Программирование параллельного порта на уровне PIN'ов средствами Lazarus в Linux.

Сергей Иванчиков
http://freepascal.ru/

19 января 2005 г.

Содержание

Для чего это надо?

Например подключить цветомузыку, или создать программируемое устройство, да мало ли для чего ещё может приспособить LPT порт радиолюбитель. Например на masterkit есть конструктор *'набор реле для для подключения к порту LPT'*, после чего получаем готовое 8-канальное устройство коммутации нагрузки.

Что для этого надо?

Конечно, необходим установленный Lazarus, с root-правом запуска среды, например через команду `SUDO`. Исполняемый файл также нужно будет запускать через `SUDO`, естественно, если не работать от root'а.

Приступаем

Для начала необходимо разобраться с цоколёвкой порта LPT, ведь необходимо же знать на каком выводе порта ловить сигнал.

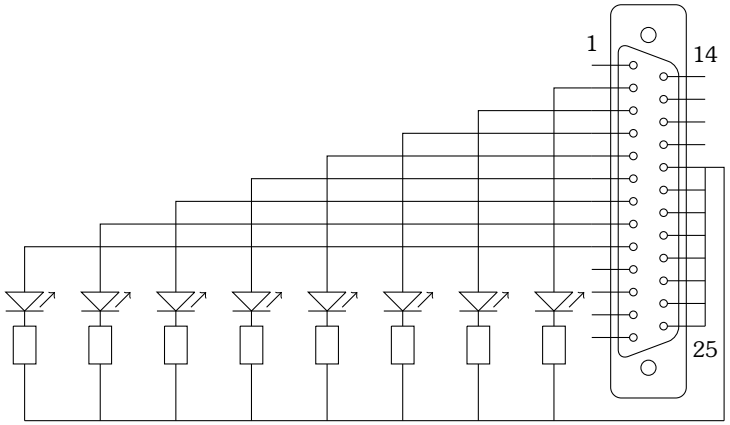

Рис. 1: Схема выходов

Для снятия сигнала используются выводы 2–9 порта, резисторы можно не использовать, (лучше все же использовать сопротивление около 400 Ом, во избежание сгорания порта) при использовании резисторов выше 600 Ом свечение светодиодов заметно уменьшится.

Линии порта с 2 по 9 соответствуют байту, причём вывод 2 младший бит, а вывод 9 старший бит. Каждому биту соответствует свой вес в двоичной системе счисления:

- линия 2 — бит 0 — 1
- линия 3 — бит 1 — 2
- линия 4 — бит 2 — 4
- линия 5 — бит 3 — 8
- линия 6 — бит 4 — 16
- линия 7 — бит 5 — 32
- линия 8 — бит 6 — 64
- линия 9 — бит 7 — 128

Для того, чтобы зажечь светодиод номер 1 необходимо заслать в порт значение 1, чтобы зажечь светодиод номер 2 необходимо заслать в порт значение 2, а если хочется зажечь одновременно светодиоды 1 и 2 засылаем в порт значение 3.

То есть, переводим двоичный код в десятичный: `0000 0011 = 3`. Единицы в данном случае соответствуют включенным светодиодам. Программно это реализовать не сложно[1].

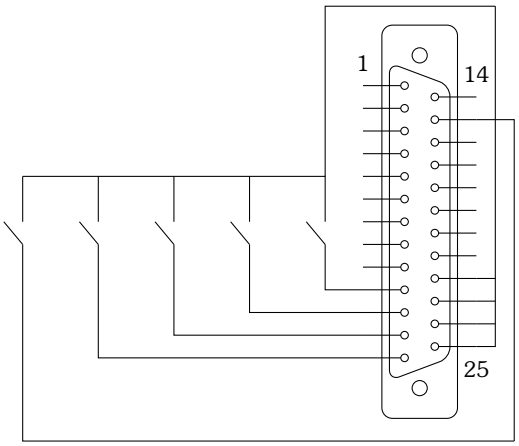

Рис. 2: Схема входов

Для того чтобы считывать данные из порта используются линии 10, 11, 12, 13, 15 порта при замыкании на 0 этих выводов, будет изменяться содержимое порта по адресу `$379`. Тут также значение байта со своими битами.

- PIN 11 — бит 7 — 128
- PIN 10 — бит 6 — 64
- PIN 12 — бит 5 — 32
- PIN 13 — бит 4 — 16
- PIN 15 — бит 3 — 8

Программная реализация

Программная реализация на самом *деле* очень проста: для начала необходимо объявить файловую переменную типа `TfileStream`, после этого открыть файл устройства, сместиться от начала файла на величину `$378`, равную адресу порта, записать туда значение, закрыть файл. Всё.

[1]Начиная с того, что Free Pascal позволяет указать число в двоичной системе счисления ("`%xxxxxxxx`", где "`x`" — ноль или единица), и заканчивая множествами (например, "**`set of`** `0..7`"). — *прим. ред.*

Пример

бросаем на форму timer и парочку edit'ов

```
. . .
var
  Form1: TForm1;
  f:TfileStream; //
  n:integer;
  ch,a:byte;

implementation
. . .
procedure TForm1.Timer1Timer(Sender: TObject);
 begin
 n:=$378;
 ch:=strtoint(edit1.Text);
 f:=TFileStream.Create('/dev/port',fmOpenReadWrite);
     // открываем файл для записи и чтения
 f.Seek(n, soFromBeginning); // смещаемся на величину n
 f.Write(ch,1); // записываем в порт величину введенную в edit1
 f.Seek(n+1, soFromBeginning);
     // смещаемся еще на 1 относительно n
 f.Read(a,1); // читаем байт по адресу n+1
 f.Free; // освобождаем память
 edit2.Text:=inttostr(a);
     // в edit2 пишем значения байта по адресу n+1
 end;
. . .
```

В примере не показан обработчик ошибок в случае некорректного ввода значений. При проведении эксперимента ни один порт ни на одном компьютере не сгорел.

PS: Удачи в экспериментах и пусть порты остаются не сгоревшими!