INTERFACE BOARD

VOOR DE

PARALLELLE POORT

concept verslag

Project: **Parallelle poort I/O Interface**
Auteur: **ing. K.H. Welling**
Email: **AdvancedVB@hotmail.com**

[titel pagina]

Samenvatting

Dit verslag is een samenvatting van het ontwerp proces van een interface board dat op de parallelle poort van de computer kan worden aangesloten. Hierdoor is het mogelijk om m.b.v. een computer programma een willekeurig systeem te kunnen sturen en/of hier data van te ontvangen voor eventuele analyse.

In dit verslag zijn de volgende onderdelen terug te vinden:

- Onderzoek naar:
    - werking & aansturing parallelle poort
    - software drivers voor de aansturing
- Ontwerp van:
    - Visual Basic class object
    - elektronica (interface board)
    - behuizing

INHOUD

Bijlagen

1 Inleiding

Om met een computerprogramma fysische processen te kunnen aansturen en/of informatie erover in te winnen, moet er data overdracht plaats vinden tussen de computer en een proces. Hiervoor kan de parallelle poort gebruikt worden. Aangezien de parallelle poort geen hoge vermogens kan leveren of ontvangen, is een interface board nodig, die deze vermogens kan omzetten, zonder dat de computer of het proces hier hinder van heeft.
Dit rapport beschrijft de ontwikkeling van een dergelijk interface board.

Hiervoor zijn de volgende stappen ondernomen:

1. Vooronderzoek:
    a. Parallelle poort
    b. Software (drivers etc)
2. Ontwerpen:
    a. Elektronisch circuit
    b. Omhuizing interface board
3. Budgettering:
    a. Totale kosten ontwerp

2 De parallelle poort

De parallelle poort (ook wel lpt-poort genoemd = Line PrinTer) is te vinden aan de achterkant van de meeste computers. Het type aansluiting is een D-Type 25 Pin connector (vrouwelijk).
Deze poort is een van de meest gebruikte poorten voor “thuisprojecten”. De poort geeft toegang tot 9 bits invoer of 12 bits uitvoer op elke willekeurig tijdstip.

De nieuwere parallelle poorten zijn sinds 1994 gestandaardiseerd onder de IEEE 1284. Deze standaarden beschrijven de volgende 5 modes:

1. Compatibility Mode
2. Nibble Mode
3. Byte Mode
4. EPP Mode (Enhanced Parallel Port)
5. ECP Mode (Extended Capabilities Mode

Het doel om nieuwe drivers en devices te ontwikkelen die downwards compatible zijn met de Standaard Parallelle Poort (SPP). De eerste 3 ondersteunen de standaard hardware, terwijl de EPP en ECP extra hardware ondersteunen, zodat de poort sneller werkt.

De “Compatibility Mode” (of “Centronics Mode”) kan alleen data verzenden met een snelheid van 50 tot 150+ kBytes/seconde. Om data te ontvangen, de parallelle poort moet omgezet worden naar “Nibble” of “Byte Mode”. De “Nibble Mode” kan alleen een input versturen van 4 bits. De “Byte Mode” gebruikt de *bi-directional* optie (wordt nauwelijks ondersteund). Hiermee kunnen 8 bits data terug naar de computer verstuurd worden.

De Extended en de Enhanced Parrallelle Poorten gebruiken extra hardware voor ‘*handshaking*’. Om met deze methodes een byte naar een printer te sturen, moet het volgende gebeuren:

1. stuur een byte naar de data poort
2. controleer of de printer bezig (busy) is. Als dat zo is, dan accepteert de printer geen data, dus de geschreven data is verdwenen.
3. deactiveer de *strobe* (Pin 1 = low). Dit verteld de printer dat er correcte data op de data lijnen aanwezig is (Pin 2 – 9).
4. na ongeveer 5 milliseconden, nadat de *strobe* op laag is gezet, wordt deze weer geactiveerd (Pin 1 = hoog).

Hierdoor wordt de snelheid van deze 2 typen dataoverdracht beperkt. De EPP & ECP poorten omzeilen dit door de hardware te laten controleren of de printer bezig is en zorgen zelf voor de *strobe* (of evt. ‘*handshaking*’). Hierdoor hoeft maar 1 I/O instructie te worden verzonden en versneld dit het proces tot ± 1 a 2 MegaBytes/senconde. De ECP poort maakt tevens gebruik van een *DMA-channel* en *FIFO* buffers. Hierdoor kan de data worden opgeschoven, zonder gebruik te maken van I/O instructies.

2.1 Hardware (IEEE 1284)

In Bijlage A: is een tabel opgenomen van de configuratie van de pinnen van de D-Type 25 pins en de Centronics 34 pins connector. De D-Type 25 pins is meestal aan de achterkant van de computer te vinden, terwijl de Centronics op de printer te vinden is.

De IEEE 1284 standaard specifeerd echter 3 poorten:

- Type A: D-Type 25 pins
- Type B: Centronics 36 pins
- Type C: 36 pins, zoals de Centronics maar dan smaller

Het type C heeft 2 extra pinnen om te bepalen of een apparaat eigen vermogen heeft.

De uitvoer van de parallelle poort is meetstal op basis van TTL logische waarden:

- Voltages:
    - 0 tot +0.8 Volt (Onwaar)
    - +2.4 tot +5v (Waar)

De stroom is gebaseerd op ASIC:

- Amperage: van 4 tot 16mA, officieel 12 mA. (*Sink/Source?*)

Het beste kan een relais worden gebruikt, zodat de stroom binnen de computer niet aangetast wordt.

2.2 Centronics

De Centronics is een vroegere standaard voor data transfer van host naar printer. De meeste printers gebruiken deze manier van '*handshake*'. Deze handshake wordt normaal toegepast door de software. (Ref. 1: Centronics) In de latere Extrended Capabilites poort, wordt een snelle Cenctronics poort gesimuleerd. De hardware neemt het standaard Centronics protocol over, het enigste wat de programmeur hoeft te doen, is een byte naar de lpt poort te sturen.

2.3 Port Adressen

De parallelle poort heeft 3 gebruikelijke poorten, zie tabel. Het 3BCh adres is tijdelijk voor lpt-poorten op video kaarten gebruikt en moet speciaal in de BIOS worden aangezet.

Adres:	**MDPA:**	**no MPDA:**
3BCh – 3BFh	LPT1	n/a
378h – 37Fh	LPT2	LPT1
278h – 27Fh	LPT3	LPT2
h: staat voor hexadecimaal getal		
MDPA: Monochrome Display & Printer Adapter		

Tabel 1: Standaard adressen parallelle poorten

Als de computer wordt opgestart, contronleerd de BIOS (Basic Input/Output System) het aantal poorten die aanwezig kunnen zijn (LPT1; LPT2; LPT3). Eerst wordt het eerste standaard adres gecontroleerd. Als hier een poort gevonden wordt, krijgt deze de naam LPT1. Vervolgens wordt op het volgende adres gekeken. Soms kan door jumper instellingen de poort namen worden vastgelegd. Daarom kan bovenstaande tabel in volgorde afwijken.

De adressen van de poorten kunnen ook altijd in de bios worden opgezocht op de volgende adressen:

Start adres:	**Functie:**
0000:0400	*COM1 adres(?) (**
0000:0402	*COM2 adres(?)*
0000:0404	*COM3 adres(?)*
0000:0406	*COM4 adres(?)*
0000:0408	**LPT1 adres**
0000:040A	**LPT2 adres**
0000:040C	**LPT3 adres**
0000:040E	*LPT4 adres(***
0000:0411	*Aantal gevonden poorten (volgens BIOS)*
(COM poorten kunnen soms meer dan 1 'word' groot zijn en dus de printer settings overschrijven.*	
*(** Adres: 0000:040E in de BIOS Data Area wordt gebruikt voor Extended Bios Data Area voor PS/2 en nieuwere BIOS-en.*	

Tabel 2 : Locatie COM & LPT adressen in BIOS

2.4 Software registers (SPP)

In Bijlage B: zijn 3 tabellen opgenomen met de software registers vanaf het basis adres. Het basis adres wordt gebruikt voor uitvoer naar de parallelle poort (pinnen 2-9). Dit register is normaal een 'write-only' poort. Als deze poort toch wordt gelezen, de laatst verzonden byte wordt geretourneerd. Behalve als de poort als Bi-Directional is ingesteld, kan dit adres ook data ontvangen vanaf een ander device.

De status register kan gevonden worden op basis + 1 byte. Alle data die naar deze poort geschreven wordt, wordt genegeerd. Denk eraan dat Bit 7 en Bit 2 omgekeerd zijn.

De control register kan gevonden worden op de het basis register + 2 bytes en was bedoeld als 'alleen schrijven' poort. Deze 4 poorten kunnen echter ook als invoer dienen. Als de computer een pin op hoog (+5V) heeft staan, zorg ervoor dat deze op 0 Volt komt. Dit wordt "open collector output" (of "open drain for CMOS devices") genoemd. Dit betekend dat deze 2 statussen heeft:

1. lage status (0V)
2. hoge status (5V) (="open circuit")

Er zijn 3 mogelijkheden om deze 4 'ingangen' aan te sturen:

1. 'internal pull-up resistors'
2. 'open collector outputs'
3. 'Totem pole outputs, the resistor will act as a load"

"An external 4.7k resistor can be used to pull the pin high. I wouldn't use anything lower, just in case you do have an internal pull up resistor, as the external resistor would act in parallel giving effectively, a lower value pull up resistor. When in high impedance state the pin on the Parallel Port is high (+5v). When in this state, your external device can pull the pin low and have the control port change read a different value. This way the 4 pins of the Control Port can be used for bi-directional data transfer. However the Control Port must be set to xxxx0100 to be able to read data, that is all pins to be +5v at the port so that you can pull it down to GND (logic 0)."
Lees ook Bijlage C: Hardware Schakelingen (Eng.).

Bits 4 & 5 zijn interne schakel opties. Bit 4 schakelt de IRQ en bit 5 schakelt de bi-directional poort modus.

2.5 Bi-Directional Ports

file:///D:/VISUAL%20BASIC/_VB_INFO/code/lpt-poort/more/parallel.htm#6

hier volgt nog meer info over….

3 Software

Om data te lezen van een parallelle poort of te schrijven naar een parallelle poort, moet eerst het adres (register) van de poort bekend zijn. (Lees hoofdstuk 2.3) Dit adres kan worden opgeslagen in een Integer (=2 bytes).
Vervolgens kunnen de bijv. 'data lijnen' in 1 byte gelezen of weggeschreven worden. Waarbij de 'high-bit' = pin 7 en de 'low-bit' = pin 0. (1 byte bestaat uit 8 bits.) Denk om de offset als de 'control' of 'status' byte gelezen worden. (Bijlage B:)

In DOS en Win3.1 konden alle applicaties direct naar de I/O poorten van de pc schrijven of lezen. Terwijl bij de win32 operating systems de bedoeling was, dat alleen device-drivers dit konden doen. Zoals ook in MS KB artikel Q112298 te lezen is, is dat in WinNT een "privileged instruction exception" foutmelding ontstaat als men buiten deze device-drivers om wil werken. Ondanks dat Win95 ook bedoelt is om via de divice-drivers te werken, wordt er geen foutmelding gegenereerd.

Om vanuit Visual Basic toegang te krijgen tot de poort, moet een driver gebruikt worden.

3.1 Drivers

3.1.1 Win95,98&ME

Ondanks dat toegang tot de parallelle poort onder Win95 geen fout melding ontstaat, dient men er wel rekening mee te houden, dat als een ander apparaat ook in verbinding staat met de parallelle poort, deze voorrang kan krijgen en zodoende lezen en schrijven onmogelijk maakt.

Onder Win95,98 of ME is de aansturing redelijk eenvoudig. Download 1 van de vele dll's die op internet te vinden zijn. Bijlage D: geeft een overzicht van enkele drivers.

3.1.2 Assembler

Poort instructies zijn ook niet terug te vinden in Visual C++ of Delphi, omdat directe poort aansturing niet wordt ondersteund. Om dit te kunnen omzeilen kunnen de volgende stukjes assembly code worden gebruikt:

```
BYTE inportb(UINT portid)
{
unsigned char value;
__asm mov edx,portid
__asm in al,dx
__asm mov value,al
return value;
}
```

Source 3-1: Assembly lezen I/O poort

```
void outportb(UINT portid,
BYTE value)
{
__asm mov edx,portid
__asm mov al,value
__asm out dx,al
}
```

Source 3-2: Assembly schrijven I/O poort

3.1.3 WinNT&XP

Windows 2000 & Windows XP ondersteunen helaas NIET de standaard dll's die onder voorgaande besturingssystemen wel werkten. Er zijn echter tricks & trucks om dit te verhelpen. Door een extra programma te gebruiken, wordt om de beveiligingen heen gewerkt. Het is echter wel noodzakelijk om zo'n programma eerst op te starten, alvorens er naar de dll uit paragraaf 3.1.1 wordt verwezen via de Visual Basic code. Zie ook hiervoor Bijlage D: .

4 Visual Basic LPT-Class

De software die voor aansturing van de parallelle poort wordt gebruikt is: MS Visual Basic 6.

Hier komt een beschrijving van de class die op zijn beurt weer gebruik maakt van de dll file. Hierdoor kan de gebruiker zeggen: pin(1)=hoog en zorgt de class voor de juiste byte.

5 Elektronica

In dit hoofdstuk staat de benodigde elektronica voor een juiste aansluiting. Aan de ene kant de I/O van en naar de computer en aan de andere kant de I/O van en naar het proces van de gebruiker. Hierbij moet tenminste een voltage van 230 wisselspanning mogelijk zijn, maar bijv. 5 volt moet ook kunnen.

6 Behuizing

Vervolgens is er nog een behuizing nodig voor de benodigde elektronica. Hierbij wordt ernaar gestreefd, om deze modulair op te bouwen, zodat op later tijdstip eventueel extra elektrische componenten gemakkelijk aan elkaar aan gesloten kunnen worden.

Referenties

Ref. 1	"*Interfacing the Standard Parallel Port*", Craig Peacock, URL: http://www.senet.com.au/~cpeacock, 19 Aug. 2001
Ref. 2	"*Interfacing the IBM PC Parallel Printer Port*", Zhahai Stewart, URL:http://www.rmii.com/~hisys/parport.html, 9 Jan. 1994

Bijlage A: Pin configuratie: D-Type 25 pins

Pin nr. (D-type 25)	Pin nr. (Centronics)	SPP Signaal	Direction In/Out	Register	Hardware Inverted
1	1	nStrobe	In/Out	Control	Ja
2	2	Data 0	Out	Data	
3	3	Data 1	Out	Data	
4	4	Data 2	Out	Data	
5	5	Data 3	Out	Data	
6	6	Data 4	Out	Data	
7	7	Data 5	Out	Data	
8	8	Data 6	Out	Data	
9	9	Data 7	Out	Data	
10	10	nAck	In	Status	
11	11	Busy	In	Status	Ja
12	12	Paper-Out/ Paper-end	In	Status	
13	13	Select	In	Status	
14	14	nAuto-Linefeed	In/Out	Control	Ja
15	32	nError/nFault	In	Status	
16	31	nItinialize	In/Out	Control	
17	36	nSelect-Printer / nSelect-In	In/Out	Control	Ja
18-25	19-30	Ground	Ground	-	
n: actief (hoog) als pin = laag					
Hardware Inverted: het signaal is omgekeerd door de hardware kaart					

Bijlage B: Software Registers

Volgens Ref. 2:

offset:	waarde:
base adres	data
base adres + 1	control
base adres + 2	status

I have labeled the Data Out register bits as D0 to D7, with D0 being the least significant bit and D7 the most signficant. The Control Out bits are labeled C0 to C3 (for the ones which go to pins) and C4 (for IRQ enable), and maybe C5 (for bidirection ports only, controls direction). The Status In bits are labeled S3 to S7 (corresponding to data and CPU bit positions). Often I will suffix C0 to C3 and S3 to S7 with a "+" or "-" as a reminder of whether or not that register bit is inverted as compared to the output or input pin it is associated with; "-" is for inverted, of course. All the Data Out bits are direct (not inverted); likewise data in for bidirectional printer ports, but I have not bothered to suffix a "+". Hexadecimal numbers are prededed by "0x", the C convention.

Port	R/W	IOAddr	Bits	Function
Data Out	W	Base+0	D0-D7	8 LS TTL outputs
Status In	R	Base+1	S3-S7	5 LS TTL inputs
Control Out	W	Base+2	C0-C3	4 TTL Open Collector outputs
"	"	"	C4	internal, IRQ enable
"	"	"	C5	internal, Tristate data [PS/2]
Data Feedback	R	Base+0	D0-D7	matches Data Out
Control Feedbk	R	Base+2	C0-C3	matches Control Out
"	"	"	C4	internal, IRQ enable readback

The Feedback registers are for diagnostic purposes (except in bidirectional ports, where Data Feedback is used for data input; and the IRQ enable C4).

5. Pin signals and register bits

<= in => out	DB25 pin	Cent pin	Name of Signal	Reg Bit	Function Notes
=>	1	1	-Strobe	C0-	Set Low pulse >0.5 us to send
=>	2	2	Data 0	D0	Set to least significant data
=>	3	3	Data 1	D1	...
=>	4	4	Data 2	D2	...
=>	5	5	Data 3	D3	...
=>	6	6	Data 4	D4	...
=>	7	7	Data 5	D5	...
=>	8	8	Data 6	D6	...
=>	9	9	Data 7	D7	Set to most significant data
<=	10	10	-Ack	S6+ IRQ	Low Pulse ~ 5 uS, after accept
<=	11	11	+Busy	S7-	High for Busy/Offline/Error
<=	12	12	+PaperEnd	S5+	High for out of paper
<=	13	13	+SelectIn	S4+	High for printer selected
=>	14	14	-AutoFd	C1-	Set Low to autofeed one line
<=	15	32	-Error	S3+	Low for Error/Offline/PaperEnd
=>	16	31	-Init	C2+	Set Low pulse > 50uS to init
=>	17	36	-Select	C3-	Set Low to select printer
==	18-25	19-30, 33,17,16	Ground		

Note: Some cables, ports, or connectors may not connect all grounds.
Centronics pins 19-30 and 33 are "twisted pair return" grounds, while
17 is "chassis ground" and 16 is "logic ground".

"<= In" and "=> Out" are defined from the viewpoint of the PC, not the
printer. The IRQ line (-Ack/S6+) is positive edge triggered, but only
enabled if C4 is 1.

Here's the same data grouped for ease of reference by Control Out and Status
In registers and pins. (Data Out is straightforward in previous table).

<= in => out	DB25 pin	Cent pin	Name of Signal	Reg Bit	Function Notes
=>	17	36	-Select	C3-	Set Low to select printer
=>	16	31	-Init	C2+	Set Low pulse > 50uS to init
=>	14	14	-AutoFd	C1-	Set Low to autofeed one line
=>	1	1	-Strobe	C0-	Set Low pulse > 0.5 us to send
<=	11	11	+Busy	S7-	High for Busy/Offline/Error
<=	10	10	-Ack	S6+ IRQ	Low Pulse ~ 5 uS, after accept
<=	12	12	+PaperEnd	S5+	High for out of paper
<=	13	13	+SelectIn	S4+	High for printer selected
<=	15	32	-Error	S3+	Low for Error/Offline/PaperEnd

==========

ps. De registers zijn alle drie: 8-bits waarden (Words).

Bijlage C: Hardware Schakelingen (Eng.)

Volgens Ref. 2.

C.1: Controlling Outputs

This can be easy; just use the Data Out TTL signals to control TTL level items. Unfortunately, they cannot source much current (providing positive voltage on the pin, relative to ground) - be careful of the 2.6 mA limit. Some LSI implementations might allow more or less than this (likely less).

```
       Dn Out ------+
                    |+
  Sourcing          Load (up to 2.6 mA @ 2.4 v)
                    |-
       Ground ------+
```

If you have an external +5 volt supply, you have two options: use the Data Out pins to sink up to 24 mA from your +5 volt supply, or use buffer chips to control (source or sink) more current (or higher voltages). I have used an exteranl 5 Volt supply (regulated wall wart) plus optocoupled solid state relays as the "load", to control AC voltages (keep the high AC voltages away from any of this logic level stuff, obviously).

```
                    +------------------------------ (+5 v)
                    |+
   Sinking          Load (up to 24 mA @ 4.2v)
                    |-                               Power Supply
       Dn Out ------+

       Ground ------------------------------------ ( Gnd)
```

Use limiting resistors if you need to limit the current.

If the load were an LED (or optocoupler) through which you wished to put 20 mA, do the calculations. The Dn Output will probably be around 0.7 volts, so you have about 4.3 volts of drop; the LED will drop about 1.9 v (check specs!), leaving 2.4 V to be dropped by a resister at 20 mA: 120 ohms. Test and measure, adjust to fit.

You can also use the Control Out pins. They can't source much of anything (about 1 mA through the 4.7K resistors to +5), and can only sink about 7mA. (The LS TTL gate actually sinks 8 mA, but one is taken up by the 4.7 K resistor to +5). Again, check on clones with different electical specs. This can control TTL inputs fine, and might be able to run an optocoupler or solid state relay in sink mode (depends on the device).

In one application, I used two 74ACT374 latches, which can source 48 mA or sink 64 mA. I connected the 8 inputs of each to the Data Out, and the latch clocks to two Cn outputs. In software, I put out 8 bits of data on the Data Out port, pulsed a Cn bit to latch it into one 74ACT374, put the next 8 bits out on Data Out, and used the other Cn bit to latch it into the other 74ACT374 - voila, 16 bits of 64mA output control. Of course, this took a separate +5 V power supply ($5 surplus regulated wall wart).

If you can still find an old TTL parallel port (especially with sockets), you can substitute the 74ACT374 chip for the original 74LS374 and get better drive capability. The back of magazine suppliers were selling *fully socketed* TTL based parallel ports for about $15 a few years back; by cutting a trace and soldering a jumper you could make them bidirectional; by plugging in a $1 chip you could make them source/sink 48/64 mA. You might still find such TTL parallel port cards in old PCs. Typically, one designed for the original PC will still work fine on the ISA bus of your 486DX2-66, so don't worry about that.

C.2: Sensing switches

There are several ways to sense external switches via a parallel printer port. If you are dealing with naked switches, the simplest is actually to connect up to 4 switches between the control outputs (pins 1,14,16, and 17) and ground, program the control outputs high (counting inversions), and use the control feedback register to read switch state (counting inversions). This is because the pull-up resisters are already implemented for the open collector Control Out pins.

```
Cn Out ------+          (Program output high; read feedback)
             |
           switch
             |
Ground ------+
```

Another simple method would be to use up to 5 pullup resistors (4.7K) from the Status inputs to +5 volts, and use switches to pull these down to ground. One disadvantage of this is that it uses an external +5 supply. TTL inputs tend somewhat to float high, so you *might* get by without the pullup resistors, but you are pushing it.

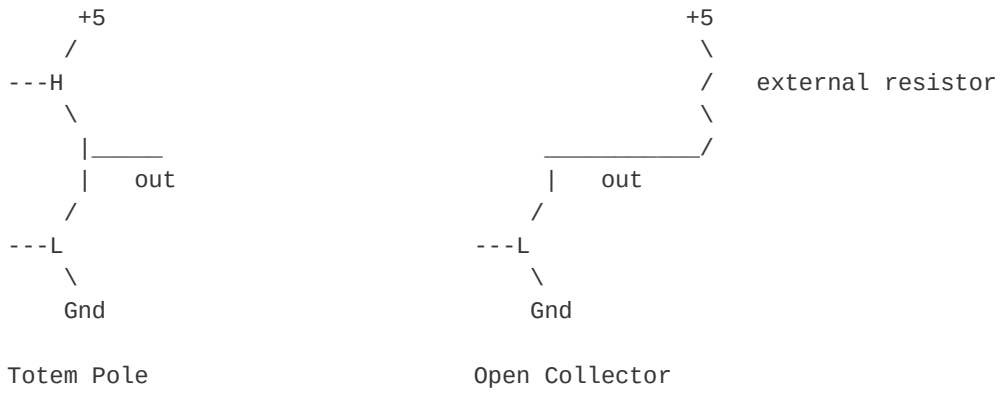
```
               4.7K
Sn In -------+-^^^^^--- (+5 v)      (Read status register)
             |
           switch
             |
Ground ------+---------- (gnd)
```

You could use this same scheme for 8 inputs on the data lines of a bidirectional printer port - but you have to be sure that the outputs are tristated beforehand, or your switches may damage the 74LS374 (or equiv). This latter may be a problem after booting or rebooting. One approach is to use current limiting resistors in series "just in case". You should ideally still have pullup resistors too. (The 1K value is chosen for LS TTL, to keep the 2.4 V output at no more than 2.4 mA sourcing).

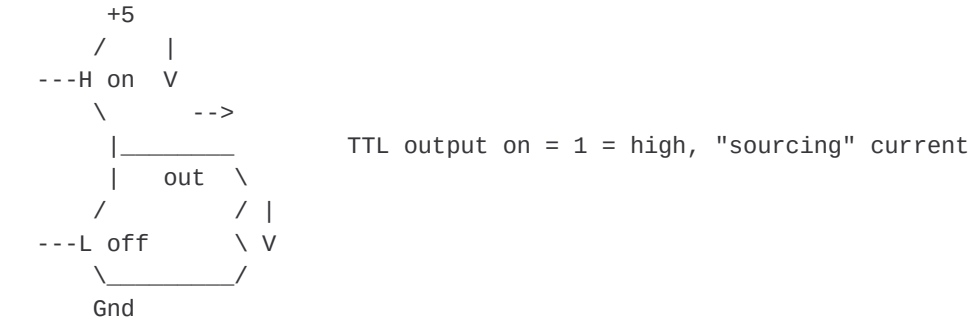
```
                1K           4.7K
Dn Out --------^^^^^-----+----^^^^^------ (+5v)
                         |
                       Switch
                         |
Ground -------------------+--------------- (Gnd)
```

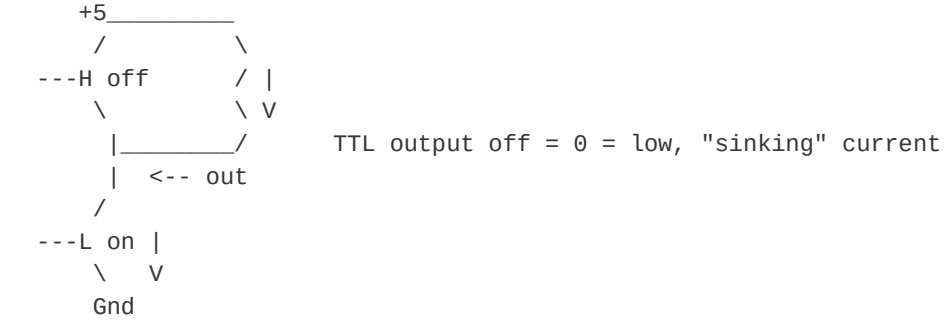
C.3: Tutorial on TTL outputs

In response to requests, here is a brief tutorial on chip outputs. As a preface, it is conventional to discuss current as flowing from positive to negative, even though we know full well that electrons actually move in the opposite directions; it's just a convention, not ignorance.



Regular TTL outputs basically consist of a two "stacked" transistor in series between +5 volts and ground, with the output coming from the connection between them. This is called a "totem pole output". At any given time one of these transistors is conducting and the other is not. To pull the output "high", the transistor from +5 to the output conducts (H), which "sources" positive current from the output to ground (that is, an external device between the output and ground will get power). To pull the output low, only the lower transistor (L) conducts, "sinking" current to ground; an external device between +5 volts and the output can be energized. Current flows into the chip for low, out of it for high.





The 0 / pull low current capacity (sinking) is larger than the 1 / pull high capacity (sourcing). If you want to drive something like an LED, or a solid state relay, you can get more current from the TTL outputs by connecting it between +5 and the gate output (second picture) - *if* it's electrically isolated from ground. You still have to check the current and voltage ratings, too.

One key here is that the chip is always trying to pull either high or low, and the currently conducting transistor has voltage and current limits, beyond which it can be damaged. It is not good design to connect two such outputs which may have different states - one pulling high and one low - because this will exceed the current specs. However, if you do, the one pulling low will "win", since TTL can sink (pull low) more strongly than it can source (pull high). And there is some slack in the specs, so this does not always immediately damage the chip; we'll get back to this. These are the type of outputs on the DATA lines on the original IBM parallel port. (Well, not exactly, but I'll get back to that too).

Another type of output is the "open collector". In this case, there is a transistor from the output pin to ground, but none to +5 volts. The two states are 0, "conducting to ground" (pulling low), and 1, "not conducting" (floating, not pulled either way). Externally, you can connect a resistor of proper value between the line and +5 volts to pull the line high when the chip output is floating. The value is chosen such that when the output is conducting to ground, it overpowers the external resistor and the line goes low. The advantage of this scheme is that you can connect multiple open collector outputs together (or even slip in one totem pole). Every output that is floating is ignored (and the resistor will pull the line high if and only if all outputs are floating); multiple outputs pulling low cause no conflict. This is the type of output used for CONTROL lines in the original IBM parallel port.

Since each parallel port control output line has a corresponding feedback bit that can be read (mainly for diagnostics, to see if the line really goes high or low), it is possible to program these CONTROL outputs "high" (really floating), and then allow external signals control the high or low state of the line. An external open collector output, or a totem pole one, is capable of pulling the line high (reinforcing the pullup resistor) or low (overpowring it) without exceeding current specs. In this way you can use the control lines as inputs. If there is no external signal, it will read high. However, you must program the CONTROL outputs high (taking into account any logical inversions between the register bits and the electrical outputs); if the open collector output is low (conducting to ground), the external signal won't be able to pull it high (or at least not without exceeding the specs).

The same thing can be done with totem pole outputs - program them high, and let external logic pull them high (no problem, both pulling up) or low (overpowering the attempt to pull high), but in this case you are overloading the H transistor, rather than a pullup resistor, and exceeding spec, with possible unreliability or damage. However, this has been successfully done with the data outputs of a conventional parallel port, and some people claim not to have seen any damage yet. Don't blame me if you do it and something dies.

A third type of gate is a totem pole in which both high and low transistors can be non-conducting at once, creating a third, floating state (this is often called a tri-state output). When it is in this third state, the line is not being pulled high nor low by this device, and thus can be safely controlled by some other device. No pullup resistor is used. The 74LS374 chip used for the standard parallel port DATA outputs actually has tristate ability, but as described elsewhere the third state is not used, and it is always trying to pull high or low - thus my initial description of the DATA lines as totem pole drivers.

The way true bidirectional parallel ports work is to allow the software to selectively put the DATA outputs into the third, undriven state. Then you can use the data feedback register to read whatever highs or lows are put onto the data lines externally.

Non-TTL logic, like CMOS, has the equivalents of these, but with a different type of transistor, and different voltage and current values. If your parallel port uses a single chip or otherwise differs from the original IBM parallel port, the above electrical description cannot be guaranteed, but it is probably pretty analogous. Let me know if you have specific knowledge about the electical specs of your single chip parallel port, for future versions of this document.

Note again that high and low in this description are electrical levels, as would me measured by a voltmeter on the pins; high is associated with TTL logic 1, but whether a given CONTROL output line is high when the corresponding register bit is 1 or 0 varies with the line, as is described elsewhere in this document. The DATA lines are straight through, with no inversions, so a 1 bit produces a high output.

Bijlage D: Drivers

Driver naam:	Besturingssysteem:	URL:
inpout16.dll	DOS	http://www.lvr.com
vbasm.dll	DOS	http://www.softcircuits.com
inpout32.dll	Win95	http://www.lvr.com
win95io.dll	Win95	http://www.softcircuits.com
port.dll	Win95	http://members.aol.com/trun40
pport.dll	Win95	http://www.pgacon.com/visualbasic.htm
dlportio.dll	Win95	http://diskdude.cjb.net/
io.dll	Win95	?
dlportio.sys	WinNT	http://diskdude.cjb.net/
userport.sys **userport.exe**	WinNT	http://www.ddj.com
tinyport.sys	WinNT	?