

Таким образом, отладка аппаратуры контроллера будет сводиться практически только к отладке элементов обрамления (внешние ОЗУ и ППЗУ, регистры, буферы, схема управления, внешние порты). При этом микросхему микроЭВМ надо удалить и эмулировать ее циклы обмена с помощью специальных отладочных средств. В качестве таких средств очень удобно использовать персональный компьютер и универсальный контроллер параллельного обмена информацией, описанный во второй главе. Эмуляция производится аналогично случаю системы статической отладки для УС, сопрягаемых с ISA (см. раздел 2.3.1). При этом внешние линии контроллера параллельного обмена с помощью кабеля присоединяются к выводам контактирующего устройства (панельки, колодки) микроЭВМ. После проведения полного цикла отладки аппаратуры можно вставлять обратно микросхему микроЭВМ и переходить к отладке программного обеспечения (точнее к совместной проверке аппаратуры и программ).

В принципе, отладку программного обеспечения можно проводить полностью на персональном компьютере, используя программную модель нашего устройства. Но в этом случае не стоит сильно удивляться, когда прекрасно работающая на модели программа будет вести себя совсем не так хорошо в реальности. Любая модель по определению ограничена. Поэтому гораздо разумнее проверять функционирование программ непосредственно на плате УС. Для этого совсем не надо каждый раз перезаписывать программное ППЗУ микроЭВМ (рассматриваем случай с внешним ППЗУ), достаточно использовать эмулятор ППЗУ, подключенный к компьютеру и соответствующие программные средства.

Идея эмулятора ППЗУ очень проста. Коды управляющих программ контроллера, написанных и оттранслированных с помощью персонального компьютера, мы записываем в ОЗУ, которое затем включается в схему контроллера вместо ППЗУ. После этого проверяется работа контроллера в целом. В случае необходимости программы очень легко поменять и снова перезагрузить их в ОЗУ. Полностью отлаженные на реальной рабочей схеме контроллера программы записывются в ППЗУ. Таким образом, процесс отладки существенно упрощается.

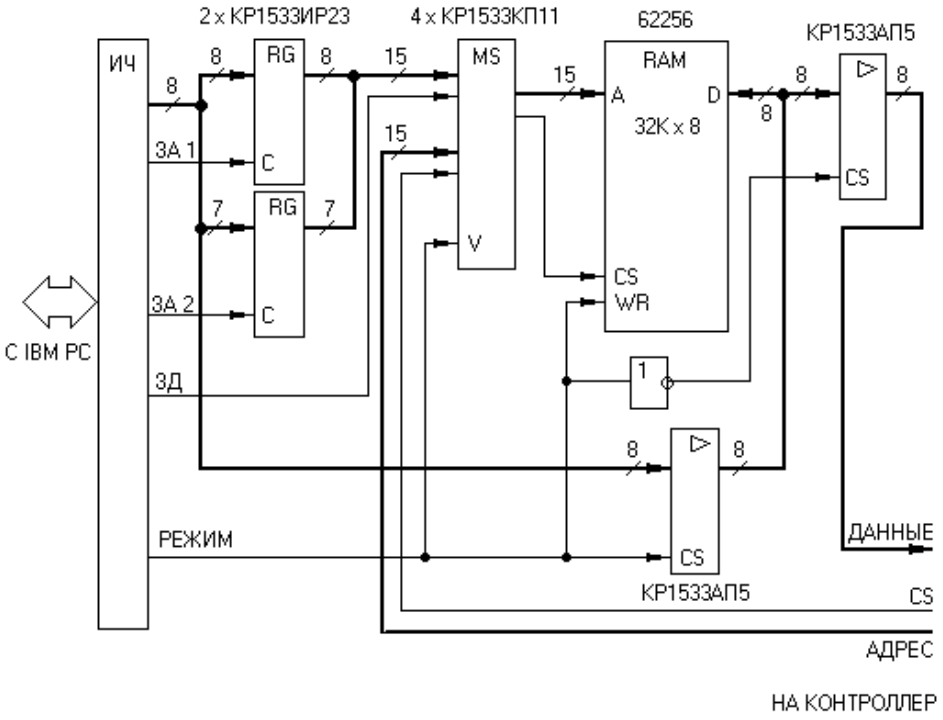


Рис. 2.14. Структурная схема эмулятора ППЗУ

Структура эмулятора (рис. 2.14) помимо собственно микросхемы ОЗУ объемом 32 К х 8 содержит интерфейсную часть (ИЧ),15-разрядный регистр адреса, 16-разрядный двухканальный мультиплексор адреса и сигнала выбора ОЗУ и два восьмиразрядных буфера данных. Схема работает в одном из двух возможных режимов (сигнал РЕЖИМ): записи информации из компьютера в ОЗУ и эмуляция ППЗУ.

В режиме записи информации ОЗУ находится в состоянии записи, мультиплексор пропускает на свои выходы код адреса ОЗУ с регистра адреса, нижний (по схеме) буфер данных открыт и пропускает на входы данных ОЗУ данные с ИЧ, верхний (выходной) буфер данных закрыт. Для записи каждой ячейки ОЗУ сначала в регистр адреса по сигналам ЗА1 и ЗА2 записывается адрес этой ячейки, а затем в нее производится запись данных по сигналу ЗД.

В режиме эмуляции ОЗУ переводится в режим чтения, мультиплексор пропускает адрес и сигнал CS с отлаживаемого контроллера, нижний буфер данных закрыт, а верхний (выходной) передает данные из ОЗУ на отлаживаемый контроллер. То есть ОЗУ выступает в качестве ППЗУ.

Сопряжение эмулятора с компьютером можно осуществлять различными способами: через ISA, через Centronics и даже через RS-232C (так как протокол обмена предельно прост, и скорость обмена с компьютером абсолютно некритична). Стоит отметить, что для отладки самой схемы эмулятора весьма полезной оказывается возможность проверки правильности информации, записанной в ОЗУ. Для реализации этого режима требуется использовать двунаправленный буфер данных для обмена с ИЧ. Но схема эмулятора при этом несколько усложнится, и протокол обмена с ней уже не будет настолько простым.

ЛИТЕРАТУРА

1. Руководство по архитектуре IBM PC AT / Ж.К. Голенкова, А.В. Заблоцкий, М.Л. Мархасин и др.; Под ред. М.Л. Мархасина. — Мн.: ООО "Консул", 1992 — 949 с.: ил.

2. Блохнин С.М. Шина ISA персонального компьютера IBM PC/AT. — М.: ПК "Сплайн", 1992 — 76 с.

3. Бычков Е.А. Архитектуры и интерфейсы персональных коипьютеров. — М.: Центр "СКС", 1993 — 152 с.: ил.

4. Фролов А.В., Фролов Г.В. Аппаратное обеспечение IBM PC: В 2-х ч. — М.: "ДИАЛОГ — МИФИ", 1992.

5. Интерфейсы систем обработки данных: Справочник / А.А. Мячев, В.Н. Степанов, В.К. Щербо; Под ред. А.А. Мячева. — М.: Радио и связь, 1989 — 416 с.: ил.

6. Мячев А.А. Персональные ЭВМ: Краткий энциклопедический справочник. — М.: Финансы и статистика, 1992 — 380 с.

7. Шевкопляс Б.В. Микропроцессорные структуры. Инженерные решения: Справочник. — 2-е изд. перераб. и доп. — М.: Радио и связь, 1990 — 512 с.: ил.

8. Сопряжение датчиков и устройств ввода данных с компьютерами IBM PC: Пер. с англ./ Под ред. У. Томпкинса, Дж. Уэбстера. — М.: Мир, 1992 — 592 с.: ил.

9. Логические ИС КР1533, КР1554: Справочник / И.И. Петровский, А.В. Прибыльский, А.А. Троян, В.С. Чувелев: В 2-х ч. — М.: ТОО "Бином", 1993.

10. Джордейн Р. Справочник программиста персональных компьютеров типа IBM PC, XT и AT: Пер. с англ./ Предисл. Н.В. Гайского. — М.: Финансы и статистика, 1991 — 544 с.: ил.

11. Уильямс Г.Б. Отладка микропроцессорных систем: Пер. с англ. — М.: Энергоатомиздат, 1988 — 253 с.: ил.

12. Нортон П. Программно-аппаратная организация IBM PC: Пер. с англ. М.: Радио и связь, 1991 — 416 с.: ил.

13. Нортон П. Персональный компьютер фирмы IBM и операционная система MS DOS: Пер. с англ. М.: Радио и связь, 1991 — 416 с.: ил.

14. Standard IBM PC. Справочник. Устройство, установка, техническое обслуживание и ремонт персональных компьютеров/ Составитель Г. Карпов. — Кишинев.: ВИРТ, 1991 — 182 с.: ил.

15. Руководство по поиску неисправностей и ремонту компьютеров IBM PC. — М.: Радио и связь, 1992 — 192 с.: ил.

16. Борзенко А.Е. IBM PC: устройство, ремонт, модернизация. — М.: ТОО фирма "КомпьютерПресс", 1995. — 298 с.: ил.

17. Бродин В.Б., Шагурин И.И. Микропроцессор i486. Архитектура, программирование, интерфейс. — М.: "ДИАЛОГ-МИФИ", 1993. — 240 с.

18. Овчинников В.В., Рыбкин И.И. Техническая база интерфейсов локальных вычислительных сетей. — М.: Радио и связь, 1989 — 272 с.: ил.

19. Борзенко А.Е. Шина EISA// КомпьютерПресс. — 1992. — №8. — с. 3 — 10.

20. Бородин С.М., Новиков Ю.В. Модуль логического анализатора для контрольно-измерительных систем на базе микроЭВМ// Микропроцессорные средства и системы. — 1987. — N 1. — с. 67 — 68.

21. Новиков Ю.В. Универсальный параллельный интерфейс для модульных микропроцессорных систем измерения, контроля и управления// Микропроцессорные средства и системы. — 1989. — N 6. — с. 71 — 72.

22. Новиков Ю.В. Функциональные модули контрольно-измерительных систем на базе микроЭВМ// Микропроцессорные средства и системы. — 1990 . — N 3. — с. 75 — 77.

Приложение 1
Габаритные размеры платы ISA

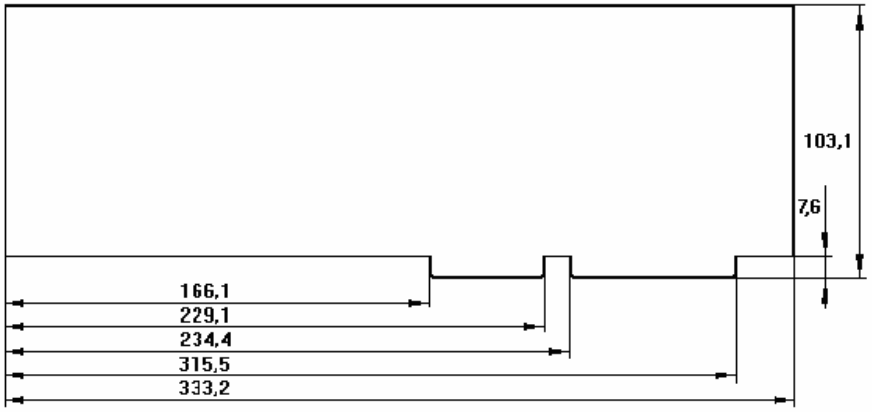


Рис. П.1. Размеры печатной платы ISA (все размеры в миллиметрах)

На рис П.1 приведены габаритные размеры платы ISA, а также размеры всех разъемов и вырезов платы. Отметим, что на рисунке указана только макимальная длина платы, но эта длина может изменяться и ограничена снизу только размерами одного разъема (правого по рисунку) для 8-битных устройств или размерами двух разъемов для 16-битных устройств.

Расстояния между центрами печатных проводников контактов разъемов должны составлять 2,54 мм.

Платы должны изготавливаться из фольгированного стеклотекстолита толщиной 1,6 мм (отклонение не больше 0,2 мм). Коробление платы не должно превышать 1,3 мм на всей длине платы. Максимальная высота компонентов на собранной плате не должна превышать 10 мм.

Приложение 2
Другие интерфейсы компьютера типа IBM PC

Системная магистраль EISA

Магистраль EISA (Extended Industry Standard Architecture) была предложена в 1989 году рядом наиболее крупных производителей компьютеров типа IBM PC (AST, Compaq, Epson, Hewlett-Packard, NEC, Olivetti, Tandy, Wise и Zenith). Необходимость разработки такой магистрали стала ясной, когда выяснилось, что быстродействие магистрали ISA становится сдерживающим фактором на пути повышения производительности компьютеров на базе новых процессоров (i80386, i486 и т.д.). Компьютеры нового поколения должны были обеспечивать 32-разрядный обмен, в том числе, в режиме ПДП, автоматическую конфигурацию системы и некоторые другие возможности, недоступные для ISA. Фирма IBM предлагала для этих целей свою магистраль MCA (Micro Channel Architecture), которая однако была полностью несовместима (как по сигналам, так и по разъему) с ISA, что предполагало полный отказ от всех изготавливавшихся ранее плат расширения.

EISA явилась не революционным, как MCA, а эволюционным шагом в развитии архитектуры, что обеспечило ей гораздо больше шансов на развитие. Она, с одной стороны, имела все преимущества высокопроизводительной 32-разрядной шины, а с

другой стороны, была полностью совместима с ISA "сверху вниз" и не требовала перехода на совершенно новую элементную базу. Вообще говоря, новую магистраль предполагалось использовать, в первую очередь, в высокоскоростных файл-серверах, но ее достоинства обеспечили ей более широкий спектр применения.

Разработчики новой магистрали позаботились не только об иформационной и электрической, но и о конструктивной совместимости с ISA. Разъем EISA состоит из двух рядов контактов, один из которых (верхний) предназначен для сигналов ISA, а другой (нижний) — для дополнительных сигналов EISA. Поэтому в EISA-разъемы можно вставлять также 8 или 16-разрядные ISA-платы.

EISA имеет 32 разряда адресной шины, что позволяет адресовать до 4 Гбайт памяти. Предельная скорость передачи информации по ней в специально предусмотренном так называемом пакетном режиме может достигать 33 Мбайт/с. На магистрали может находиться несколько задатчиков, включая, конечно, центральный процессор, контроллер ПДП, контроллер регенерации динамической памяти. Управление предоставлением магистрали централизованно выполняется специальным арбитром по циклическому принципу. Для арбитража используются особые линии магистрали, индивидуальные для каждого разъема расширения (всего таких разъемов не может быть более 15).

По сравнению с ISA в EISA добавлено несколько новых сигналов:

BE0 ... BE3 — разрешение байтов;

M/IO — память/устройство ввода/вывода — признак типа цикла обмена;

-START — идентификация начала цикла магистрали;

-CMD — разрешение управления временной диаграммой цикла магистрали;

-MS BURST — признак групповой (пакетной) передачи от задатчика;

-SL BURST — признак групповой (пакетной) передачи от исполнителя;

-EX32, EX16 — признак 32- и 16-разрядных данных;

EXRDY — идентификация окончания цикла магистрали текущим задатчиком;

-MREQ k — признак запроса на захват магистрали от k-го потенциального задатчика (возможно объединение запросов по ИЛИ). Сигнал индивидуален для каждого из 15 разъемов расширения магистрали;

-MACK k — признак разрешения на захват магистрали k-му задатчику. Сигнал индивидуален для каждого из 15 разъемов расширения магистрали;

D16 ... D31 — дополнительные разряды данных;

LA2 ... LA16, LA24 ... LA31 — дополнительные разряды адреса (не фиксируемые до конца цикла, как и LA17 ... LA23 в ISA).

Распределение дополнительных сигналов по нижней части магистрального разъема EISA представлено в табл. П.1. Специальные механические ключи, устанавливаемые в нижней части разъема, обеспечивают возможность правильного использования ISA- и EISA-совместимых модулей.

Контакт	Сигнал		Контакт	Сигнал	
A1	-CMD		B1	GND	
A2	-START		B2	+5 В	
A3	EXRDY		B3	+5 В	
A4	EX32		B4	X	
A5	GND		B5	X	
A6	Kod		B6	Kod	
A7	EX16		B7	X	
A8	-SL BURST		B8	X	
A9	-MS BURST		B9	+12 В	
A10	W/R		B10	M/IO	
A11	GND		B11	LOCK	
A12	RES		B12	RES	
A13	RES		B13	GND	
A14	RES		B14	RES	
A15	GND		B15	BE3	
A16	Kod		B16	Kod	
A17	BE1		B17	BE2	
A18	LA31		B18	BE0	
A19	GND		B19	GND	
A20	LA30		B20	+5 В	
A21	LA28		B21	LA29	
A22	LA27		B22	GND	
A23	LA25		B23	LA26	
A24	GND		B24	LA24	
A25	Kod		B25	Kod	
A26	LA15		B26	LA16	
A27	LA13		B27	LA14	
A28	LA12		B28	+5 В	
A29	LA11		B29	+5 В	
A30	GND		B30	GND	
A31	LA9		B31	LA10	

Табл. П.1. Дополнительные контакты разъема EISA (продолжение на следующей странице)

Контакт	Сигнал		Контакт	Сигнал	
C1	LA7		D1	LA8	
C2	GND		D2	LA6	
C3	LA4		D3	LA5	
C4	LA3		D4	+5 В	
C5	GND		D5	LA2	
C6	Kod		D6	Kod	
C7	D17		D7	D16	
C8	D19		D8	D18	
C9	D20		D9	GND	
C10	D22		D10	D21	
C11	GND		D11	D23	
C12	D25		D12	D24	
C13	D26		D13	GND	
C14	D28		D14	D27	
C15	Kod		D15	Kod	
C16	GND		D16	D29	
C17	D30		D17	+5 В	
C18	D31		D18	+5 В	
C19	MREQ k*		D19	MACK k*	

Табл. П.1. (продолжение) Дополнительные контакты разъема EISA
(Kod — ключ, RES — резерв, X — географическая адресация,
** — раздельные линии для каждого разъема магистрали)*

Из компьютеров, реализованных на базе магистрали EISA, пожалуй, наиболее известны компьютеры серии Vectra фирмы Hewlett Packard.

Системная магистраль MCA

Магистраль MCA (Micro Channel Architecture) была предложена фирмой IBM в 1987 году для нового семейства персональных компьютеров PS/2 (Personal System/2). Новая архитектура была ориентирована на операционные системы OS/2 и UNIX с их возможностями многозадачного и многопользовательского режимов. С самого начала политика фирмы IBM по внедрению новой архитектуры резко отличалась от политики по выпуску компьютеров типа IBM PC. IBM решила не выпускать производство новых компьютеров из своих рук, поэтому все спецификации MCA были запатентованы, и возможности других производителей IBM-совместимых компьютеров и плат расширения для них были существенно ограничены. Даже конструктивное решение MCA-плат расширения было выбрано таким образом, чтобы их можно было выполнить только на специализированных БИС (платы на универсальных ИС занимают больший объем). Никакой совместимости с компьютерами типа IBM PC не предусматривалось. Это приносило IBM все выгоды монополиста, но затрудняло завоевание рынка новым стандартом. Сейчас, правда, компьютеры на базе MCA выпускают и некоторые другие фирмы.

Основным достоинством MCA по сравнению с ISA было, конечно, увеличение разрядности шины данных до 32 и даже 64 (при мультиплексированном использовании шины адреса), а также доведение разрядности шины адреса до 32 разрядов, что позволяет адресовать до 4 Гбайт памяти. Максимально возможная скорость передачи данных достигает 160 Мбайт/с.

Как и в магистрали EISA, в MCA предусмотрена возможность включения многих задатчиков. Но арбитраж в MCA не централизованный, а распределенный, и приоритеты могут более гибко устанавливаться программным путем.

В принципе, MCA имеет очень большие возможности для развития, так как она по своей природе не зависит от типа центрального процессора и от его временных характеристик, являясь полностью асинхронной.

В отличие от ISA-компьютеров, здесь отсутствуют контроллеры прямого доступа к памяти, хотя и предусмотрены восемь каналов ПДП, которые могут использоваться задатчиками магистрали. Для увеличения скорости передачи в режиме ПДП используется специальный блочный режим (burst mode), при котором за единицу времени передается целый блок информации.

В магистрали MCA предусмотрена и автоматическая конфигурация системы. При этом пользователь может изменять и назначать приоритеты различных устройств.

Стоит отметить, что полное использование преимуществ MCA возможно только при использовании в компьютере интеллектуальных периферийных устройств, способных выступать задатчиками магистрали. Таких устройств пока выпускается недостаточно.

Главный недостаток магистралиMCA — это ее полная несовместимость с ISA, что на данном этапе ограничивает ее распространение. Слишком много фирм во всем мире связало свою деятельность с производством компьютеров типа IBM PC. Платы расширения для ISA стоят сейчас очень дешево, к тому же их номенклатура чрезвычайно широка. Поэтому многие пользователи делают выбор все-таки в пользу IBM PC. Но что

будет дальше, пока не совсем ясно.

Системная магисталь NuBus

Магистраль NuBus была разработана в 1979 году рабочей группой M.I.T. совместно с фирмой Western Digital и стандартизована ANSI/IEEE в 1987 году. Она применяется в компьютерах Macintosh и системах Next, а также в некоторых разработках фирмы Texas Instruments. Как видим, данная магистраль не имеет прямого отношения к теме данной книги, но несколько слов о ней все-таки стоит сказать.

Как и рассмотренные выше магистрали, NuBus имеет 32 разряда данных, но отличается от них своей простотой. Она содержит всего 96 сигналов (в EISA 188 сигналов, в MCA — 178), из которых 22 линии земли и 23 линии питания. Этого оказалось достаточно для большинства задач.

NuBus имеет 32 разряда адреса и позволяет адресовать 4 Гбайта памяти. Макимальная скорость передачи данных достигает 37,5 Мбайт/с. Тактовая частота магистрали — 10 Мгц.

В NuBus используется мультиплексирование адреса и данных. Предусмотрена автоматическая конфигурация. Возможно использование нескольких задатчиков магистрали с децентрализованным арбитражом. Имеется режим блочной передачи данных.

К недостаткам NuBus можно отнести слабые возможности режима ПДП, сложный метод обработки прерываний (предусмотрен всего один сигнал запроса прерывания и программный опрос потенциальных источников прерываний).

Локальные шины VLB и PCI

Для ускорения работы компьютеров типа IBM PC с новыми высокопроизводительными процессорами многие разработчики пошли по пути использования так называемых локальных шин (магистралей) в дополнение к системной магистрали ISA. Это позволило не только организовать обмен процессора с памятью, минуя сравнительно медленную ISA, но и подключать к компьютеру 32-разрядные платы расширения, что значительно ускоряет обмен с видеоконтроллерамим, контроллерами накопителей и другими устройствами сопряжения.

Наибольшее распространение в настоящее время получили две локальные шины — VLB (Video Local Bus), предложенная ассоциацией VESA (Video Electronics Standards Association), и PCI (Peripheral Component Interconnect), разработанная фирмой Intel. Обе эти шины позволяют организовать обмен с тактовой частотой до 33 Мгц и обе они используют разъемы типа MCA. Но есть между ними и существенные различия.

VLB представляет собой, по сути, расширение шины процессора без промежуточных буферов. Это сразу резко ограничивает ее нагрузочную способность. На материнских платах обычно расположено всего 2-3 разъема VLB (112 контактов). Чаще всего они используются для подключения видеоадаптеров, контроллеров накопителей и сетевых контроллеров. VLB имеет 32-разрядную шину данных и 30-разрядную шину адреса. Максимальная скорость передачи теоретически может достигать 130 Мбайт/с. Арбитр магистрали не предусмотрен. Достоинство VLB — ее простота и, как следствие, низкая стоимость. В настоящее время готовится вторая версия VLB, использующая 64 разряда данных и тактовую частоту 40-50 Мгц.

PCI — это принципиально новая магистраль, близкая по своим характеристикам к системным. Она не привязана к типу процессора, как VLB, и имеет гораздо большую нагрузочную способность (возможно подключение до 10 устройств, но на материнскую плату обычно устанавливается только 3 разъема расширения). Используются 124-

контактные (для 32 разрядов данных) или 188-контактные (для 64 разрядов данных) разъемы. Тактовая частота PCI фиксирована и составляет 33 Мгц. Максимальная теоретически возможная скорость обмена достигает 132 или 264 Мбайт/с для 32 и 64 разрядов данных соответственно. Предусмотрена возможность включения плат с напряжением питания как 5 В, так и 3,3 В (в раздельные разъемы). На магистрали предусмотрен арбитраж. Возможна автоконфигурация.

Стоит отметить, что изготовители персональных компьютеров часто устанавливают на материнские платы несколько магистралей.
Встречаются, например, комбинации ISA+EISA+VLB или ISA+EISA+PCI или даже все четыре упомянутые здесь магистрали одновременно. Это дает пользователю больше свободы при создании нужной ему конфигурации.

Интерфейс PCMCIA

Стандарт PCMCIA (Personal Computer Memory Card Inter-national Association) был предложен в 1990 году для портативных компютеров (notebook) и используется для подключения к ним различных внешних устройств: модулей памяти (в том числе флэш-памяти), модемов и факс-модемов, сетевых контроллеров, дополнительных накопителей и т.д. Устройства, предназначенные для PCMCIA, отличаются очень малыми габаритами (с обычную кредитную карточку) и довольно высокой по сравнению с другими аналогичными устройствами стоимостью. Сейчас уже выпускаются PCMCIA-адаптеры для обычных (настольных) компьютеров.

Если первая версия PCMCIA была предназначена только для модулей памяти, то вторая (1991 год) позволяла включать устройства ввода/вывода и поддерживала два напряжения питания (5 В и 3,3 В). Для подключения PCMCIA-карт используется 68-контактный разъем.

Разрядность передаваемых данных — 16, количество разрядов адреса — 26, что позволяет адресовать до 64 Мбайт памяти. Стандарт определяет три различных длины контактов разъема для обеспечения правильной последовательности подачи напряжения питания при подключении и отключении карты во время работы компьютера.

Интерфейс для периферии SCSI

Помимо уже упомянутых интерфейсов для подключения устройств сопряжения, в принципе, могут быть использованы и другие интерфейсы, например, SCSI, который был разработан для организации обмена с периферийными устройствами, в частности, с накопителями на магнитных и оптических дисках. Интерфейс SCSI был стандартизован ANSI в 1986 году и поддерживается фирмой IBM для компьютеров PS/2.

SCSI позволяет подключать к компьютеру до 7 внешних устройств через один адаптер. Скорость 8-разрядной синхронной передачи по SCSI может достигать 3-4 Мбайт/с. Длина линии связи может достигать 6 м и даже 25 м при дифференциальной передаче.

Особенность SCSI — это то, что устройства на шине имеют не физические, а логические адреса. Это обеспечивает высокую гибкость взаимодействия. Предусмотрена также возможность присоединять и отсоединять устройства на уровне контроллера.

В настоящее время уже имеется новая версия этого стандарта — SCSI-II. Она обеспечивает уже не 8-, а 16- и 32-разрядную шину данных. В результате оптимизации временных соотношений обмена достигается скорость передачи данных до 40 Мбайт/с, то

есть выходит на уровень системных 32-разрядных магистралей.

Хотя основное назначение интерфейса SCSI — сопряжение с дисковыми накопителями, его характеристики позволяют рассматривать его, как достаточно универсальную внешнюю шину компьютера.

Так что возможности разработчика устройств сопряжения отнюдь не ограничиваются тремя интерфесами, подробно рассмотренными в данной книге.

Приложение 3
Микросхемы серий КР1533 и КР1554 и их аналоги

Серия 1533	Аналог SN74		Серия 1533	Аналог SN74		Серия 1533	Аналог SN74	
ЛА1	ALS20		ЛЛ4	ALS1032		ИР13	198	
ЛА2	ALS30		ЛН8	ALS1004		ИР15	LS173	
ЛА3	ALS00		ЛН10	ALS1005		ИР16	LS295	
ЛА4	ALS10		ЛП8	LS125		ИР22	ALS373	
ЛА7	ALS22		ЛП16	ALS1034		ИР23	ALS374	
ЛА8	ALS01		ЛП17	ALS1035		ИР24	ALS299	
ЛА9	ALS03		ТВ6	LS107		ИР26	LS670	
ЛА10	ALS12		ТВ9	ALS112		ИР27	LS377	
ЛЕ1	ALS02		ТВ10	ALS113		ИР29	ALS323	
ЛЕ4	ALS27		ТВ11	ALS114		ИР30	ALS259	
ЛИ1	ALS08		ТВ15	ALS109		ИР32	LS170	
ЛИ2	ALS09		ТЛ2	LS14		ИР33	ALS573	
ЛИ3	ALS11		ТМ2	ALS74		ИР34	ALS873	
ЛИ4	ALS15		ТМ7	LS75		ИР35	ALS273	
ЛИ6	ALS21		ТМ8	ALS175		ИР37	ALS574	
ЛЛ1	ALS32		ТМ9	ALS174		ИР38	ALS874	
ЛН1	ALS04		ТР2	LS279		КП2	ALS153	
ЛН2	ALS05		ИЕ2	LS90		КП7	ALS151	
ЛП5	ALS86		ИЕ5	LS93		КП11А	ALS257	
ЛП12	ALS136		ИЕ6	ALS192		КП12	ALS253	
ЛР4	LS55		ИЕ7	ALS193		КП13	LS298	
ЛР11	LS51		ИЕ9	ALS160		КП14А	ALS258	
ЛР13	LS54		ИЕ10	ALS161		КП15	ALS251	
ЛА21	ALS1000		ИЕ11	ALS162		КП16	ALS157	
ЛА22	ALS1020		ИЕ12	ALS190		КП17	ALS353	
ЛА23	ALS1003		ИЕ13	ALS191		КП18	ALS158	
ЛА24	ALS1010		ИЕ18	ALS163		КП19	ALS352	
ЛЕ10	ALS1002		ИЕ19	LS393		ИД3	LS154	
ЛЕ11	ALS33		ИР8	ALS164		ИД4	LS155	
ЛИ8	ALS1008		ИР9	ALS165		ИД7	ALS138	
ЛИ10	ALS1011		ИР10	ALS166		ИД14	ALS139	
Серия 1533	Аналог SN74		Серия 1533	Аналог SN74		Серия 1533	Аналог SN74	
АП3	ALS240		АП15	ALS466		ИП3	LS181	
АП4	ALS241		АП16	ALS643		ИП4	S182	
АП5	ALS244		ИП6	ALS242		ИП5	LS280	
АП6	ALS245		ИП7	ALS243		СП1	LS85	
АП9	ALS640		ЛН7	ALS368		ЛП3	-	
АП14	ALS465		АГ3	LS123		ИР39	-	

Серия 1554	Аналог		Серия 1554	Аналог		Серия 1554	Аналог	
ЛА1	74AC20		АП10	74AC640		ИР46	НС4015	
ЛА3	74AC00		АП20	74AC646		ИР47	НС4006	
ЛА4	74AC10		КП2	74AC153		ИР51	НС4035	
ЛЕ1	74AC02		КП11	74AC257		ТВ9	74AC112	
ЛЕ4	74AC27		КП12	74AC253		ТВ15	74AC109	
ЛИ1	74AC08		КП14	74AC258		ТМ2	74AC74	
ЛИ6	74AC21		КП16	74AC157		ТМ8	74AC175	
ЛИ9	74AC34		КП18	74AC158		ТМ9	74AC174	
ЛЛ1	74AC32		ИР22	74AC373		ИЕ6	74AC192	
ЛН1	74AC04		ИР23	74AC374		ИЕ7	74AC193	
ЛП5	74AC86		ИР24	74AC299		ИЕ10	74AC161	
АП3	74AC240		ИР29	74AC323		ИЕ18	74AC163	
АП4	74AC241		ИР35	74AC273		ИЕ23	НС4520	
АП5	74AC244		ИР40	74AC533		ИД14	74AC139	
АП6	74AC245		ИР41	74AC534		ИП5	74AC280	

Приложение 4

Форматы обмена с приемопередатчиком RS-232C

Контроллер параллельного обмена (универсальный асинхронный приемопередатчик, УАПП), входящий в состав персонального компьютера, реализует следующие функции:

— преобразование параллельного кода в последовательный при передаче и обратное преобразование при приеме;

— формирование стартового, стопового битов и бита четности при передаче и контроль их правильности при приеме;

— прием и передача данных на заданной скорости;

— формирование и контроль состояния сигналов интерфейса RS-232C.

УАПП может быть выполнен на специальной микросхеме (обычно i8250 или 16550A) или входить в состав БИС вместе с другими контроллерами, но все форматы обмена с ним сохраняются неизменными.

Обычно в состав компьютера входят два последовательных порта, обозначаемых COM1 (адреса 3F8 ... 3FFh, прерывание IRQ4) и COM2 (адреса 2F8 ... 2FFh, прерывание IRQ3).

Рассмотрим назначение отдельных битов, записывемых по этим адресам и читаемых из этих адресов. Но сначала отметим, что назначение битов портов 3F8 и 3F9 зависит от значения специального бита управления, записываемого в 7 разряде в порт 3FB (здесь и далее рассматриваем COM1, имея в виду, что для COM2 все делается аналогично).

Порт 3F8

При нулевом значении управляющего бита этот порт служит для записи в него передаваемого байта данных и чтения из него принимаемого байта. При единичном значении управляющего бита этот порт используется для записи в него младшего байта кода делителя частоты тактового генератора, определяющего скорость передачи и приема. Связь этого кода и скорости передачи следующая:

Шестнадцатеричный код частоты	Скорость передачи, бит/с	Шестнадцатеричный код частоты	Скорость передачи, бит/с
0410	110	0018	4800
0300	150	000C	9600
0180	300	0006	19200
00C0	600	0003	38400
0060	1200	0002	57600
0030	2400	0001	115200

Порт 3F9

При значении управляющего бита, равном единице, этот порт используется для записи старшего байта кода делителя частоты. При нулевом значении управляющего бита этот порт используется для управления прерываниями. При этом он имеет следующий формат:

Номер бита	Назначение бита
0	1 — разрешение прерывания по окончании приема данных, 0 — запрещение прерывания
1	1 — разрешение прерывания по окончании передачи данных, 0 — запрещение прерывания
2	1 — разрешение прерывания при обнаружении сбоя на линии, 0 — запрещение прерывания
3	1 — разрешение прерывания по изменению входных управляющих сигналов RS-232C, 0 — запрещение прерывания
4, 5, 6, 7	не используются

Порт 3FA

Это регистр идентификации прерывания. Используется только для чтения. Его содержимое указывает на причину прерывания.

Формат регистра следующий:

Номер бита	Назначение бита
0	1 — нет прерываний, требующих обслуживания, 0 — есть прерывания
1, 2	00 — переполнение приемника, ошибка четности или формата данных при приеме, сброс — по чтению из 3FD; 01 — данные переданы, сброс — по записи в 3F8; 10 — данные приняты и доступны для чтения, сброс — по чтению из 3F8; 11 — изменение состояния входных управляющих сигналов RS-232C, сброс — по чтению из 3FE
3 ... 7	не используются

Порт 3FB

Это управляющий регистр, доступный по чтению и записи. Его формат следующий:

Номер бита	Назначение бита
0, 1	Количество бит передаваемых данных: 00 — 5 бит, 10 — 6 бит, 01 — 7 бит, 11 — 8 бит
2	Количество стоповых битов: 0 — 1 бит, 1 — 2 бита
3, 4	Контроль четности: 0X — контроль четности не используется, 10 — контроль на нечетность, 11 — контроль на четность

5	Задание контрольного бита: 1 — контрольный бит всегда равен 0 (если выбран кнтроль на четность) или всегда равен 1 (если выбран контроль на нечетность)
6	1 — постоянная передача нуля, 0 — нормальная передача символов
7	Управляющий бит для выбора назначения портов 3F8 и 3F9

Порт 3FC

Данный порт используется для управления модемом. Управляет состоянием управляющих линий интерфейса RS-232C. Применяется довольно редко. Его формат следующий:

Номер бита	Назначение бита
0	Состояние линии DTR
1	Состояние линии RTS
2	Состояние выходного сигнала УАПП OUT1
3	Состояние выходного сигнала УАПП OUT2
4	Режим работы УАПП: 0 — рабочий, 1 — диагностический
5, 6, 7	Не используются

Порт 3FD

Это регистр состояния линии. Его формат следующий:

Номер бита	Назначение бита
0	1 — данные получены и готовы для чтения, флаг сбрасывается при чтении данных
1	1 — ошибка переполнения при приеме (принят новый байт раньше, чем прочитан предыдущий, преды-дущий байт теряется)
2	1 — ошибка четности при приеме
3	1 — ошибка синхронизации (не принята стоповая посылка)
4	1 — обнаружен запрос на прерывание передачи (постоянная передача нуля)
5	1 — буферный регистр передачи пуст можно записывать следующий передаваемый байт
6	1 — регистр сдвига передатчика пуст, передача закончена
7	1 — тайм-аут

Порт 3FE

Это регистр состояния модема. Используется редко. Его формат следующий:

Номер бита	Назначение бита
0	Линия CTS изменила состояние после предыдущего чтения из регистра состояния модема
1	Линия DSR изменила состояние
2	Линия RI изменила состояние
3	Линия DCD изменила состояние
4	Состояние линии CTS
5	Состояние линии DSR
6	Состояние линии RI
7	Состояние линии DCD

Для передачи данных необходимо записать их по адресу 3F8 (предварительно надо

убедиться, что буферный регистр передатчика пуст). Принятые данные читаются из адреса 3F8 (предварительно надо убедиться, что данные приняты). УАПП обеспечивает дуплексный обмен данными, то есть возможно одновременно передавать и принимать данные. Но все параметры обмена (скорость, формат знака и т.д.) для приема и для передачи должны быть одинаковыми.

Для инициализации УАПП необходимо проделать следующее:

— записать по адресу 3FB управляющий байт с единицей в 7 бите;
— записать код делителя частоты по адресам 3F8 и 3F9;
— записать по адресу 3FB управляющий байт с нулем в 7 бите и с требуемыми значениями остальных битов;
— записать управляющий байт по адресу 3F9;
— записать управляющий байт по адресу 3FC.

Оглавление