Работа с портами COM и LPT на низком уровне.

Ключевые слова: Работа с портами КОМ и ЛПТ на низком уровне, Работа с портами КОМ и ЛПТ на высоком уровне, Работа с портами COM и LPT на высоком уровне, программирование портов, программирование КОМ порта, программирование COM порта, управление COM портом, программирование ЛПТ порта, программирование LPT порта, управление LPT портом, применение АПИ функций, применение API функций, вывод информации через порты, ввод информации через порты, работа с портами ПК, работа с портами PC, зажечь лампочку от порта, управление реле портом компьтера, работа с АЦП через порт, управление АЦП портом, ADC и компьютер, Binary Coded Decimal, BCD представление, преобразование в BCD, захватить мир.

Введение.

Данная статья ни в коей мере не претендует на звание учебного пособия! Многие понятия, приведённые в ней, весьма условны, а многие – совсем условны ☺. Целью написания данной статьи было не желание рассказать о великом и загадочном мире внутренностей ПК, а объяснить в максимально понятных терминах как при помощи клавиатуры и языка программирования заставить выставиться нужный уровень на нужном выводе нужного порта. Ведь все мы начинали со счётных палочек в 1 классе, и лишь потом узнали, что есть области натуральных, комплексных чисел и деление на ноль вовсе не невозможно ;-)

Так вот эта статья и есть те самые палочки. Если кто-то желает залезть в «высшую математику» и изучить все тонкости обращения к портам ПК, путь ему к соотв. литературе или, хотя бы, к списку оной в конце данной статьи.

Архитектура ПК.

Сначала определимся, что же такое работа на высоком и на низком уровне.

Под работой на высоком уровне понимают обращение к порту через ОС и под её контролем.

Дело в том, что в Windows обращение к любому устройству аналогично обращению к одноимённому файлу. То есть, если мы хотим считать данные из порта, например, COM1, то мы должны открыть файл с именем “COM1”. Для этого используются WinAPI функции OpenFile("COM1",,,,,), ReadFile, WriteFile, CreateFile и т.д. Их описание есть в «помощи» любого языка и в соответствующей литературе, в т.ч. и в **[1]**, по этому мы не будем на них подробно останавливаться. Эти функции являются функциями ОС, а не языка, по этому одинаковы во всех языках программирования. Отличается лишь форма их записи.

Под работой на низком уровне понимают непосредственное обращение к регистрам контроллера порта из адресного пространства. АП - это абстрактное понятие, выражающее отправную точку, относительно которой происходит адресация всех устройств компьютера.

Работа на высоком уровне позволяет легко реализовывать протоколы и функции, заложенные разработчиками ОС, но затрудняет реализацию собственных. Работа на низком уровне напротив, открывает широкие возможности для творчества, но реализация более-менее сложного протокола может потребовать знания ассемблера, много времени и, скорее всего, будет нагружать процессор лишними функциями. Кроме того, доступ к некоторым регистрам (а след. и ножкам) порта невозможно (без специальных знаний) получить из адресного пространства и наоборот, некоторые комбинации выводов невозможно «зажечь» с помощью API. Таким образом, работа на высоком и на низком уровне, это не взаимоисключающие понятия, а всего лишь разный подход к управлению портом.

Рассмотрим структуру стандартного порта.

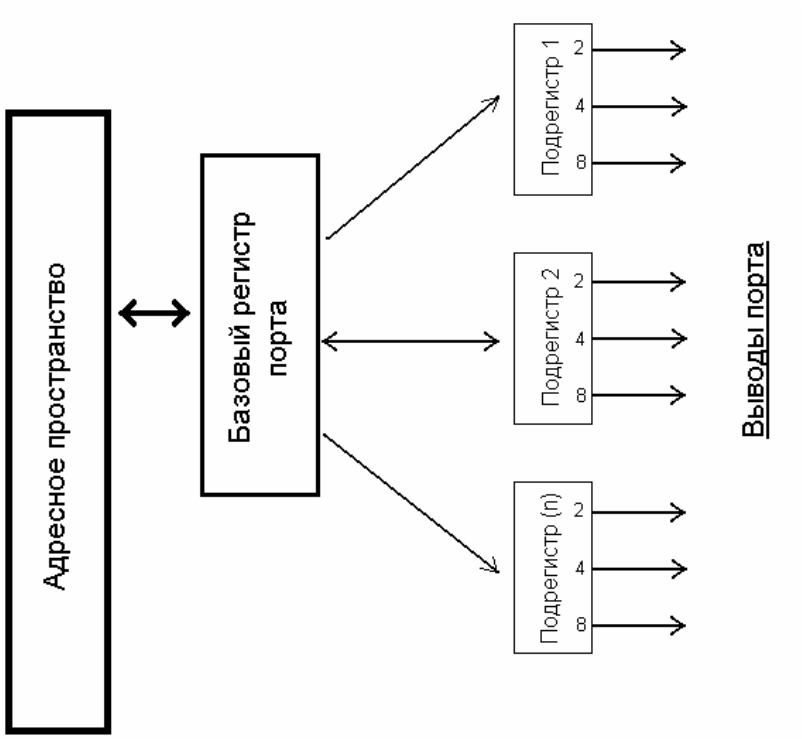


Рис.1 Структура стандартного порта.

Этот макет наиболее наглядно представляет структуру стандартного порта ввода/вывода, к которым относятся COM и LPT.

Как видно, порт содержит базовый регистр – это первый этап на пути к нужной ножке из АП. Базовый регистр содержит в себе несколько подрегистров (ПР) (называемых ещё «смещением адреса»). В зависимости от их типа, связь их с БР может быть двусторонней (запись и чтение) или односторонней (запись или чтение).

Каждому ПР соответствует какое то количество ножек порта. Т.е. разные ножки порта принадлежат разным подрегистрам и для того, чтобы выставить на них какое-то состояние или считать его с какой нибудь ножки, надо сначала обратится к соотв. ей ПР.

Некоторые ПР вообще не имеют непосредственного отношения к выводам порта, т.к. являются регистрами состояния. Их используют для настройки порта. Например, некоторое количество ножек порта могут работать на передачу/приём. Для того, чтобы установить режим их работы (передача, приём или передача и приём) в соотв. ПР порта записывают определённый код. Ориентируясь на него, контроллер порта устанавливает соотв. режим работы этих ножек.

Каждой ножке в ПР соответствует весовой коэффициент в двоичном коде. Т.е. для того, чтобы выставить “1” на каком то выводе порта, нужно записать в соответствующий ей подрегистр её весовой коэффициент. Напротив, при считывании состояния входной линии порта, порт вернёт не состояние “0” или “1”, а “0” или “N”, где N – весовой коэффициент данной ножки. Возвращение портом “0” означает, что на входе “0”. Возвращение же “N” означает, что на входе “1”.

Указанные действия в языке Паскаль 7.0 выполняет процедура port[]. Формат её записи следующий: **port[b+n]:=m;** Где:

b – адрес БР порта.

n – номер подрегистра.

m – весовой коэффициент вывода.

Попробуем с её помощью установить “1” на выводе RTS(7) COM порта.

Для этого обратимся к **таблице 1**:

Сигнал.	Контакт разъёма DB-9S.	Направление.	Подрегистр.	Вес.
TxD	3	Передача.	X	X
RxD	2	Приём.	X	X
DTR	4	Передача.	4	1
DSR	6	Приём.	6	32
RTS	7	Передача.	4	2
CTS	8	Приём.	6	16
DCD	1	Приём.	6	8
RI	9	Приём.	6	64
GND	5	Земля.	X	X

Таблица 1. Назначение контактов COM порта (разъём DB-9S.)

Как следует из таблицы, выводу RTS соответствует 7 контакт стандартного девятиконтактного COM порта. Он относится к 4 подрегистру и имеет в нём вес равный 2.

Адрес базового регистра, это адрес порта. Для COM1 он равен $3F8 (для COM2 - $2F8, для LPT1 - $378, LPT2 - $278).

Таким образом, мы можем записать:

```
program prim1;
uses crt,dos;
var
base: integer;

begin
base:=$3F8;  {Присвоение переменной base баз. адреса}
port[base+4]:=2;   {Установить RTS}
readln;          {Ждать нажатия Enter}
port[base+4]:=0;   {Сбросить все выходы порта}
end.
```

Эти строчки приведут к установке +12в на выводе 7 порта.

Сразу напрашивается вопрос - как запитать несколько выходов сразу. Ответ - просто сложить коды выходов. Например, мы хотим «зажечь» выходы RTS и DTR. Тогда пишем:

```
port[base+4]:=3;
```

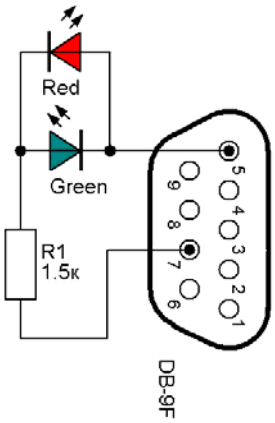
Сбросить все выходы порта можно отправив туда 0, а вот для того, чтобы сбросить конкретную ножку, оставив остальные без изменения, требуется отправить в порт значение, логически обратное весу сбрасываемого вывода. Например, для RTS это будет выглядеть так:

```
port[base+4]:=port[base+4] and (not 2);
```

Для экспериментов с портом удобно собрать индикатор, состоящий из вилки DB-9F и припаянных к ней светодиодов последовательно с резистором на 1.5 кОм между GND и нужным выходом (**Рис. 1**). Зачем два светодиода? Просто для COM порта логической единице на входе приемника соответствует уровень напряжения -3 ... -12 В. Логическому “0” соответствует напряжение +3 ... +12 В. Между уровнями +3 ... -3 В существует зона нечувствительности, обуславливающая гистерезис приемника. Состояние на выходе приемника изменяется только при пересечении напряжением порога +3 или -3 В. Таким

образом, в отличие от LPT, где используются ТТЛ уровни (уровень “1” при токе нагрузки 14 мА, не менее +2.4 В, уровень “0” при токе нагрузки 14 мА, не более +0.4 В) одной полярности, у COM-а полярность сигнала меняется от -12 до +12 вольт. При этом ток нагрузки на один выход COM порта не должен превышать 10 мА.

Но это ещё не всё. Логика в COM порту ещё и инверсная! Это значит, что активным уровнем у неё считается ноль, а пассивным – единица. Таким образом, «зажигая» вывод, мы устанавливаем на нём “0” (+12в), а «сбрасывая» - “1” (-12в). Это нужно учитывать.



Теперь разберёмся, как читать состояние входов.

Как уже говорилось выше, порт возвращает “0”, если на входе “0”, и “N”, если на данном входе “1”. Причём в некоторых случаях (особенно, при работе с LPT) “N” может быть равно сумме кодов ножек, если “1” присутствует не на одном входе.

Осуществляется приём следующим образом:

```
e:=port[b+n] and m;
```

Где:

e – переменная типа byte, в которую сохраняют принятое значение.

b – адрес БР порта.

n – номер подрегистра.

m – весовой коэффициент входа (см. **таблицу 1**).

Для проверки этой записи соединим перемычкой выход RTS(7) и вход CTS(8) и запустим приведённый ниже код:

```
program prim2;
uses crt,dos;
var
base,result: integer;
e:byte;

begin
base:=$3F8;
port[base+4]:=2;    {Установить RTS}
e:=port[base+6] and 16;  {Присвоить e состояние CTS}
if e=16 then result:=1   {Если e=16, то result=1}
        else result:=0;  {...иначе result=0}
writeln('Принятое состояние = ',result); {Вывод результата}
readln;           {Ждать нажатия Enter}
port[base+4]:=0;    {Сбросить все выходы порта}
end.
```

На экране должно появиться сообщение: «**Принятое состояние = 1**».

Отправляя в порт значения от 0 до 3 можно видеть, как изменяется состояние на любом выходе.

К сожалению, работать таким образом с выводами TxD и RxD нельзя, т.к. они относятся к асинхронному приёмопередатчику, встроенному в контроллер порта. Работает он по протоколу RS232 и управление им куда боле сложное.

Настало время LPT.

Приёмы работы с ним абсолютно такие же, однако надо помнить, что LPT способен работать аж в 4 режимах (Centronics, SPP, EPP, ECP) и их вариациях. Причём в каждом

режиме назначение и активный уровень выводов могут отличаться. Режим, устанавливающийся ОС при загрузке по умолчанию - SPP (Standart Parallel Port). Переключать режимы из своей программы можно записывая “1” в соотв. биты управления.

Сигнал.	Контакт вилки DB-25F.	Направление.	Подрегистр.	Вес.
DATA 0	2	Передача (SPP).	0	1
DATA 1	3	Передача (SPP).	0	2
DATA 2	4	Передача (SPP).	0	4
DATA 3	5	Передача (SPP).	0	8
DATA 4	6	Передача (SPP).	0	16
DATA 5	7	Передача (SPP).	0	32
DATA 6	8	Передача (SPP).	0	64
DATA 7	9	Передача (SPP).	0	128
Error	15	Приём.	1	8
Select	13	Приём.	1	16
PaperEnd	12	Приём.	1	32
/Busy	11	Приём.	1	128
Ack	10	Приём.	1	64
/Selectln	17	Передача.	2	8
Init	16	Передача.	2	4
/AutoLF	14	Передача.	2	2
/Strobe	1	Передача.	2	1
GND	18-25	Земля.	X	X

Как видно, порт имеет 3 ПР. [base+0], или просто [base] – это основной регистр порта. Служит для вывода информации на ножки DATA0(2) – DATA7(9). Как и в COM, для того, чтобы зажечь конкретную ножку, нужно послать в [base] её код. Несколько – сумму кодов. Это очень удобно, т.к. порт сам преобразует десятичное число (в диапазоне 0-255) в обычный 8-и разрядный код. Таким образом, к выводам порта можно напрямую подключать ЦАП-ы, микросхемы-драйверы 7-и сегментных индикаторов и т.д.

Например, подключив к LPT микросхему КР514ИД1(ИД2) можно выводить числа на внешний 7-и сегментный дисплей. Пример такой схемы можно увидеть здесь: (http://ra9wof.qrz.ru/files/Shemes/Digital/kr514id.htm). Простейший код для этой схемы такой:

```
program prim3;
uses crt,dos;
var
base,data: integer;

begin
Randomize; {Инициализация генератора сл. чисел.}
base:=$378;
repeat  {Повторять…}
  data:=random(9);  {data = сл. число от 0 до 9}
  Port[base]:=data;  {Отправить data в порт}
  Delay(600); {Задержка.}
Until KeyPressed; {…пока не нажата любая клавиша.}
Port[base]:=0; {Сбросить все выводы.}
End.
```

Данная программа генерирует случайное число от 0 до 9 и отправляет его в порт. Порт выдвигает на выводы его 8-и разрядный код (но мы используем лишь первые 4 разряда), который преобразуется микросхемой-драйвером в код для семисегментного индикатора, который и отображает его в привычной нам десятичной форме.

Микросхем может быть и две (входы первой м-мы подключают к разрядам 0-3, второй – к 4-7). Тогда можно выводить числа от «00» до «99». Только в этом случае лучше применить микросхемы К176ИД2(ИД3) или аналогичные, без псевдографики.

Прежде чем выводить числа на двухразрядный дисплей, их надо преобразовать в BCD (Binary Coded Decimal) представление.

Например, если требуется на индикаторе показать число 16 (десятичное), то в порт надо записать 0x16h, что является представлением числа 16 (десятичное) в формате BCD. Как очевидно, 0x16h = 22 (десятичное), что не одно и то же.

Например: (число в формате integer -> код для записи в порт)

0x09h -> 0x09h
0x0Ah -> 0x10h
0x0Bh -> 0x11h
...
0x10h -> 0x16h
0x11h -> 0x17h
...
0x16h -> 0x22h
и т.д.

Указанная операция в pascal/delphi выполняется следующим образом:

```
Port[base]:=((data div 10) shl 4) or (data mod 10);
```

Где **data** – исходный байт.

Подрегистры [base+1] и [base+2] являются служебными. Первый - для чтения битов состояния, второй - для записи битов управления. Работают с ними так же, как с RTS и CTS COM-а. Единственной особенностью [base+2] является то, что 4 (16) и 5 (32) биты являются настроечными. Если Ваш компьютер поддерживает стандарт EPP (все компьютеры, начиная с Пентиум1), то чётвёртым битом Вы сможете разрешить прерывание от принтера (“1” - разрешить прерывание по спаду ACK(10)), а пятым битом перевести линии порта d0-d7 в режим принятия данных (“1” - режим ввода, “0” - режим вывода).

Значок «слеш» (/) перед наименованием сигнала означает, что данный сигнал инвертирован. Для инвертированных линий активным уровнем считается “0”, а пассивным “1”.

Так, выходы 1, 14, 17 порта являются инвертированными. Это значит, что по умолчанию на них высокий уровень напряжения. На выходе же 16 по умолчанию уровень низкий. Отправляя в этот ПР весовые коэффициенты инвертированных ножек мы будем «гасить» соотв. вывод. “Ноль”, посланный в порт, «зажгёт» все выходы (кроме 16), а число “15” (битов то всего 4) изменит состояния на противоположные.

Аналогично для входов: при отключённом передатчике на инверсных входах будет держаться “1”, а на прямых “0”.

В любом случае, не пренебрегайте инициализацией! Какое бы состояние ни должно выставиться на выходах по умолчанию (даже если оно Вас устраивает), хорошим тоном программирования является «ручная» установка этого состояния из программы при её старте (т.к. по целому ряду причин начальное состояние выходов порта разных

компьютеров может отличаться). Это существенно повышает надёжность проектируемой системы.

Также следует отметить, что вход ACK соединен с питанием +5 В через резистор 10 кОм. Это сделано для исключения ложных прерываний, т.к. прерывание генерируется по отрицательному перепаду сигнала на входе ACK.

Если знания позволяют программировать в Дельфи, то для управления портом можно использовать ассемблерные вставки. Вот пример кода, отправляющего в LPT число и считывающего его от туда (не путать с приёмом данных – идёт простое считывание числа, которое нами же и было записано):

```
asm
MOV DX,0378H    ;DX теперь содержит адрес порта.
MOV AL,data     ;data - то, что послать. Тип byte.
OUT DX,AL       ;Запись байта в регистр данных.
IN  AL, DX      ; Читаем регистр данных.
end;
```

К слову сказать, запись [$378+1] эквивалентна записи [$379]. Так что в указанном коде можно сразу указывать адрес к нужному ПР и не производить лишнее сложение программно.

Можно поступить и проще. Для Дельфи был разработан модуль vv55rWin, содержащий процедуры работы с портами, написанные на ассемблере указанным выше способом. Его (а так же его исходник) можно найти в папке Progy\Unit.

Называются эти процедуры **inport(port)** и **outport(port,aa,data)**. Где:

port – полный адрес к нужному ПР.

data – информация, отправляемая в порт.

aa – зарезервированный параметр, всегда должен быть нулевым.

Например, программа **prim2** в переводе на «Дельфийский» примет вид:

```
var
aa,e:word;
port,base,data:word;
{.........}
aa:=0;
base:=$3F8;

begin
   port:=base+4;
   data:=2;
   outport(port,aa,data);
       sleep(10);
       port:=base+6;
       e:=inport(port) and 16;
    port:=base+4;
    data:=0;
    outport(port,aa,data);
end;
```

Пример взят из **[4]**.

Управлять конкретными выводами порта из под Дельфи можно и с помощью API. Это хоть и сложнее, зато такая программа будет работать на любом Windows, в т.ч. на NT, 2000, XP (приведённые выше примеры работоспособны лишь в 95/98/ME). Очень хороший пример тому – исходник из **[5]**.

Вот и всё по работе на низком уровне.

В заключение отмечу, что подробнее про структуру портов (и программирование их на ассемблере) можно прочитать в **[2]**, про работу на высоком уровне (API) в **[1]**, а в **[3]** есть примеры программ на С.

Практическая реализация.

Не раз на страницах журналов и в сети встречалась идея применения лазерной указки в охранной технике. Было предложено немало схем блоков контроля, однако ни одна из них не сравнится с возможностями даже самого древнего 286 компьютера. Простая приставка к COM порту компьютера и несложная программа под ДОС способны создать неприступный барьер даже для сравнительно хорошо экипированного взломщика.

Принцип работы системы таков: программа в цикле, с помощью функции Randomize, вычисляет случайное число. Это число преобразуется в N разрядный двоичный код, который сохраняется в N разрядном массиве. Далее, этот массив побитно читается и биты передаются через выход RTS (и далее – через луч указки) на фотоприемник, сигнал с которого усиливается элементами DD1.4 – DD1.3, подключёнными ко входу CTS COM порта. Далее код снова преобразуется в десятичную форму и сравнивается с отправленным. Если равенство не выполняется M раз подряд, программа выставляет “1” на выходе DTR (и на RTS, чтобы луч во время тревоги не гас), к которому подключено исполнительное устройство (сирена, пиропатрон и т.п.) на время T.

Константы N, M и T можно изменять в широких пределах. Например, чем больше N, тем больше возможных комбинаций будет иметь код (их число определяется по формуле 2^N), но тем больше времени потребуется на передачу N бит, ведь частотные возможности светодиода указки не безграничны, да и фотодиоды имеют достаточно большое время восстановления после засветки.

Таким образом, перекрытие луча или засвечивание фотоприёмника взломщику не поможет. Также, становится невозможным подбор кода для подмены, ведь он меняется по случайному закону. Единственный способ нелегально отключить систему – добраться до компьютера.

Схема приставки изображена на **рис. 2:**

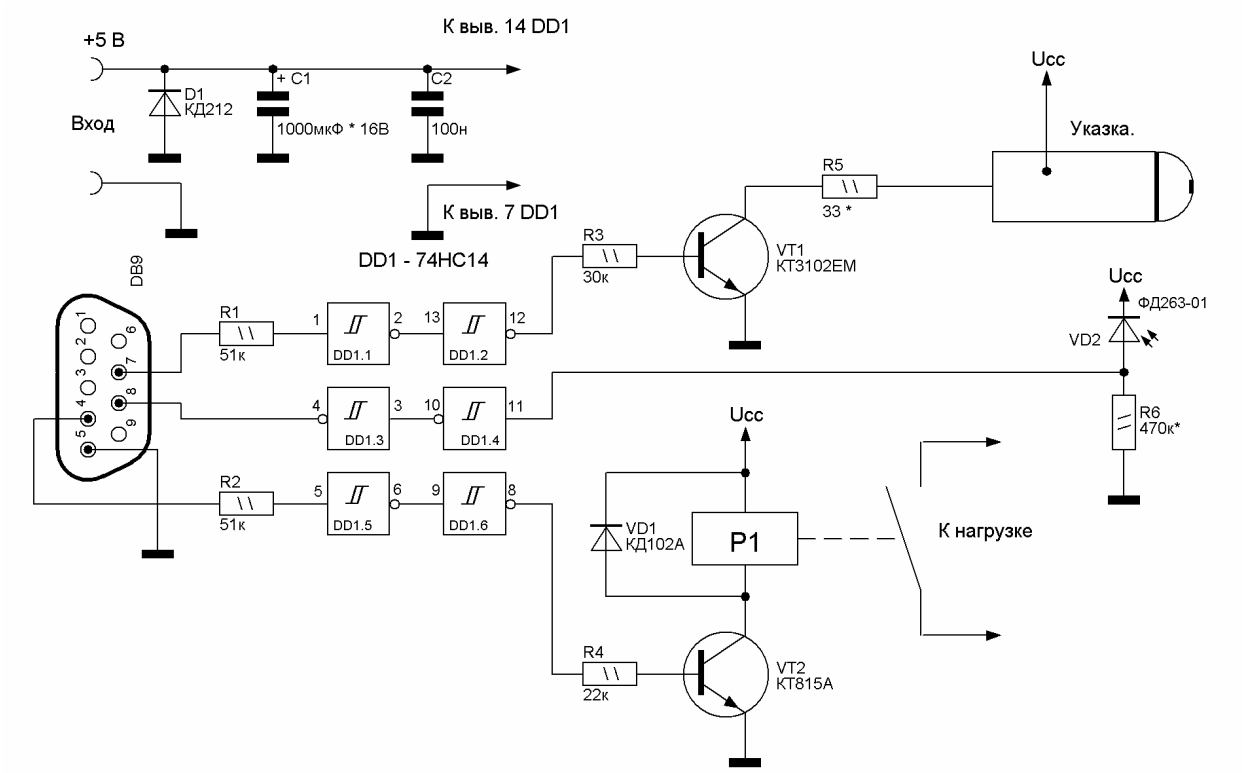


Рис. 2. Принципиальная схема приставки.

Микросхема DD1 выполняет сразу несколько функций – преобразователя уровня (за счёт свойств её входных и выходных каскадов), усилителя для фотодиода и фильтра помех (т.к. имеет входной гистерезис 400 мВ). Резистор R5 подбирают исходя из требуемой яркости свечения лазера. С одной стороны, чем ярче он светится, тем чётче реагирует на него фотоприёмник. С другой же стороны, чем больший ток течёт через светодиод, тем быстрее идёт его деградация. Учитывая невысокое качество изготовления китайских указок, такой излучатель прослужит недолго.

Для питания у-ва можно использовать сам компьютер, подключив вход платы к свободному шлейфу питания БП (контакты +5В и GND). Если доступа внутрь компьютера нет, +5В можно взять с гнезда USB, однако нагрузка на него не должна превысить 1А.

Приставка может работать и совместно с LPT. В этом случае номиналы R1 и R2 следует уменьшить до 100 Ом.

Все резисторы – МЛТ 0,125. Конденсатор С1 – К50-35 или аналог. С2 – любой керамический однослойный. Полный аналог микросхемы 74НС14 – КР1564ТЛ2, Диод VD1 заменим на КД522, транзистор КТ3102ЕМ – на любой другой npn с током коллектора не менее 100 mA, а VT2 – 200 mA (зависит от тока реле). Защитный диод D1 – на КД213, 1N4001 – 1N4007. Фотодиод – любой, надо лишь подобрать номинал R6 для надёжного срабатывания у-ва на необходимом расстоянии. Реле – любое, срабатывающее от 5 вольт. Например, РЭС55А.

Если расстояние между указкой и фотодиодом предполагается больше 10м или просто наблюдается нечёткая работа приёмника, у-во можно снабдить усилителем, схема которого показана на **рис. 3.**

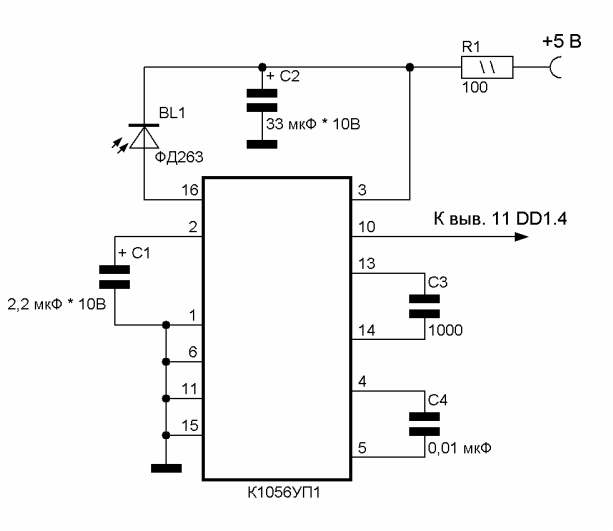


Рис. 3. Схема усилителя.

Микросхема К1056УП1 – специализированный усилитель для фотодиодов. Заменима на К1054УП1 или TBA2800 с учётом немного иных схем включения.

Текст программы для этой приставки выглядит так:

```
program security;
uses crt, dos;
 const n=8;       {число бит}
 const m=3;       {число допустимых пропусков}
 const b=$3F8;    {базовый адрес порта}
 const t=1000000;  {время работы сирены (мкС) }

 type mas=array[1..n] of integer;

var
 a:mas;
 i,alarm,ch_al,s:integer;
 e:byte;
 Fr,temp:real;
 rdm,x,d:integer;

PROCEDURE DelayMks(Del:longint);  {Процедура-таймер}
assembler;
asm
mov ah,86h   {ф-я микросекундной паузы BIOS}
mov dx,word ptr Del   {Del - величина паузы}
mov cx,word ptr Del+2
int 15h
end;

 BEGIN  {начало}
 clrscr;
 writeln('Охранная система включена!');
 writeln;
 {инициализация ф-й и переменных}
 Randomize;
 x:=0; temp:=0; Fr:=0; alarm:=0; ch_al:=0;
 port[b+4]:=0;
```

```
 {вычисление предела значения, возвр. ф-й Randomize}
 rdm:=round(exp(n*ln(2))-2);

 repeat {начало цикла}
 x:=random(rdm); {получение сл. числа}
 temp:=x;
 {цикл преобр. x в n разрядный код}
 for i:=1 to n do
  begin
  Fr:=Frac(temp/2);
  temp:=Int(temp/2);
    if Fr=0
    then a[i]:=0
    else a[i]:=1; {код записан в массив}
  end;
    {передача/приём/дешифрация кода}
  d:=0;
  for i:=1 to n do
    begin
    if a[i]=1 then port[b+4]:=2
              else port[b+4]:=0;
    DelayMks(10000);  {задержка между битами}
    e:=port[b+6] and 16;
    if e=16 then
    d:=round(d+exp(((n-1)-(n-i))*ln(2)));
    end;
    {d - принятое десятичное значение}

  port[b+4]:=0;

  if d<>x then alarm:=alarm+1   {фильтр помех}
          else alarm:=0;

  if alarm>=m then   {если пропусков >= m}
    begin
    port[b+4]:=3; {Зажигаем RTS и DTR}
    DelayMks(t);
    port[b+4]:=0;
    ch_al:=ch_al+1;
    alarm:=0;
    end;

 until KeyPressed;  {конец цикла}
 {подготовка к закрытию и формирование отчёта}
 port[b+4]:=0;

 if ch_al=0 then writeln('За время работы срабатываний небыло.')
 else writeln('Система срабатывала ',ch_al,' раз!');

 readln;
END.
```

После запуска программа выдаёт сообщение «Охранная система включена!» и начинается контроль объекта. Число генерируется и передаётся через луч. В случае трёхкратного (подряд) принятия неверного кода, на время T выставляется высокий уровень на выходе DTR(4) и RTS(7). По истечении периода T работа программы возобновляется.

Следует обратить внимание на задержку между битами – она необходима для обеспечения необходимой паузы между сменой состояния на выходе RTS(7), дабы фотоприёмник успевал надёжно реагировать на изменения луча. Величина её зависит от многого: какова яркость лазера (устанавливается резистором R5), расстояние между лазером и фотодиодом, типа фотодиода и, конечно же, какая скорость передачи битов необходима для конкретного случая. Подбирают задержку экспериментально, уменьшая до предельного значения, когда система ещё работает стабильно (здесь следует также учитывать возможные изменения прозрачности воздуха в течение охранного периода). Если всёже скорости недостаточно для передачи 3 чисел в течение ½ – 1 сек. (а именно на такое время пересекает луч идущий человек), имеет смысл уменьшить число бит (понизить разрядность).

Остановить программу можно, нажав любую клавишу на клавиатуре (однако ничто не мешает установить пароль), после чего на экран будет выведен отчёт с информацией о числе срабатываний за охранный период. Нажатие Enter окончательно закроет программу.

В программе используется процедура DelayMks, написанная на ассемблере. Она необходима для более-менее точного замера временных интервалов, т.к. «родная» паскалевская Delay очень непредсказуемо работает на процессорах семейства Пентиум. Вникать в её суть не обязательно, достаточно просто вставлять её в текст своей программы.

Программа работает на один канал, однако не трудно переделать её на любое количество каналов. Ведь многие старые компьютеры имеют до 4 портов LPT. Легко посчитать, сколько линий контроля получится сделать. При этом на каждую линию можно вести протокол, записывая события и время в текстовый файл. Можно подключить к порту АЦП **[4],[6]** и следить за параметрами некоего процесса (например, температурой в оранжерее) и регулировать этот параметр программно, при этом отображая его на экране в виде графика или диаграммы, параллельно записывая эти данные в файл на жёстком диске. Можно сделать автоматический станок, например, для гравировки, и управлять им от компьютера. Возможности ограничены лишь Вашей фантазией!

Исп. Литература:

1) Работа с коммуникационными портами (COM и LPT) в программах для Win32. **http://bcb.net.ru/article/hard/index003.html**

2) Коммуникационные порты персонального компьютера. **http://electronics.org.ua/techinfo/lpt/lpt.htm**

3) Основы программирования LPT для DOS и Win9x. **http://radiopirat.h11.ru/prog/prog004.htm**

4) Примеры работы с АЦП. **http://dikoy44.narod.ru/photoalbum.html**

5) **Вакуленко А.** Программа LPTtest. – Радио, 2004, №8, с. 23,24.

6) **Патрик Гёлль.** Как превратить ПК в измерительный комплекс. ДМКпресс, 2002г.

В настоящее время почти готова статья по работе с портами с использованием Delphi or C++Builder и работой в WinXP, но, если на сайте её нет, значит на данный момент я её так и не дописал – в силу занятости или лени ☺ Так что пишите вопросы в мыло.

Все вопросы и пожелания направляйте на мыло nm1456t01@yandex.ru , nm1456t01@rambler.ru или dikoy44@yandex.ru . А лучше на все ящики сразу ;-)

WBR Dikoy, http://dikoy44.narod.ru/