«Работа с внешними устройствами при помощи портов ввода-вывода»

Тигран Калайджян

Цель работы
Разработать класс программных и аппаратных схем, предназначенных для: управления внешними приборами с помощью персонального компьютера (ПК), совершения измерений различных физических величин с помощью ПК, управления персональным компьютером пультами дистанционного управления.

Задачи работы
Изучить методы программирования портов ввода\вывода и разработать собственные схемы управления внешними приборами и схемы управления ПК с помощью пультов дистанционного управления.

Введение
ПК включает в себя целый комплекс компонентов для обработки и передачи данных, необходимый для создания измерительной лаборатории. Наличие микропроцессора и встроенных АЦП упрощает процесс проектирования аппаратных схем управления и сокращает время от создания принципиальной схемы до готового продукта.
В последние годы приобрела популярность тема автоматического управления бытовыми приборами и как готовый коммерческий продукт так называемый «Интеллектуальный дом», управление которым выполняет некоторое программируемое устройство. Интересно, что этим «программируемым устройством» не является ПК, хотя именно этот вариант является наиболее универсальным. Введение дополнительных опций (скажем, голосовое управление светом в квартире) в систему с ПК потребует лишь замену программной части и установку довольно простого внешнего прибора, не включающего в себя дополнительные средства оцифровки или микропроцессора. В данном проекте рассматриваются приборы такого типа, а также программы для обработки данных от таких приборов.

Принятые ограничения
Определимся с аппаратной реализацией проекта. Внешние устройства можно подключать как внутренним способом (ISA, PCI и др.), так и внешним (COM, LPT, USB, GAME и др.) Остановимся на втором варианте, так как это частично избавит нас от риска вывести из строя внутренние компоненты и обеспечит простоту подключения. Мы будем использовать LPT- и Game-порты. USB и COM не берутся или рассматриваются частично из-за сложности используемых протоколов.

Описанная программная часть рассчитана на операционные системы MS-DOS и MS-Windows9x. Для достижения того же результата для систем Windows2000\XP\NT, Unix и прочих следует использовать принципиально отличающиеся от предложенных методы программирования, связанные, например, с разработкой драйверов, что выходит за рамки данного проекта.

LPT-порт

Это именно тот порт, который чаще всего используется принтером, хотя в дальнейшем можно видеть, что это только узкая область его применения.
Итак, рассмотрим внешний вид порта:

Это стандартный разъем DB-25 на 25 выводов типа «мама». Разберемся с распайкой порта:

№ контакта	Назначение	Направление	Адрес порта	Бит
1	*Strobe*	*out(in)*	*37Ah*	*0*
2	*Data 0*	*Out*	*378h*	*0*
3	*Data 1*	*Out*	*378h*	*1*
4	*Data 2*	*Out*	*378h*	*2*
5	*Data 3*	*Out*	*378h*	*3*
6	*Data 4*	*Out*	*378h*	*4*
7	*Data 5*	*Out*	*378h*	*5*
8	*Data 6*	*Out*	*378h*	*6*
9	*Data 7*	*Out*	*378h*	*7*
10	*Ack*	*In*	*379h*	*6*
11	*Busy*	*In*	*379h*	*7*
12	*Paper Out*	*In*	*379h*	*5*
13	*Select*	*In*	*379h*	*4*
14	*Autofeed*	*out(in)*	*37Ah*	*1*
15	*Error*	*In*	*379h*	*3*
16	*INIT*	*out(in)*	*37Ah*	*2*
17	*Select IN*	*out(in)*	*37Ah*	*3*
18-25	*Общий*	*-*	*-*	*-*

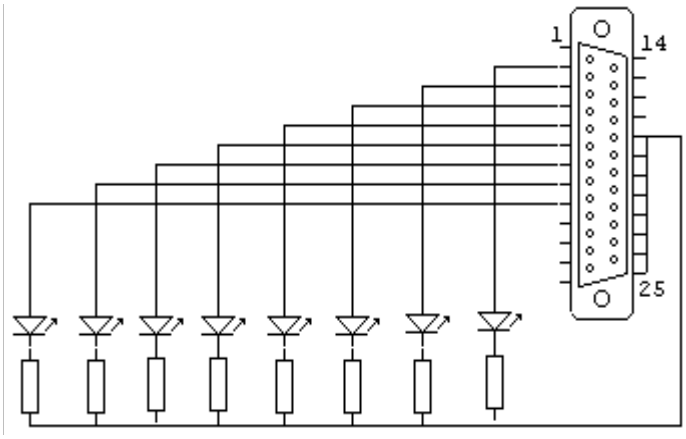
Рассмотрим по-подробнее:
Контакт №1: Строб - нужен для проверки подключения принтера.
Контакты № 2-9, 14, 16-17 – вывод информации (на каждый вывод по биту).
Контакты № 10-13,15 – ввод информации.
Контакты №18-25 – минус или «земля». Практически всегда соединен с корпусом компьютера.

В каком виде представляется информация на выходе?
При посылке какой-либо информации в порт, на линиях d0-d7 появится набор сигналов, т.е. распределение напряжений низкого уровня и высокого уровня, соответствующих логическому нулю или единице.

Лог. 1 → 3,5 В
Лог. 0 → 0В

Напряжение останется на выводах разъёма до тех пор, пока туда не будет переслано другое число или не будет выключен компьютер.

Определить, что было послано в порт, можно с помощью классической схемы на светодиодах:



Выводы 18-25 можно не замыкать между собой, резисторы можно подключить к любому из них. Все резисторы на схеме по 470 ом.

Здесь нужно учесть один довольно важный момент: следует ограничивать токи если следует использовать элементы с малым сопротивлением, потому как сила тока не должна превышать 2мА, а напряжение между сигнальным выводом и землей не более 2,6 В.

Ввод в порт данных происходит следующим образом: следует замкнуть нужный(-ые) вход(-ы) с «землей», тогда по соединению потечёт ток и бит, соответствующий данному каналу, установится в единицу (или в ноль, если вход инвертированный).

Универсальные схемы управления

Приведём схемы, которые можно использовать для решения практически любых задач проектирования схем управления. Схемы позволяют замыкать\размыкать электрическую цепь.

Схема 1. Классическая.

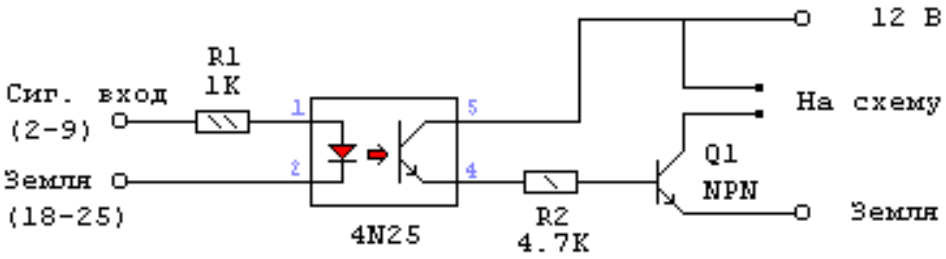


Схема предназначена для питания маломощных схем (до 12В) и их управления. Разберём подробно работу схемы:

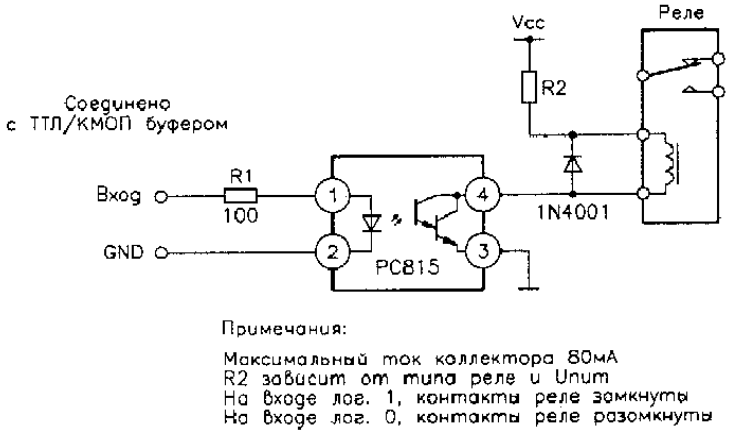
1. Основу составляет оптоизолятор 4N25 или какой-либо другой на 12В (см. Приложение). Основная функция этой детали в нашей схеме - ключ. То есть ток начинает течь в направлении 5 → 4 только тогда, когда ток течёт в направлении 1 → 2. Также оптоизолятор необходим для развязки потенциалов корпуса и нашего внешнего устройства, т.е. исключает пробой в схеме. Это делает устройства сопряжения абсолютно безопасными для порта подключения.
2. Транзистор Q1 необходим для усиления тока в цепи (т.е. является вторым каскадом усиления после фототранзистора оптоизолятора), а также в качестве ключа. Тип транзистора – любой биполярный n-p-n структуры.
3. Резистор R2 ограничивает ток базы транзистора Q1 (из опыта: для транзистора КТ683Б можно брать совсем малые сопротивления).

Итак, рассмотрим типичную работу схемы:

На сигнальном выводе, к которому прикреплена схема, возникло напряжение. Между контактами 1-2 изолятора возник ток, в результате чего открылась цепь и пошел ток между контактами 5-4. Транзистор Q1 открылся для работы в режиме усиления и нагрузка оказывается подключённой к источнику питания 12В. В роли нагрузки могут выступать различные маломощные приборы или же электромеханическое реле, которое может, в свою очередь, коммутировать большие токи и работать в цепях большого напряжения. Если прибор потребляет много меньше 12В, то следует поставить переменный резистор (между Q1 и землей, например) и настроить на оптимальный ток.

Вместо оптопары можно использовать диод и фототранзистор ИК-диапазона, не объединенные в один корпус и разнесенные на расстояние для контроля удалённых приборов. Пучок излучения должен быть узким, дабы не влиять на работу других приборов, контролируемых таким способом.
Можно также использовать лазерную указку вместо реле и фототранзистор с усилителем на выключателе у контролируемого прибора для управления на расстояниях порядка сотен метров.

Схема 2.
Эта схема аналогична предыдущей за исключением нового компонента - оптопары Дарлингтона. Эта оптопара влючает в себя транзисторную пару Дарлингтона, которая, в свою очередь, представляет собой двухкаскадный усилитель. В данной схеме использована оптопара PC815, хотя может быть использован любой другой аналог (см. Приложение).



При работе на индуктивную нагрузку (реле, например) следует применять защитные диоды, включенные параллельно нагрузке так, как показано на рисунке. Это свзано с эффектом самоидукции таких нагрузок, из-за чего после выключения внешнего питания возможен пробой соединенных с нагрузкой приборов. При работе на активную нагрузку использовать подобные диоды необязательно.

Схема 3.
Схема для коммутации токов электроосветительной сети. Схема полезна, если для нагрузки нужно напряжение более 12В. Такой нагрузкой может быть лампа накаливания, вентилятор, телевизор, магнитофон и пр.

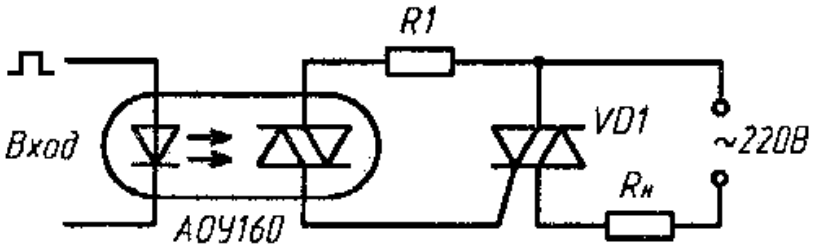


Схема состоит из оптосимистора (здесь АОУ160), симистора VD1 (т.е. симметричного тиристора, пропускающего ток в обоих направлениях), резистора R1, ограничивающего ток оптосимистора до 10мА и любого бытового прибора $R_н$.

Повторимся, что схема не работает для напряжений меньше 12В, т.к. существует так называемое остаточное напряжение на оптосимисторе, что не даёт полностью закрываться симистору в правой части схемы.

Схема 4.
Далее представлена принципиальная схема устройства, управляющего маломощной нагрузкой, основанная на оптосимисторе MOC3061 (см. Приложение). Он сотоит из светоизлучающего диода, фотосимистора и схемы контроля перехода фазы через нуль (ZC). Схема контроля перехода фазы через нуль позволяет упростить управление симистором. Дело в том, что в нормальном состоянии симистор закрыт и ток через него не течет. Для того, чтобы его открыть, нужно подать на импульс на управляющий электрод. Закрывается симистор в момент, когда ток через него становится равным нулю, то есть в момент изменения знака переменного напряжения. Чтобы мощность на нагрузке была максимальной, необходимо подавать открывающий импульс сразу после начала очередного полупериода, всего два импульса за период. Меняя задержку от перехода фазы через нуль до начала подачи открывающего импульса на управляющий электрод (т.е. меняя скважность), можно изменять мощность на нагрузке.

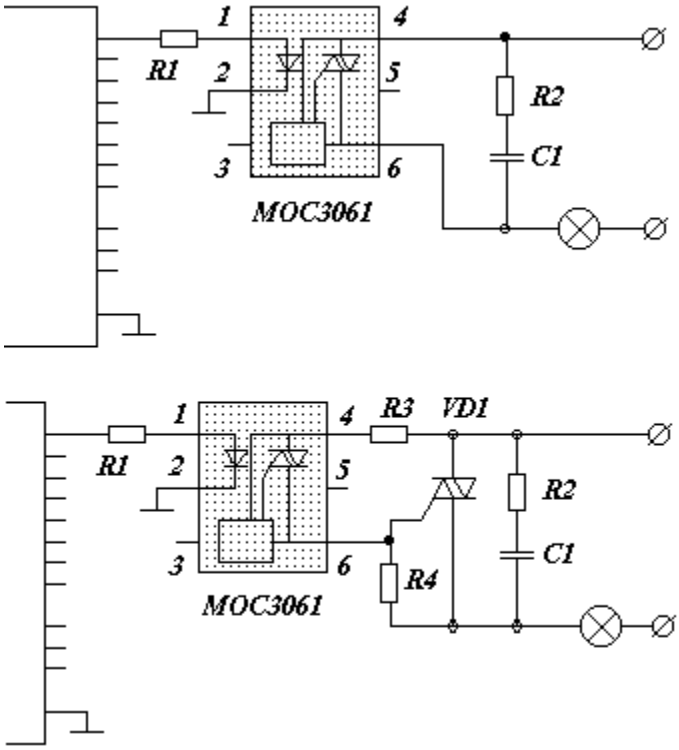
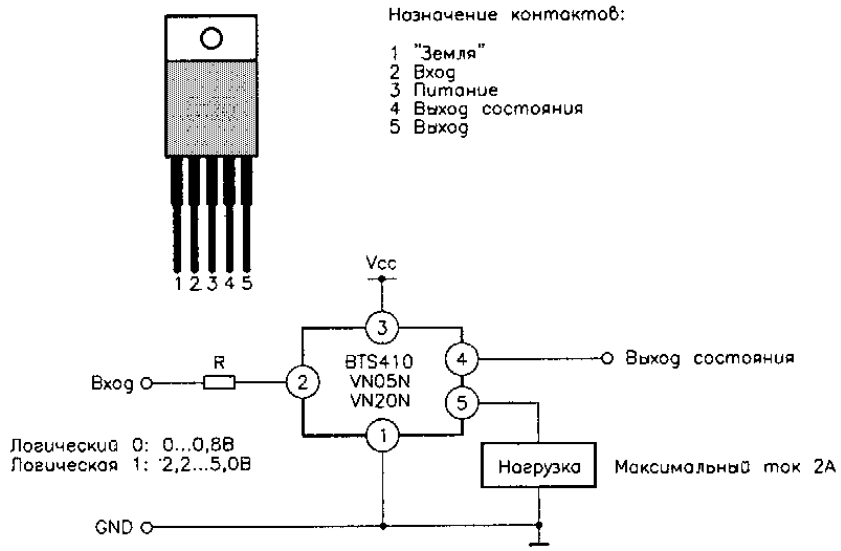


Схема 5.

Мощные устройства управления на базе МОП транзисторов с защитой, иногда называемые *твердотельными реле*, играют роль переключателей в силовых цепях цифровых схем управления. Входное управление совместимо с пятивольтовыми логическими уровнями. В этих элментах используется встроенная схема термоконтроля, которая защищает их от перегрева, короткого замыкания и больших величин тока. Она прекрывает выход при температуре 140 °C, а когда температура падает до 125 °C, термозащита выключается. такие устройства, как правило имеют информационный выход, который низким уровнем сигнализирует о срабатывании встроенных цепей защиты.



Транзистор BTS410 (см. рис) способен управлять напряжениями в диапазоне 4,9-40 В, порог срабатывания защиты от перенапряжения порядка 42-52 В. Максимальная рабочая температура равна 150 °C. В зависимости от температуры уровень превышающих значений тока колеблется в пределах 3,1-21 А. Это устройство имеет низкое сопротивление во включенном состоянии во всем диапазоне температур. Время включения и выключения составляет 60 и 50 мкс соответственно. Входное напряжение включения изменяется от 2 до 5 В, выключения - от 0 до 0,8 В. Входной ток равен 25мкА при входном напряжении 3,5 В.

Два других примера таких устройств - VN05N и VN20N (см. рис). Они рассчитаны на выходной ток 12 и 28 А, соответственно.

Ограничимся этими разработками.

Программирование LPT

Рассмотрим программирование на Паскале и Ассемблере. Тексты на C++ и Бейсика здесь также представлены, но только в виде примечаний.

Базовый пример на Паскале:

```
Uses Dos,CRT;

Var
   data:byte;          {определяем переменную DATA типа byte}
Begin
   Readln(data);       {заносим переменную}
   Port[$378]:=data;   {посылаем значение переменной в LPT-порт}
End.
```

Какое число необходимо ввести :

	Десят.	Двоичн.	Шестнадцат.
первый выход	**1**	**1**	**1**
второй выход	**2**	**10**	**2**
третий выход	**4**	**100**	**4**
четвёртый выход	**8**	**1000**	**8**
пятый выход	**16**	**10000**	**10**
шестой выход	**32**	**100000**	**20**
седьмой выход	**64**	**1000000**	**40**
восьмой выход	**128**	**10000000**	**80**

0 (0h) - "гасит" все выходы **255** (ffh) - включает все
Если нужно подать сигнал на несколько выходов, то следует сложить числа, соответствующие этим выходам.

Посылка данных идёт по определённому адресу памяти, который выделяется для каждого порта отдельно. Стандартные адреса заданы таблицей:

	LPT1	LPT2	LPT3
Вывод	378h	278h	3BCh
Ввод	379h	279h	3BDh
Статус	37Ah	27Ah	3Beh

Окончание “h” свидетельствует о шестнадцатеричной системе исчисления.

На ассемблере нужный участок кода может выглядеть так (для вывода):

```
mov dx, 378h            ;посылка байта в порт LPT1
mov al, 15h             ;значение байта 15h
out dx, al              ;собственно посылка
```

Или так (для ввода):

```
mov dx, 379h            ; адрес вывода
in  al, dx              ; прием байта в регистр al
```

Если мы не знаем, что на данный момент находится на выводах порта, но нам нужно изменить один бит, то можно воспользоваться операциями логического сложения и умножения по маске. Например:

```
mov     dx, 378h        ; адрес порта
in      al, dx          ; приём из порта
or      al, 00001000b   ; установить 4-й бит в единицу
and     al, 10111111b   ; погасить 7-й бит
xor     al, 00000001b   ; инвертировать 1-й бит
out     dx, al          ; вывод в порт
```

Т.е., если у нас было на выходе порта 11101011, то после этой операции установится 10101010.

Таблицы истинности для этих операций:

and01000101 or01001111 xor01001110

На BASIC :

```
OUT &H378, DATA         ' вывод в порт
DATA% = INP(&H379)      ' приём из порта
```

На C/C++ :

```
outp(0x378,data);       // вывод в порт
или
outportb(0x378,data);

in = inportb(0x379);    // приём из порта
или
in = inp(0x379);
```

Существуют системы, в которых адреса портов не совпадают с указанными в таблице. В этом случае следет пройтись по следующим адресам памяти:

```
0040:0008h 2 байта    Базовый I/O адрес порта LPT1, нуль, если нет
0040:000Ah 2 байта    Базовый I/O адрес порта LPT2, нуль, если нет
0040:000Ch 2 байта    Базовый I/O адрес порта LPT2, нуль, если нет
```

Например, вывод в “нестандартный” LPT1 на паскале можно сделать таким образом:

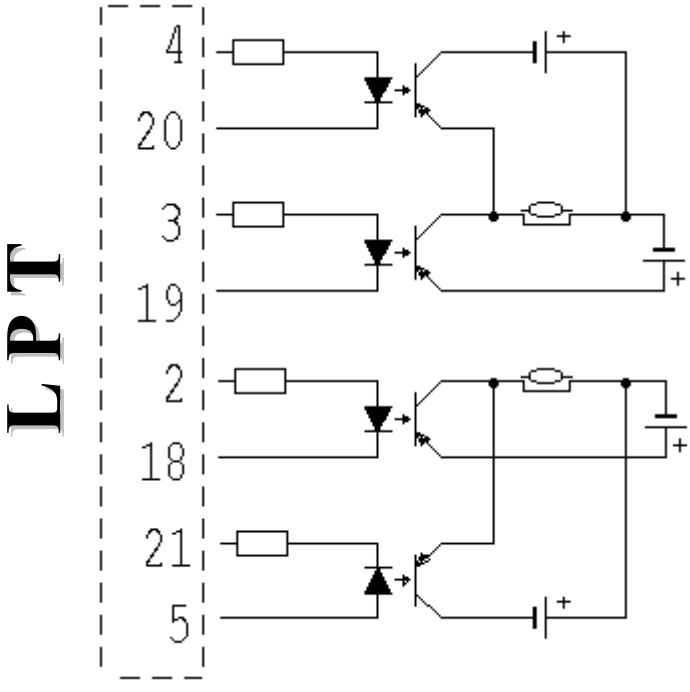
```
Port[MemW[$0040:$0008]]:=data;
```

Функция MemW подразумевает чтение слова (word) по адресу – аргументу функции.

Пример по теме:

Электронная управляемая черепашка

Следующая разработка представляет из себя некоторое подобие компьютерной мышки (устройство, а не курсор, естественно), «движениями» которой можно управлять с помощью клавиш ← ↑ ↓ → клавиатуры (скан-коды 4Bh,48h, 50h, 4Dh, соответственно).
Упрощенная схема устройства представлена на рисунке.
Здесь можно узнать 4 резистора для ограничения тока в цепях порта, 4 оптопары для развязки потенциалов порта и внешней схемы, а также 2 электродвигателя , которые являются «ногами» нашей «черепашки».
PIN N2 и 18 отвечают за движение левого колесаноги в «прямом» направлении
PIN N3 и 19 отвечают за движение правого колесаноги в «прямом» направлении
PIN N5 и 21 отвечают за движение левого колесаноги в «обратном» направлении
PIN N4 и 20 отвечают за движение правого колесаноги в «обратном» направлении
Естественно, что при движении только одного двигателя, черепашка будет разворачиваться.

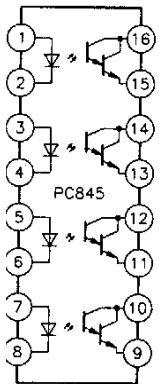


Если используются оптопары, типа 4N25, имеющие малый коэффициент передачи по току ток, то следует добавить в схему несколько транзисторов для усиления по току (см. предыдущие схемы). В случае оптопар 4N33 и им подобных следует ограничивать ток через нагрузку.

Если есть возможность, то часть схемы можно заменить микросхемой PC845 (недефицитный аналог – TLP521-4). В этом случае можно обойтись без оптопар и транзисторов второго каскада усиления, т.к. в микросхеме используются оптопары Дарлингтона.

Программная часть написана на паскале для среды Turbo Pascal:

```
program cherepashka;
Uses CRT;
var i: char;
    j: longint;
BEGIN
Port[$37A]:=32;                 {открываем для посылки данных}
Repeat                          {Повторяем...}
i:=readkey;
case i of                       {проверка буфера нажатых клавиш}
#75 : Port[$378]:=1;            {→}
#72 : Port[$378]:=3;            {↑}
#77 : Port[$378]:=2;            {←}
#80 : Port[$378]:=12;           {↓}
end;
delay(1000);                    {задержка}
port[$378]:=0;                  {гасим}
until i = #27;                  {...пока не нажата клавиша ESC}
END.
```



Поющий принтер

Идея разработки состоит в том, что можно посылать байты FFh и нуль-байт в порт подряд с некоторой задержкой, тем самым добиваясь смены напряжений на выходе порта с некоторой частотой (импульсный генератор). Оказалось, что эта частота может совпадать со звуковыми частотами и даже превосходить их по величине.
Для проверки можно подключить любой динамик к выходам LPT-порта следующим образом:
- первый контакт посадить на любой контакт из диапазона [2-9] (выход данных)
- второй контакт посадить на корпус или "земляной" контакт порта из диапазона [18-до конца].
Потом следует запустить программу, которая будет отправлять данные в порт.
Ниже - подобная программа, в которой, ради наглядности, можно менять частоту, двигая мышью вправо или влево. В результате можно услышать довольно громкие и отчётливые звуки (U=3.5V).

```
uses crt, graph;
var i, driver, mode: Integer;
begin
i:=0;
driver:=detect;                 {переходим в графический режим...}
initgraph(driver, mode,'');
Repeat
asm
mov dx, 378h                    {посылаем байт 11111111b в LPT-порт}
mov al, 0FFh
out dx, al
end;
delay(i);                       {задержка...}
asm
mov dx, 378h                    {посылаем байт 00000000b в LPT-порт}
mov al, 0h
out dx, al
end;
delay(i);
asm
mov ax, 03h                     {считываем положение мыши на оси X}
int 33h                         {и ставим задержку}
mov i, cx
mov ax, 01h
int 33h                         {показываем курсор мыши}
end;
```

```
until keypressed;                    {нажмите любую клавишу для выхода}
closegraph;                          {переход в текстовый режим}
end.
```

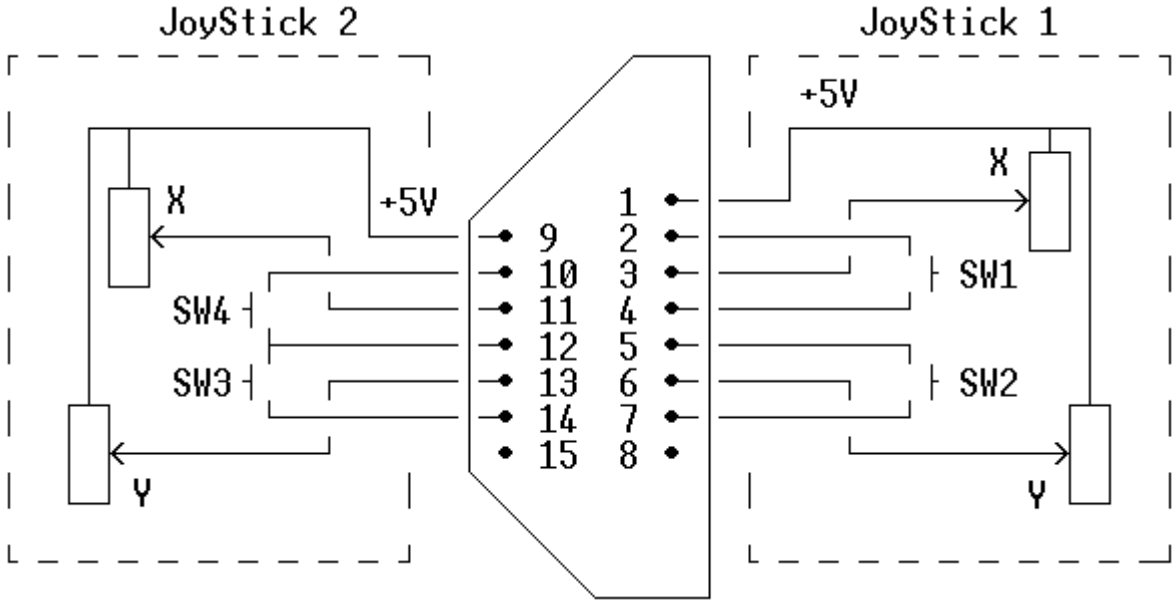
GAME-порт

Теперь от программирования LPT-порта перейдём к программированию игрового порта.
Этот порт чаще всего используется для подключения джойстиков, рулей и прочих игровых приспособлений. Однако с его помощью можно решать широкий круг задач таких, как, например, измерение температуры, сопротивления, уровни освещенности и пр.

Для начала рассмотрим распайку порта. Выглядит она следующим образом:

N	Направление	Сигнал
1	*Выход*	*+5V*
2	*Вход*	*Кнопка 1 джойстика A*
3	*Вход*	*Значение X джойстика A*
4		*Ground*
5		*Ground*
6	*Вход*	*Значение Y джойстика A*
7	*Вход*	*Кнопка 2 джойстика A*
8	*Выход*	*+5V*
9	*Выход*	*+5V*
10	*Вход*	*Кнопка 1 джойстика B*
11	*Вход*	*Значение X джойстика B*
12		*Ground*
13	*Вход*	*Значение Y джойстика B*
14	*Вход*	*Кнопка 2 джойстика B*
15	*Выход*	*+5V*

Для наглядности подключения рассмотрим следующую картинку:



Как видно, к одному порту можно подключить 2 джойстика. Каждый джойстик состоит из 2 переменных резисторов (0-250КОм) и 2 переключателей\кнопок. При изменении угла поворота ручки резистора меняется его сопротивление. Компьютер воспринимает и оцифровывает значение этого сопротивления, и далее происходит обработка некоторого целого значения, соответствующего сопротивлению. Это значит, что, используя GAME-порт, возможно построить цифровой омметр. Рассмотрим на конкретном примере:

Цифровой омметр

Перед запуском программы следует подключить щупы будущего «омметра» к выходам 1 и 6 GAME-порта (см. схему).

```
uses crt;
const k=1.36;      {коэфициент для перевода условных единиц в КОмы (можно менять)}
var y,y1: word;
    a: char;
begin
clrscr;
repeat
asm
mov ah, 84h        {вызов прерывания 15h ф-ция 84h подфункция 1h}
mov dx, 1h
int 15h
mov y,bx           {смотрим состояние джойстика A по оси ординат}
end;
Writeln('Нажмите ESC для выхода или любую другую клавишу для замера');
if y<>0 then writeln('Сопротивление ', y*k:1:1, ' КОм')
else writeln('Резистор не подключён !');
writeln;
a:=readkey;        {выводим измерения на экран}
Delay(3000);       {задержка}
until a=#27;       {выход, если нажали ESC}
end.
```

Достоинства такого виртуального "омметра" перед реальным следующие:

- Простота сборки и использования для случая, когда нет под рукой тестера, но есть компьютер.
- Легкость организации баз данных по резисторам для возможности дальнейшего поиска или сортировки.
- Возможность модернизации программы для решения различных задач (к примеру: поиск во множестве резисторов определённого по номиналу и задание программе воспроизведение сигнала, если номинал резистора близок к искомому)
- Быстрая скорость измерения (если убрать задержки, то можно пропускать до нескольких десятков резисторов в секунду)
- При подключении вместо резистора терморезистора или опторезистора и правильном подборе коэфициента k можно измерять температуру в доме или на улице, затем эти данные обрабатывать и выводить, к примеру, на рабочий стол. Также можно следить за температурой (одновременно) процессора, HDD, микросхем видеокарты и т.п., и задавать программе реакцию при перегреве...
- 4 входа для измерений. Можно одновременно проверять до 4-х резисторов одновременно и программно сравнивать эти значения. (например сравнивать разность температур внутри помещения и снаружи и, в зависимости от результата, включать программно нужные приборы)

Недостатки :

- Диапазон измеряемых значений - от 3 КОм до 1 МОма, то есть нельзя измерять сопротивления порядка нескольких сотен\десятков\единиц Ом
- Погрешность около 1-2 Ком. Видимо, коэфициент k следует изменять от порядка измеряемой величины
- Может не работать под системами WinXP\2000\NT. Для этих систем следует писать отдельный драйвер.

Теперь подробно рассмотрим работу нашей программы. Ядро, т.е. смысловая часть, выделена красным цветом. Здесь вызывается прерывание 15h (функция 84h) для опроса состояний кнопок джойстика и угла поворота составляющих его потенциометров. Результаты мы получаем в виде целых чисел, которые потом нужно перевести в Омы. С высокой точностью можно утверждать, что выдаваемые значения прямо пропорциональны замеряемому сопротивлению. Тогда мы можем ввести коэфициент пропорциональности k=1.36, например.

Подробнее о функции 84 прерывания INT15h (переведено из списка прерываний Ральфа Брауна):
Входные параметры:
AH = **84h**
DX = подфункция:
0000h считать положение кнопок джойстика
На выходе: в регистре **AL** биты 7-4 - это состояния соответствующих кнопок.
0001h считать текущее положение «курсора» (т.е. замерить сопротивление)
На выходе:**AX** = координата X джойстика A
BX = координата Y джойстика A
CX = координата X джойстика B
DX = координата Y джойстика B

На выходе:
Если **CF** ≠ 0, то **AH** = статус:
80h ошибочная команда (PC, PCjr)
86h функция не поддерживается (XT)
Если **CF** = 0, то все норамально
Примечание1: Если нет установленного игрового порта , подфункция 0000h возвращает AL=00h (все кнопки отжаты) и подфункция 0001h возвращает AX=BX=CX=DX=0000h
Примечание2: Джойстики на 250Ком возвращают значения в диапазоне [0000h <-> 01A0h], т.е. от 0 до 416

Цифровой омметр термометр для среды Windows 98

При желании сделать программу с Windows-интерфейсом возникает проблема: под Windows, как и для любых других систем, работающих в защищенном режиме, нельзя пользоваться прерыванием int15h, как и другими прерываниями реального режима, а значит нужно искать другой способ работы с GAME-портом.
Такой способ существует. Статус порта можно получить, приняв байт из порта 201h (513d).
Формат принимаемого байта (ячейка соответствует одному биту):

Кнопка 2 джойстика B (1=отжато, 0=нажато)	**Кнопка 1** джойстика B (1=отжато, 0=нажато)	**Кнопка 2** джойстика A (1=отжато, 0=нажато)	**Кнопка 1** джойстика A (1=отжато, 0=нажато)	**Ось Y** джойстика B (1=ожидание, 0=конец)	**Ось X** джойстика B (1=ожидание, 0=конец)	**Ось Y** джойстика A (1=ожидание, 0=конец)	**Ось X** джойстика A (1=ожидание, 0=конец)
Бит 7	**Бит 6**	**Бит 5**	**Бит 4**	**Бит 3**	**Бит 2**	**Бит 1**	**Бит 0**

С кнопками всё понятно, но как забирать сопротивления потенциометров? Оказывается, сопротивление пропорционально времени, в течение которого установлена единица в битах 0-3, т. е. программа должна подсчитывать количесто циклов в ожидании, пока установится нужный бит.

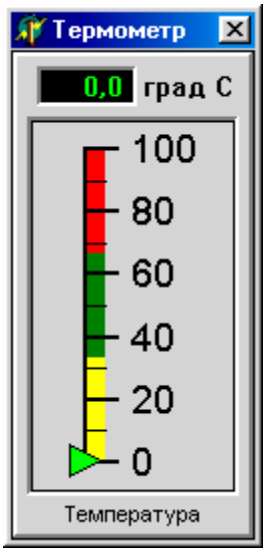
Разберём на примере программы на Си:

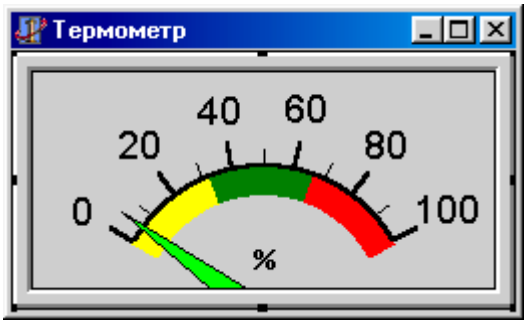
```
#include <conio.h>                    /* Библиотека ввода\вывода */

int main(void)    {
int i;
unsigned int joyState;                /* Байт из порта 201h */
unsigned short joyPotVal[4], limit;/*Счётчик цикла и максимальное значение циклов
*/

clrscr();                             /* Очистка экрана */
gotoxy(10, 9);                        /* Установить позицию курсора */
cputs("Joystick potentiometer values:");

/* Задание начальных параметров */
joyPotVal[0] = joyPotVal[1] = joyPotVal[2] = joyPotVal[3] = limit = 0;

outp(0x201, 0);                       /* Обнуляем статус */
do {
            joyState = inp(0x201);        /* Опрос порта */
                                          /* Увеличиваем значения для каждого
счётчика */
            joyPotVal[0] += ((joyState & 0x01) != 0);
            joyPotVal[1] += ((joyState & 0x02) != 0);
            joyPotVal[2] += ((joyState & 0x04) != 0);
            joyPotVal[3] += ((joyState & 0x08) != 0);
            limit++;                                  /* Увеличиваем максимальное значение
*/
} while ((joyState & 0x0F) && limit);     /* Продолжаем пока биты 0-3 не будут
                                                 обнулены или не достигнем
максимального                                                    значения */
/* Вывод статуса */
gotoxy(15, 11);
cprintf("Joystick A's X-axis = %u", joyPotVal[0]);
gotoxy(15, 12);
cprintf("Joystick A's Y-axis = %u", joyPotVal[1]);
gotoxy(15, 13);
cprintf("Joystick B's X-axis = %u", joyPotVal[2]);
```

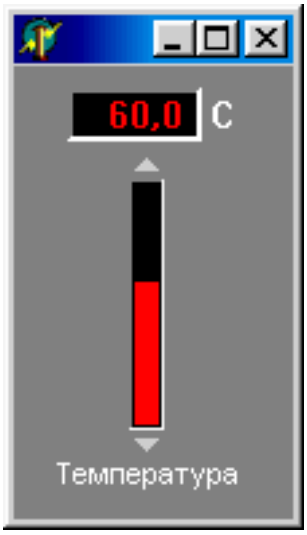
```
gotoxy(15, 14);
cprintf("Joystick B's Y-axis = %u", joyPotVal[3]);
return 0; }
```

Большой недостаток такой программы – зависимость результата от производительности компьютера. Чтобы получить аппаратно-независимый вариант программы, следует воспользоваться возможностями ассемблера (см. FAQ по ассемблеру на http://kalaidjian.narod.ru/rusfaq.rar).

Разработки под Windows напишем на Delphi. Нам нужна будет форма, таймер(вызывается 100 раз в секунду) и компонет из библиотеки Abakus (можно обойтись и текстовым полем, но так будет нагляднее).







Опишем саму процедеру таймера: считаем положение оси ординат первого джойстика и передадим Abakus-компоненту в виде свойства объекта:

```
procedure TForm1.Timer1Timer(Sender: TObject);
const k=0.27675276; b=114.6051660;
var count: word;                // счётчик времени
label nac, konec, en;           // метки
begin
count:=0;                       // обнуляем счётчик

asm
mov  dx, 201h                   // адрес порта
mov  al, 0h                     // обнуляем статус
out  dx, al

nac:                            // начало цикла
in   al, dx                     // опрос порта 201h
test al, $02                    // если первый бит=0, то выходим из цикла
jz   konec

inc  count                      // увеличиваем значение счётчика
cmp  count, $FFF                // сравниваем с максимальным значением задержки
jne  en                         // если не равно, то продолжаем раюоту. Если равно, то
                                      // обнуляем счётчик и выходим из цикла
mov  count, 0
jmp  konec

en:
jmp  nac                        // переходим к следующей итерации

konec:
end;
if Abs(count-last) > 3 then                        // пренебрегаем малыми изменениями
Abvmeter1.value:=k*count+b;    // устанавливаем атрибут Abakus-//компонента
last:=count;
```

```
end;
```

Значение count часто бывает полезно обрабатывать, для того, чтобы вписать его в нужный промежуток(скажем, 1..100), а также для того, чтобы пренебречь дребезгом контактов и различными шумами (без этого замеряемое значение будет быстро меняться в малых пределах и отображаемое на экране значение будет трудно воспринимать). В нашем случае используется термистор с сопротивлением порядка нескольких МОм при комнатной температуре.

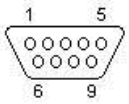
Если термистор имеет линейный характер зависимости сопротивления от температуры, то следует ввести линейную функцию зависимости температуры (T) от значения, выдаваемого АЦП GAME-порт a(count) **T = k * count + b** , где k и b определяются экспериментально. Исходя из значений этой функции будет определяться результат работы нашей программы.

Дистанционное управление

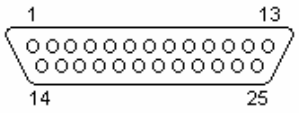
Приемная часть системы дистанционного управления выполняет три основные функции: прием сигнала, распознавание, формирование управляющей команды. Таким образом, задача разбивается на две части: Программная и аппаратная. Логично две последние полностью поручить компьютеру, хотя некоторые производители приемников ИК-сигнала считают иначе, оставляя функции распознавания внешнему устройству. Это значительно упрощает программную часть, но приводит к удорожанию самого устройства, примеры таких устройств можно встретить в любом серьезном компьютерном салоне и здесь они рассматриваться не будут. Использование ИК-приемника, например встроенного TV-тюнера, тоже достаточно удобный вариант, правда требует наличие самого тюнера, что не всегда приемлемо.
Наибольший интерес представляют простые, но тем не менее достаточно надежные методы решения этой задачи.

Подключение к Com-порту

Для начала рассмотрим распиновку COM порта (9 PIN и 25PIN):

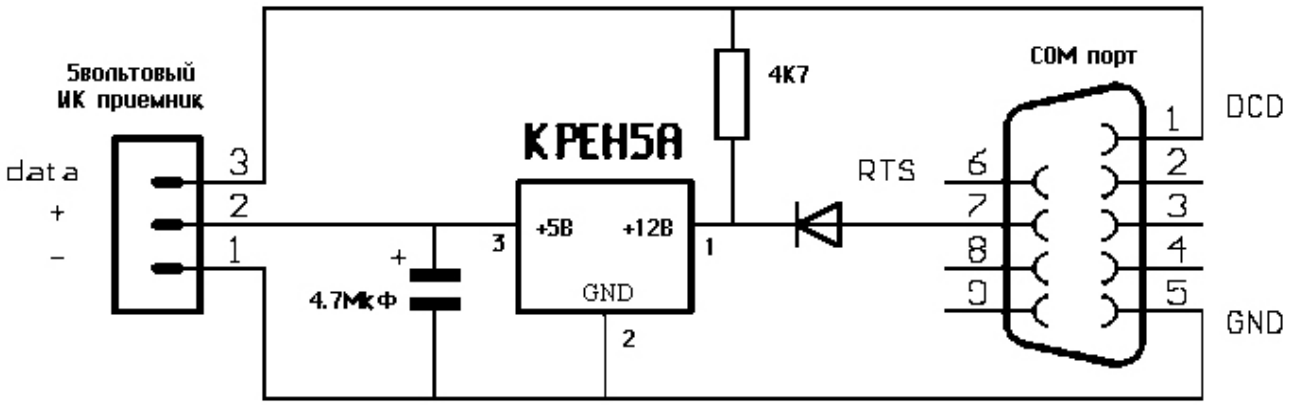


Pin	Name	Dir	Description
1	DCD	←	Carrier Detect
2	RXD	←	Receive Data
3	TXD	→	Transmit Data
4	DTR	→	Data Terminal Ready
5	GND	—	System Ground
6	DSR	←	Data Set Ready
7	RTS	→	Request to Send
8	CTS	←	Clear to Send
9	RI	←	Ring Indicator



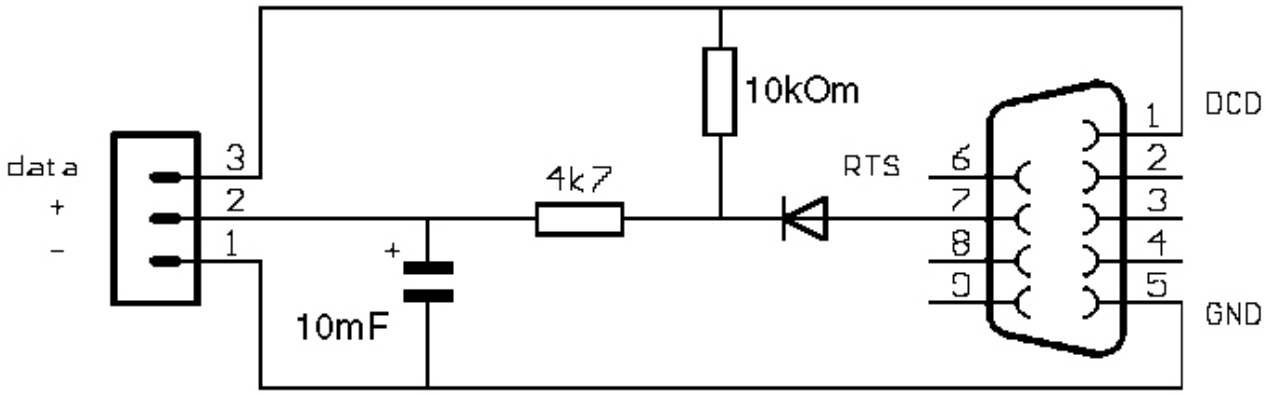
Pin	Name	Dir	Description
1	SHIELD	-	Shield Ground
2	TXD	→	Transmit Data
3	RXD	←	Receive Data
4	RTS	→	Request to Send
5	CTS	←	Clear to Send
6	DSR	←	Data Set Ready
7	GND	-	System Ground
8	DCD	←	Carrier Detect
9	n/c	-	
10	n/c	-	
11	n/c	-	
12	n/c	-	
13	n/c	-	
14	n/c	-	
15	n/c	-	
16	n/c	-	
17	n/c	-	
18	n/c	-	
19	n/c	-	
20	DTR	→	Data Terminal Ready
21	n/c	-	
22	RI	←	Ring Indicator
23	n/c	-	
24	n/c	-	
25	n/c	-	

Схема ИК-приемника, используемая для дистанционного управления:

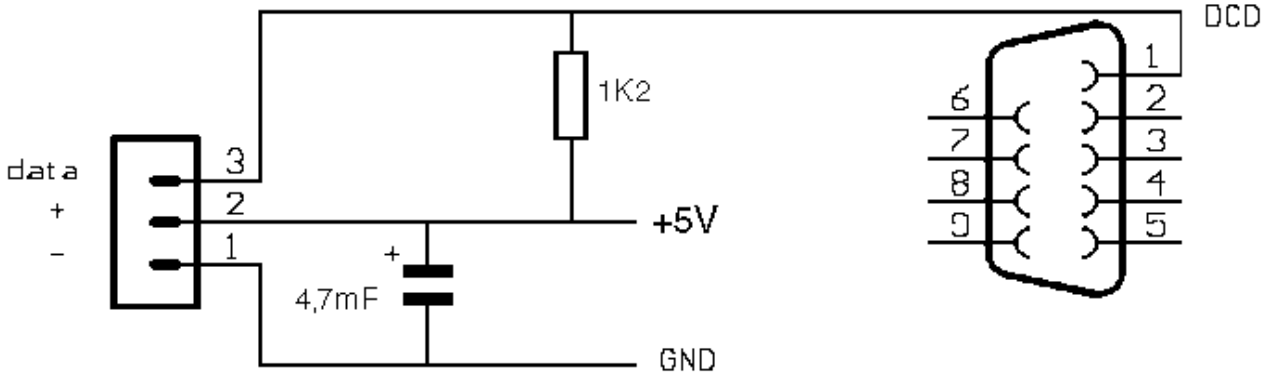


Для ИК-приемника подойдет любой 5 вольтовый инфракрасный приемник, используемый в бытовой аппаратуре, например, фотомодуль TSOP1836. Аналоги: IS1U60L, GP1U52X, SFH506-36 или отечественный ТК1833. Микросхема - стабилизатор напряжения КРЕН5А(или популярный аналог 78L05), которая необходима для питания ИК приемника 5 вольтовым напряжением (с 7-го контакта COM-порта поступает напряжение 12-15 вольт). Резистор можно выбрать из диапазона 3-5 кОм, конденсатор 4.7-10 мкФ. Диод - любой маломощный.
В приведенной схеме выходной сигнал подается на 1 контакт COM порта (DCD) этот контакт не используется стандартной мышью для COM порта, поэтому устройство можно использовать параллельно с мышью. (но не с модемом!) Выходной сигнал можно подавать не только на DCD но и на другие контакты(CTS, DSR). Все эти параметры можно выставить в программе, которая работает в ИК приемником.

Предыдущую схему можно упростить, убрав стабилизатор напряжения:



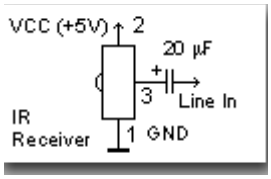
В случае питания от внешнего источника питания 5 вольт схему можно упростить еще:



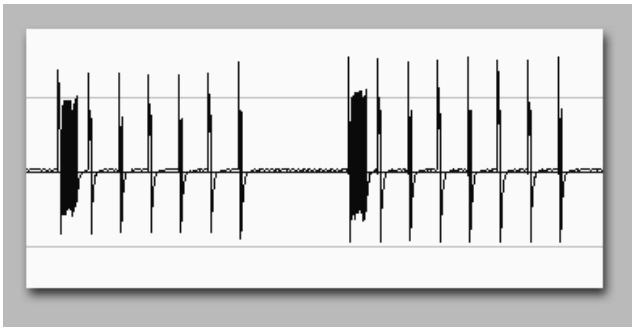
В этом случае важно сравнять потенциалы корпусов источника питания и компьютера, дабы предотвратить ток утечки и как следствие ложные показания замеряющей программы. Замеряющая программа может быть как самодельной (для нестандартных пультов), так и готовой WinLIRC, предназначенной для обработки различных сигналов в общем случае и сигналов от пультов, в частности. Написание собственной программы для обработки является темой отдельного проекта.

Подключение к звуковой карте

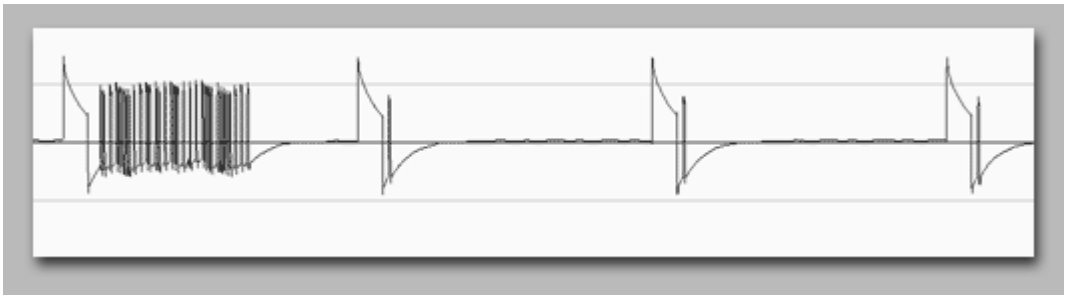
Подключим IR-приёмник к звуковой карте, как показано на рисунках:



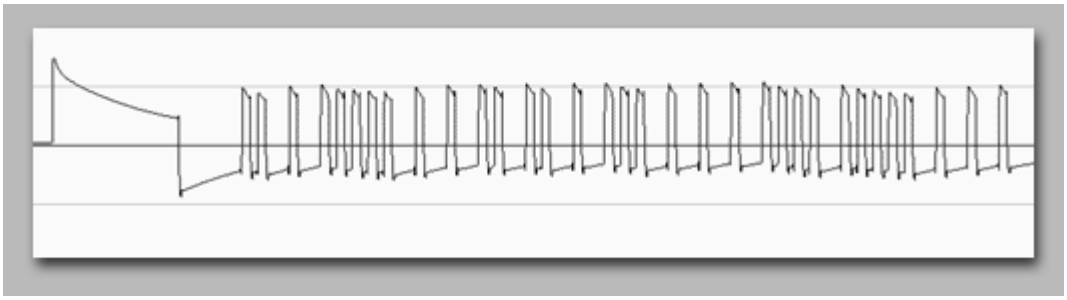
Проанализируем сигнал, полученный при нажатии кнопки ДУ-пульта. Сперва идет заголовок, содержащий информацию о нажатой кнопке, а потом серия повторяющихся импульсов, говорящих о том, что она еще нажата.



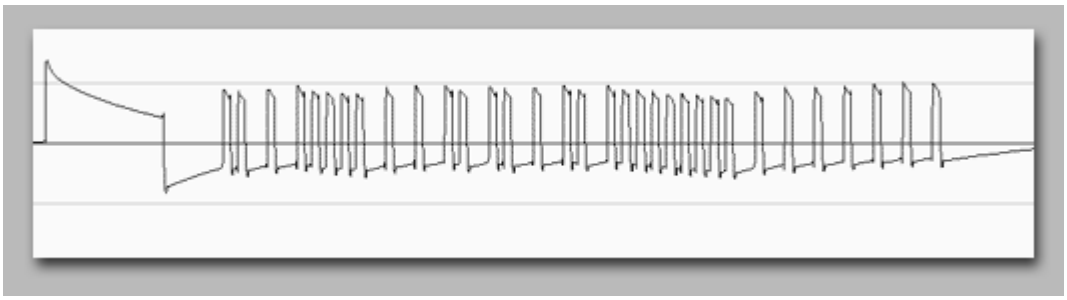
Общий вид сигнала от нажатия и непродолжительно удержания двух кнопок.



Увеличенный вид сигнала от нажатия кнопки «power».



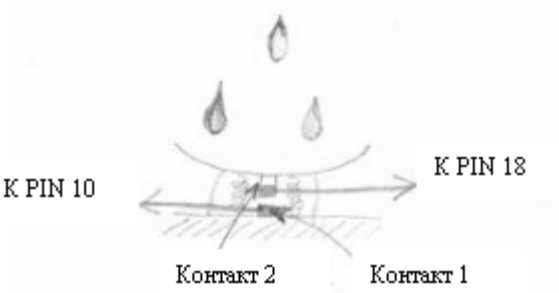
Увеличенный заголовок сигнала «power».

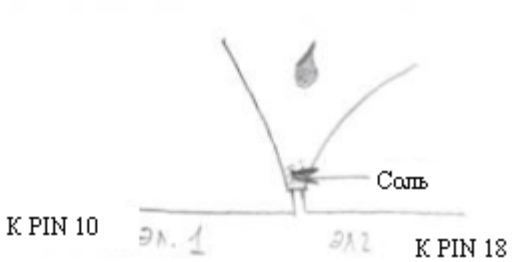


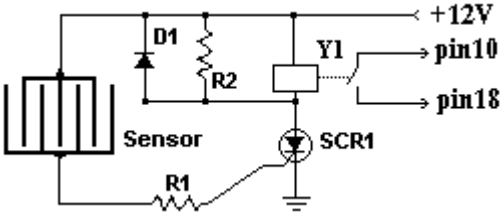
Увеличенный заголовок сигнала «one».

После просмотра этих осциллограмм может возникнуть идея написать программу, которая будет сканировать сигнал со входа микрофона и преобразовывать его в команды управления компьютером.

В процессе написания проекта обнаружилось, что такая программа уже существует. Называется она Sly Control. Предназначена для различных видов пультов ДУ, в том числе способна обрабатывать импульсы поступаемые с фотодиода подключенного к микрофонному входу звуковой карты.





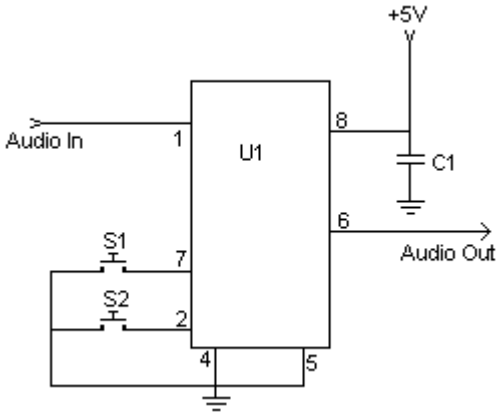


R1 - 1k/0.25Вт, R2 - 680/0.25Вт
D1 - 1N4001, SCR1 - C106B1
Y1 – реле на 12В, замыкающее pin10 LPT на землю.
Sensor – контактная площадка от клавиатуры.
Все, кроме сенсора следует убрать в корпус.

Общие идеи

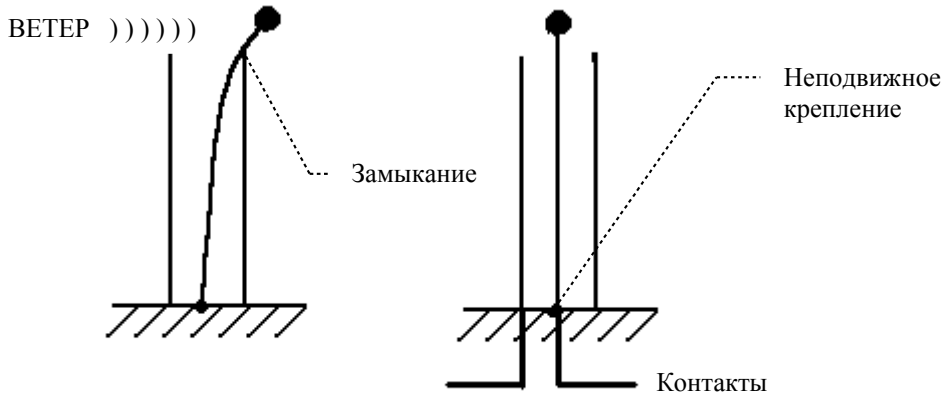
Далее следует список идей, которые автор не раскрыл в этом проекте, не привёл в жизнь или которые находятся на стадии разработки.

1. «Электронный чайник» (термистор погружён в воду. Его сопротивление контролируется «цифровым
2. омметром». При переходе через некоторое пороговое значение, соответствующее сопротивлению при 100°C, происходит выключение чайника из сети и голосовое оповещение). Важно, чтобы ножки термистора были изолированы от воды, иначе показания будут соответствовать сопротивлению воды, а не сопротивлению термистора
3. «Индикатор солнца» (фоторезистор, подключенный к GAME-порту).
4. Регулировка тока в цепи для достижения определённой температуры(утюга, например) или освещённости:
5. с помошью цифровых потенциометров. При посылке импульса по определённому каналу происходит увеличение\уменьшение общего сопротивления на некоторую величину. При этом достаточно всего 2 каналов данных. Например, можно регулировать громкость звука:
6. с применением каскада транзисторов (или опто -тиристоров\-симисторов) с резисторами; при открытии транзистора активизируется часть цепи с дополнительным сопротивлением, что уменьшает ток в цепи. Регулировка может производиться с помощью параллельного порта (15 каналов, т.е. 15 режимов)
7. С применением симисторов-триаков (схема с *http://www.aaroncake.net/*):



R1- 50k (или цифровой потенциометр, управляемый с ПК), R2 - 15k(0.5Вт)
C1, C2 – 0.068\250В
L1 - контролируемая лампа (до 350В)
L2 – неоновая лампа
TR1 - 40502 TRIAC. Питание 117VAC (не 220 !)

8. Ускоренный набор номера телефона в пульсовом режиме. Набор в режиме пульса может происходить быстрее, чем это делается модемом или телефоном (испробовано на практике). Можно замыкая\размыкая линию нужным образом, набирать номер, причём в быстром темпе. Для этого нужен всего один канал порта, по которому можно посылать импульсы в определённой очерёдности, которые в дальнейшем открывают\закрывают транзисторные цепи (см. схемы в начале доклада). Приспособление полезно для автоматического прозвона на часто занятые номера.
9. Детектор присутствия – лазерная указка, направленная на фотодиод. Если диод закрывается, то это значит, что на пути луча возникло препятствие. Необходима 1 линия LPT-, COM- или GAME-порта.
10. Отпугивание комаров и грызунов с помощью разработки «поющий принтер» при переходе в ультразвуковой диапазон.
11. Индикаторы дождя. Верхний левый и нижний вариант можно подключать к LPT, верхний правый используется как с LPT (с солевыми добавками для понижения сопротивления), так и с GAME-портом (без добавок).
12. Индикатор ветра. Индикатор состоит из полой проводящей трубки и закреплённого гибкого стержня или пружины. При ветре стержень касается поверхности трубки и замыкает цепь с портом ввода вывода. Эти данные могут последовательно анализироватся программой.



13. Управление импульсными приборами (с регулировкой скважности), создание таймеров и часов. Контроль любых приборов, содержащих светодиоды. В этом случае сместо светодиода можно использовать один из входов LPT-порта и контролировать зажигание такого «виртуального» светодиода, а значит и предпринимать какие-то запрограммированные действия в зависимости от состояния светодиода.
14. Электронная система «РОБО7» для интеллектуального управления домом. Подразумевает под собой голосовое управление домом и отдельными приборами. Основывается на 3 идеях:
    1. распознование голоса человека.
    2. управление бытовыми приборами последством портов ввода-вывода.
    3. воспроизведение текста, генерируемого программой в голосовые сообщения.

Такая система осуществляет:

- Контроль температуры и освещенности помещения.
- Замер температуры в помещении, на улице, определение освещенности и выдача данных при запросе пользователя. В зависимости от освещенности можно контролировать яркость света в помещении.
- Автополив растений и автоподкормка домашних животных (особенно полезно для ухода за аквариумными рыбками).
- Голосовой контроль электрических и СВЧ-печек (установка температуры и интервала времени)
- Контроль чайника, кофеварки (автоподогрев)
- Голосовое управление телевизором, радио и магнитофоном. Регулировка частоты радиоприёма происходит с помощью цифрового потенциометра и варикапа с помощью замены обычного резистора, прикрепляемого к ручке настройки, на цифровой потенциометр и управление им с помощью любого порта ввода/вывода. Также для управления можно использовать шаговые моторы. Для регулировки громкости телевизора или номера канала можно подключить провода к соотвествующим кнопкам регулировки громкости или переключения каналов на панели телевизора или пульта ДУ и замыкать эти выводы с помощью открывания отоизоляторов, которые в свою очередь соединяются с I/O портом.
- Набор номера телефона с помощью шагового мотора и дискового аппарата.
- Включение ультразвукового отпугивателя насекомых в определённое время суток (см. «поющий принтер»)
- Включение определённых приборов (лампа, вентилятор, обогрев и пр.) при нахождении человека в помещении (здесь организуется реле присутствия с помощью пары диод-фототранзистор инфракрасного диапазона и параллельного порта).

И многое другое – это зависит от нужд пользователя. Управление происходит методами, описанными в начале статьи. Важная деталь – управление происходит голосом, т.е. следует поставить распознаватель речи, который будет записывать (может быть даже с ошибками) интерпретируемый им текст в файл, который затем будет обрабатываться головной программой. Если пользователь хочет ознакомиться с какой-либо информацией (температура, например), то головной программой отсылается текстовый файл программе – преобразователю текстовой информации в голос. Чтобы не смешивать обычную речь с командами, можно включить реакцию программы только после произнесения какой-либо ключевой фразы или слова (например «имени» системы – «Робо7»). Например вкл\выкл света можно производить командой: «РОБО7 свет». Программа в ответ инвертирует один из битов выхода LPT-порта, в результате чего открывается\закрывается цепочка соединнения источника питания и лампы.

Возможен удаленный контроль такой системы, для чего следует включить в основную программу блок передачи данных по сетям.

Заключение

В процессе создания проекта были изучены методы программирования портов ввода\вывода и разработаны собственные схемы управления внешними приборами и схемы управления ПК с помощью пультов дистанционного управления. Также были описаны некоторые известные ранее методы сопряжения, отсутствующие в русскоязычной литературе.

В результате был разработан класс программных и аппаратных схем, предназначенных для управления внешними приборами с помощью персонального компьютера (ПК), совершения измерений различных физических величин с помощью ПК, управления персональным компьютером пультами дистанционного управления.

Приложение

Далее представлены характеристики оптопар, упоминавшихся в данном докладе или им аналогичных.

ОПТОПАРЫ ТРАНЗИСТОРНЫЕ

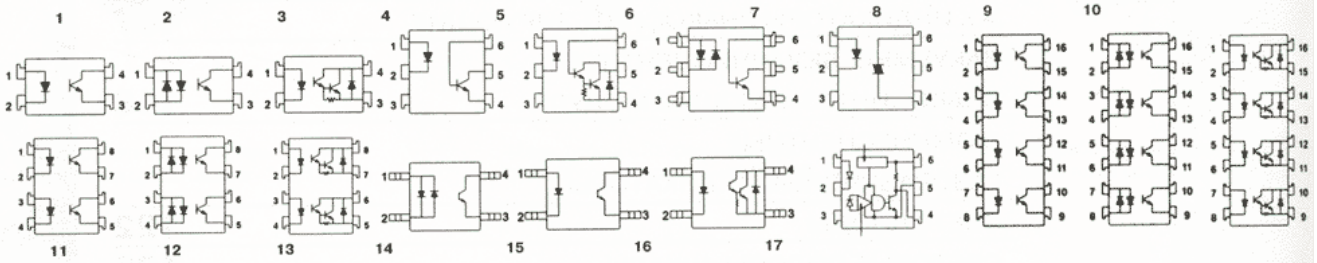
Наим-е	Максимальное выходное напряжение коллектор-эмиттер	Максимальная выходная рассеиваемая мощность	Напряжение изоляции	Максимальное прямое напряжение	Максимальный выходной темновой ток коллектора	Коэффициент передачи тока	Сопротивление изоляции	Время нарастания	Время спада	Функциональная схема (номер рисунка)	Тип корпуса
	Uceo, В	Pd max., мВт	Uiso, В, ср.кв.	Uf max., В	Iк, А	CTR, %	Riso, МОм	Trise, мкс	Tfall, мкс		
KP1010	60	150	5000	1,4	10^{-7}	60-600	5×10^{10}	20	20	1	PDIP-4
KP1020	60	150	5000	1,4	10^{-7}	60-600	5×10^{10}	20	20	11	PDIP-8
KP1040	60	150	5000	1,4	10^{-7}	60-600	5×10^{10}	20	20	8	PDIP-16
KP2010	60	150	5000	1,4	10^{-7}	60-600	5×10^{10}	20	20	4	PDIP-6
KP3010	60	150	5000	1,4	10^{-7}	60-600	5×10^{10}	20	20	2	PDIP-4
KP3020	60	150	5000	1,4	10^{-7}	60-600	5×10^{10}	20	20	12	PDIP-8
KP3040	60	150	5000	1,4	10^{-7}	60-600	5×10^{10}	20	20	9	PDIP-16
KP4010	300	200	5000	1,4	10^{-6}	600-9000	5×10^{10}	300	250	3	PDIP-4
KP4020	300	200	5000	1,4	10^{-6}	600-9000	5×10^{10}	300	250	13	PDIP-8
KP4040	300	200	5000	1,4	10^{-6}	600-9000	5×10^{10}	300	250	10	PDIP-16
KP5010	300	200	5000	1,4	10^{-6}	600-9000	5×10^{10}	300	250	5	PDIP-6
KP6010	60	150	5000	1,4	10^{-7}	60-600	5×10^{10}	20	20	6	PDIP-6
KP7010	4,5 - 17 [1]	150	5000	1,4	1мА [2]	[3]	5×10^{10}	15	9	17	PDIP-6
KP7110	4,5 - 17 [1]	150	5000	1,4	1мА [2]	[4]	5×10^{10}	9	15	17	PDIP-6
KPC354NT	60	150	3750	1,4	10^{-7}	20-400	5×10^{10}	18	18	14	Micro-6
KPC357NT	60	150	3750	1,4	10^{-7}	50-600	5×10^{10}	20	20	15	Micro-6
KPC452	300	150	3750	1,4	10^{-7}	1000	5×10^{10}	300	100	16	Micro-6

ОПТОПАРЫ ТИРИСТОРНЫЕ

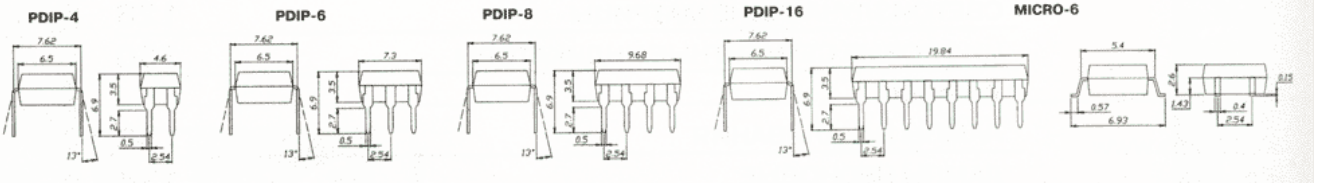
Наим-е	Выходное напряжение отсечки	Максимальная выходная рассеиваемая мощность	Напряжение изоляции	Максимальное прямое напряжение	Максимальный входной обратный ток утечки	Напряжение включения	Минимальный ток переключения	Скорость нарастания напряжения отсечки	Макс. время выключения	Функциональная схема (номер рисунка)	Тип корпуса
	Udrm, В	Pd max., мВт	Uiso, В, ср.кв.	Uf max., В	Ir, мкА	Uon, В	Ift, мА	dv/dt, В/мкс	Toff, мкс		
KMOC3022	400	300	5000	1,5	10	3	10	10	100	7	PDIP-6

1) Напряжение питания
2) Ток включения
3) Нормально замкнутые контакты
4) Нормально разомкнутые контакты

ФУНКЦИОНАЛЬНЫЕ СХЕМЫ



ТИПЫ КОРПУСОВ



Краткие характеристики

TEMIC

QUALITY TECHNOLOGIES CORPORATION

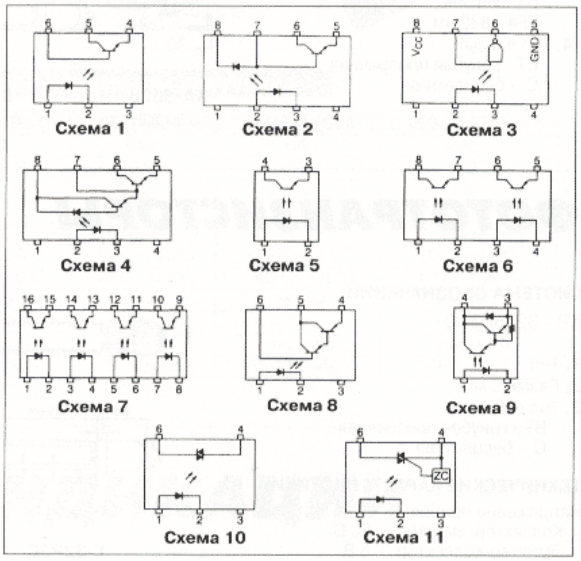
SIEMENS MOTOROLA

TOSHIBA

ТРАНЗИСТОРНЫЕ ОПТОПАРЫ						
Наименование	СХЕМА №	Коэфф.К передачи по току (%), при I пр.=10мА	U кэ, В	I k,вых., макс., В	U из. кВ	Тип корпуса
4N25	1	20	30	150	2.5	PDIP 6
4N26	1	20	30	150	2.5	PDIP 6
4N27	1	20	30	150	2.5	PDIP 6
4N28	1	20	30	150	2.5	PDIP 6
4N29	8	100	30	150	2.5	PDIP 6
4N32	8	500	30	150	2.5	PDIP 6
4N33	8	500	30	150	2.5	PDIP 6
4N35	1	100	30	150	2.5	PDIP 6
4N37	1	100	30	150	2.5	PDIP 6
4N38	1	10	80	100	2.5	PDIP 6
4N38A	1	10	80	100	2.5	PDIP 6
6N136	2	16			2.5	PDIP 8
6N137	3	700			2.5	PDIP 8
6N139	4	400			2.5	PDIP 8
CNY17-2	1	125	70	100	2.5	PDIP 6
CNY17-4	1	320	70	100	2.5	PDIP 6
TLP521-1	5	600	55		2.5	PDIP 4
TLP521-2	6	600	55		2.5	PDIP 8
TLP521-4	7	600	55		2.5	PDIP 16
TLP621	5	600	55		2.5	PDIP 4
TLP626	5	600	55		5	PDIP 4
TLP627	9	1000	300		5	PDIP 4
TLP651	2	10			5	PDIP 8
TLP721	5	600	55		4	PDIP 4
TLP731	1	600	55		4	PDIP 4

ТИРИСТОРНЫЕ ОПТОПАРЫ						
Наименование	СХЕМА №	Uкомм,пик., В	Iсраб.вх., мА	Zero-Cross*	U из. кВ	Тип корпуса
MOC3020	10					PDIP 6
MOC3021	10	400	15		7.5	PDIP 6
MOC3023	10	400	5		7.5	PDIP 6
MOC3041	11	400	15	+	7.5	PDIP 6
MOC3042	11	400	10	+	7.5	PDIP 6
MOC3061	11	600	15	+	7.5	PDIP 6
MOC3062	11	600	10	+	7.5	PDIP 6
MOC3063	11	600	5	+	7.5	PDIP 6

* **Zero-Cross:** схема управления переключением (открыванием симистора) в момент перехода фазы через ноль.



ТИПЫ КОРПУСОВ

Литература

В процессе написания проекта использовалась следующая литература.

1. Тигран Калайджян "Работа с внешними устройствами с помощью портов ввода\вывода". Москва. 2002г.
2. Ralf Brown Interrupt List. Release 61.
3. Pei An «PC Interfacing. Practical Guide to Centronic RS232 and Game Ports» Newnes
4. Каталог товаров фирмы «Чип и Дип»