

## Re: Signal generation on parallel port

**Source:** <http://linux.derkeiler.com/Newsgroups/comp.os.linux.questions/2004-11/0025.html>

---

**From:** MS. Mo ([mdfmk\\_at\\_perso.be](mailto:mdfmk_at_perso.be))

**Date:** 11/19/04

Date: Fri, 19 Nov 2004 20:26:07 +0100

On Fri, 19 Nov 2004 22:45:25 +1100

<HLHL> wrote:

> *I want to create square wave signal on parallel port with programmable*  
> *frequency between 10-100Hz.*  
>

Few times ago, I had to write a software that does control an old adc, connected to an old SPP parallel port. I didn't had to use EPP/ECP so I dunno if it works too, but you can check that by yourself with an oscillo :)

The || port is composed of 25 pins, and part of those pins can be controled separately by throwing bits in the parallel port registers.

in outgoing way,

1 mean +5V  
0 0V

first here's the pin assignation:

Pin No (D-Type 25) SPP Signal Direction In/out Register

1 nStrobe In/Out Control  
2 Data 0 OUT Data  
3 Data 1 OUT Data  
4 Data 2 OUT Data  
5 Data 3 OUT Data  
6 Data 4 OUT Data  
7 Data 5 OUT Data  
8 Data 6 OUT Data  
9 Data 7 OUT Data  
10 nACK IN Status  
11 BUSY IN Status  
12 PaperO/E IN Status  
13 Select IN Status  
14 nAutoLF IN/out Control  
15 nError/nFault IN Status  
16 Initialize IN/out Control

17 nSelect Printer/ nSelectin IN/out Control  
18–25 ground gnd

then to access the parallel port you'll have to use a special address called the BASE address usually located @ 378h.

To create a square wave, on a certain pin, all you have to do is writing a 1 at the corresponding bit in the right register. The correspondance bit <--> pin is shown below

in BASE+0 (Data register)

you can control all the data bits, bit 0 <--> Data 0 and so on

their main purpose is to send information to a peripheral device so you might use those to generate your wave.

in BASE+1 (status register)

Bit 7 Busy

Bit 6 Ack

Bit 5 Paper Out

bit 4 Select

bit 3 Error

bit 2 IRQ (tell the system to use IRQ or not)

bit 1 and 0 are reserved (?)

and the last

BASE+2 (Control port)

bit 7 and 6 are unused

bit 5 enable bidirectionnal mode

bit 4 enable irq via ack line

bit 3 select printer

bit 2 init printer

bit 1 auto linefeed

bit 0 strobe

so now you have all the register <--> physical pin association

The registers are used via 2 kernel functions, inb (read the register) and outb (write on it)

those are fast but require to run as root (other people use read/write on /dev/lp)

and require also to use ioperm to tell the kernel to allow the application to access the port.

here's an example:

```
#define BASE 0x378 /* || port base adress */
```

then use ioperm to grant the permissions:

```
int ioperm(unsigned long from, unsigned long num, int turn_on);
```

in your case you'll only use the base address

```
if(ioperm(BASE,1,1)) {perror("Ioperm b0rked"); exit (0);}
```

and then you can use

outb function to create your square with a loop

it might look like this:

```
char byte = 0;
int i;
for(i = 0; i < 1000; i++){
    outb(byte, BASE);
    byte ^= byte; /* alternate 0 - 1 */
    usleep(10000); /* use this for the frequency adjustments */
}
```

the time specified for usleep is in microseconds.

You might also have a look at beyond logic  
<http://www.beyondlogic.org/spp/parallel.htm>

> *Can one of the Linux guru guide me what is the best way of generation of*  
> *square wave with highest possible frequency stability regardless of the*  
> *process load on the computer?*  
>

Dunno if I'm clear, but  
I hope it helps :)

Mo