Parallel Port on a PC

C Programming for Engineers

Nick Urbanik nicku@nicku.org

2005 October

Outline

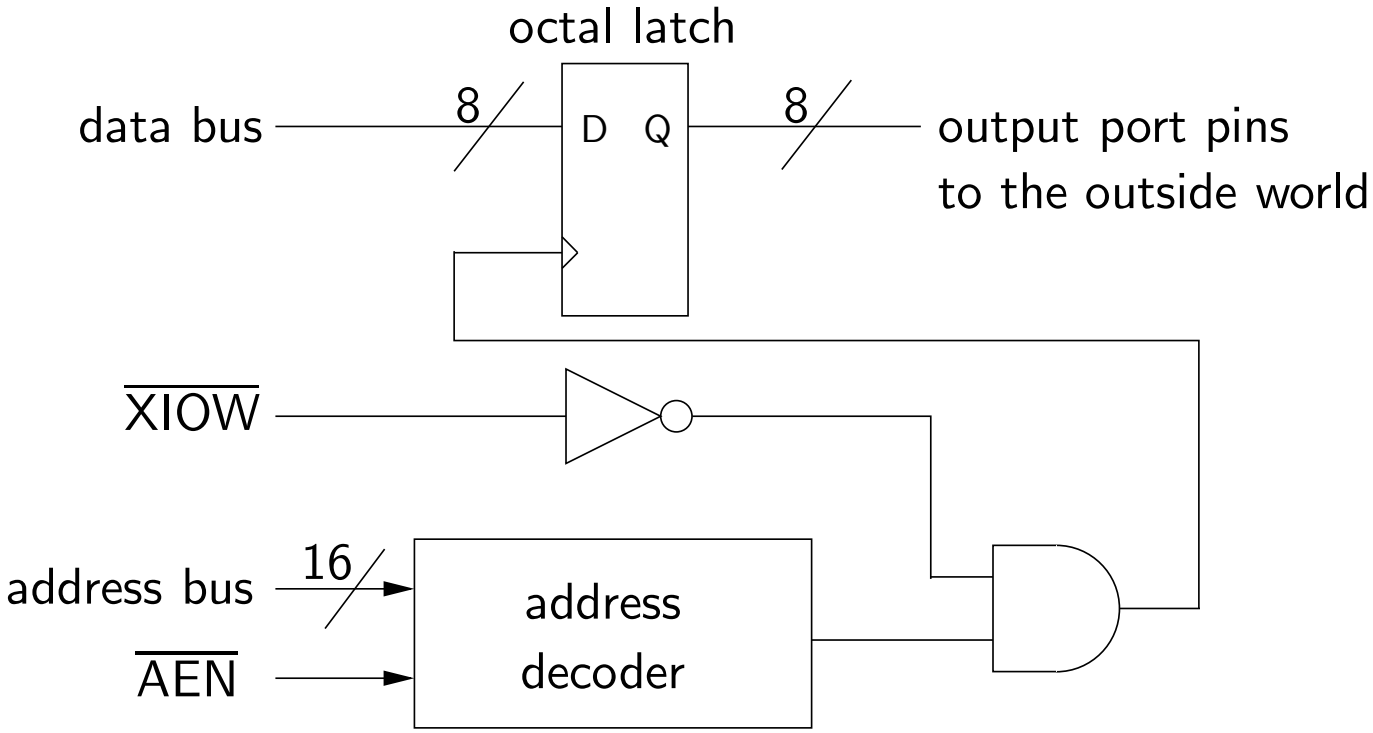
I/O Ports on a PC

- There are $2^{16} = 65536$ I/O addesses
- each of these is called an *I/O port*
- They are accessed with the `in` and `out` Intel assembly language instructions
- The I/O ports are separate from ordinary memory addresses
  - We say, "I/O ports have a separate *address space* from memory addresses".
- I/O ports usually connect to *registers* on integrated circuits on the motherboard or on cards plugged into the motherboard

Hardware of I/O ports

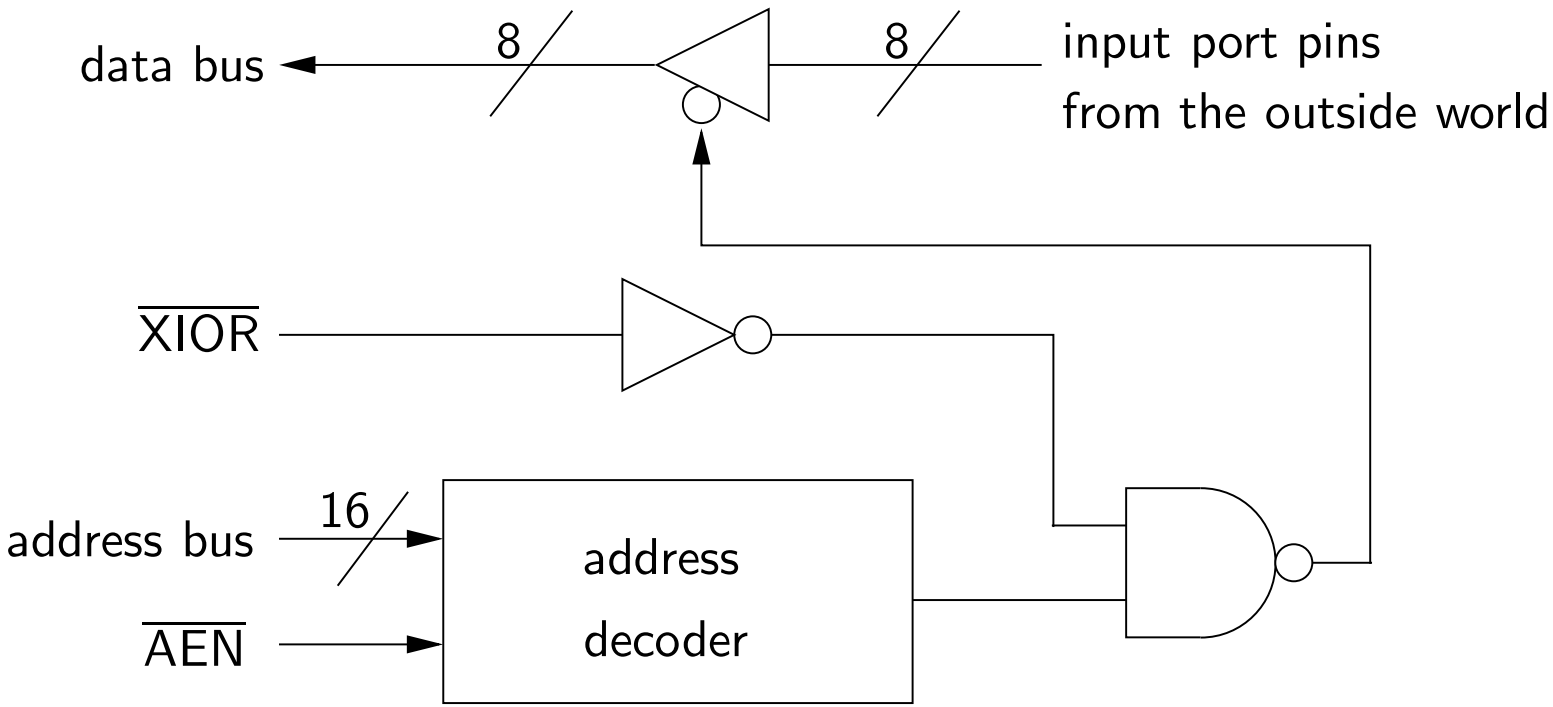
- We cannot connect hardware directly to the data bus on the CPU
- CPU may not source or sink enough current
- but the main reason is that the data bus is changing all the time
  - Carries instructions and other data, continuously passing back and forth
- For output: need a latch (set of flip-flops) to catch the data when the output instruction is executed, and hold the data steady
- For input: a tristate buffer (e.g., 571) that connects input pin to data bus at the time the input instruction is executed

Hardware of Output Port



- the latch “catches” the data and holds it when the output instruction is executed to the correct address
- The $\overline{\text{XIOW}}$ control line from the CPU’s control bus is activated by the output instruction
- This keeps the I/O addresses separate from memory addresses even when they have the same address number

Hardware of Input Port



- The tristate buffer connects the input pin to the data bus only when the input instruction is executed with the appropriate address
- The $\overline{\text{XIOR}}$ control line from the CPU's control bus is activated by the input instruction

Outline

Five modes of Operation

- Newer parallel ports are standardised under IEEE standard 1284
  - released in 1994
- The standard defines *five modes of operation*:

Compatibility mode — sometimes called "Centronics Mode"
  - can send data out only
  - upper limit: 50 kBps to 150 kBps, depending on hardware

nibble mode Can input 4 bits at a time

byte mode can input a byte at a time using parallel port's bi-directional feature

EPP mode (Enhanced Parallel Port) — Uses additional hardware to perform *handshaking*

ECP Mode (Extended Capabilities Port) Uses DMA and FIFO buffers to move data without using I/O instructions

Handshaking with a printer in Compatibility Mode

To output a byte from the parallel port to the printer in *compatibity mode*:

1. Write the byte to the Data Port
2. Check if the BUSY line is active
   - If the printer is busy, the port will not accept any data, so any data sent to the data port will be lost
3. Take the $\overline{\text{STROBE}}$ line low
   - Tells printer that valid data is waiting on the data pins 2–9
4. Put $\overline{\text{STROBE}}$ high again after about 5 microseconds.

Outline

The Three Printer Port Base Addresses

Address	Notes
0x3bc – 0x3bf	Used for parallel ports that were incorporated into video cards, and now an option for an additional port. Does not support ECP
0x378 – 0x37f	Usual address for LPT1 (first parallel port)
0x278 – 0x27f	Usual address for LPT2 (second parallel port)

There are three I/O Ports

- Data port
  - At printer port base address
  - all eight bits normally output
  - Can input data if port has bi-directional hardware
- Status port
  - at base address + 1
  - read only
- Control Port
  - at base address + 2
  - read and write, though was originally intented as a write only port.

Outline

The Data Port

- At: base address of printer port
- Write only, unless the port hardware is bi-directional

pin number	Bit number	signal name
2	bit 0	D_0
3	bit 1	D_1
4	bit 2	D_2
5	bit 3	D_3
6	bit 4	D_4
7	bit 5	D_5
8	bit 6	D_6
9	bit 7	D_7

Outline

The Status Port

- At: Base address + 1
- Read only

pin number	Bit number	signal name
	bit 0	reserved
	bit 1	reserved
	bit 2	$\overline{\text{IRQ}}$
15	bit 3	$\overline{\text{ERROR}}$
13	bit 4	SLCT
12	bit 5	PE (Paper End)
10	bit 6	$\overline{\text{ACK}}$
11	bit 7	BUSY

Outline

The Control Port

- At: base address + 2
- Read and Write

pin number	Bit number	signal name
1	bit 0	$\overline{\text{STROBE}}$
14	bit 1	$\overline{\text{AUTOFEED}}$ (Auto Linefeed)
16	bit 2	INIT PRN
17	bit 3	$\overline{\text{SELECT}}$
	bit 4	Enable IRQ via Ack
	bit 5	Enable Bi-Directional Port
	bit 6	Unused
	bit 7	Unused

Using the Printer Port for I/O

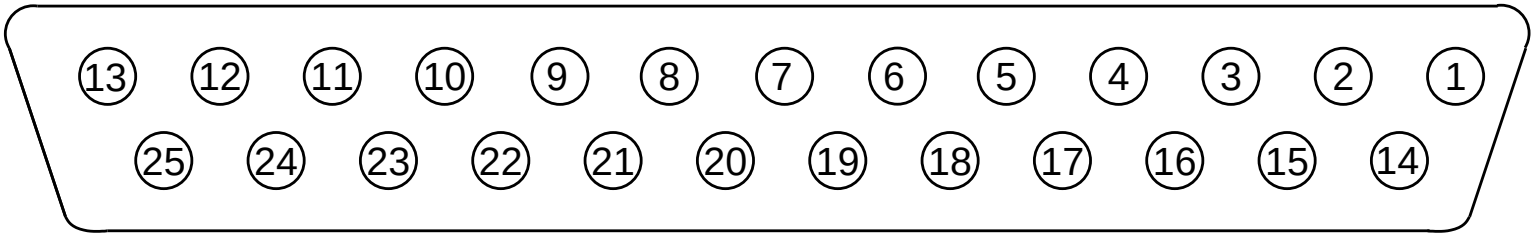
- Here, we use the printer port in compatibility mode
- In this mode, the three ports are not available as general purpose *8-bit* input/output ports
  - They are set up to talk to a printer
  - But you can still use these ports for many purposes

Signals and pin numbers for general purpose I/O

Port		*Signal Name*	*DB25 pin number*	*Comments*
Data base		D_0	2	All outputs latched
		D_1	3	
		D_2	4	
		D_3	5	
		D_4	6	
		D_5	7	
		D_6	8	
		D_7	9	
Status base + 1	bit 3	ERROR	15	input
	bit 4	SLCT	13	input
	bit 5	PE	12	input
	bit 6	ACK	10	input
	bit 7	BUSY	11	inverted input
Control base + 2	bit 0	STROBE	1	inverted output
	bit 1	AUTOFEED	14	inverted output
	bit 2	INIT PRN	16	output
	bit 3	SELECT	17	inverted output
		GND	18–25	

Pin numbers on DB25 Connector

- This views the *female* connector
- i.e., on the back of the computer



View of female DB25 connector

Pin Numbers on Parallel Port DB25

Pin No (D-Type 25)	Pin No (Centronics)	SPP Signal	Direction (In or Out)	Register	Inv?
1	1	$\overline{\text{STROBE}}$	In/Out	Control	Yes
2	2	D_0	Out	Data	
3	3	D_1	Out	Data	
4	4	D_2	Out	Data	
5	5	D_3	Out	Data	
6	6	D_4	Out	Data	
7	7	D_5	Out	Data	
8	8	D_6	Out	Data	
9	9	D_7	Out	Data	
10	10	$\overline{\text{ACK}}$	In	Status	
11	11	BUSY	In	Status	Yes
12	12	PE (PaperEnd)	In	Status	
13	13	SELECT	In	Status	
14	14	$\overline{\text{AUTOFEED}}$ ($\overline{\text{Auto-Linefeed}}$)	In/Out	Control	Yes
15	32	$\overline{\text{ERROR}}$ / $\overline{\text{Fault}}$	In	Status	
16	31	$\overline{\text{INIT PRN}}$	In/Out	Control	
17	36	$\overline{\text{SELECT}}$ $\overline{\text{Select-In}}$	In/Out	Control	Yes
18 – 25	19 – 30	Ground	GND		

Do not run your programs as root/Administrator

- Normally, to access I/O ports requires administrator priveleges
- . . . but it is a bad idea to do everything as the root or administrative user
  - A small mistake can stop the system from functioning correctly
  - In Windows XP/2000/NT, additionally, special unsupported software is required.
  - Linux provides a *system call* `ioperm()` that allows the root user to grant normal user access to particular ports
  - The ports must be at port address 0x3ff or below

Performing I/O in Windows XP, 2000, NT

- Port I/O on Windows XP, Windows 2000, Windows NT is a complex, barely supported mess.
- Use Linux if you want something simple, standardised and supported:
  `http://linuxgazette.net/112/radcliffe.html`
- Several people have built device drivers to work around the limitations of Windows:
  - `inpout32.dll`:
    `http://www.logix4u.net/inpout32.htm`
  - PortTalk:
    `http://www.beyondlogic.org/porttalk/porttalk.htm`
  - `io.dll`:
    `http://www.geekhideout.com/iodll.shtml`
  - `giveio.sys`:
    `http://www.physik.rwth-aachen.de/group/IIIphys/CMS/tracker/en/silicon/arcs_nt.html`
  - directio:
    `http://www.direct-io.com/`
- None of these are Open Source, but `inpout32.dll` seems to be best supported and have the most open license, so we will use that.

Outline

Installing Andy Eager's wrapper for logix4u inpout32.dll

Note: this is for use with Microsoft Windows. The procedure with Linux is different, simpler and faster: see the references.

- Download Andy's handy package from `http://www.linuxivr.com/c/week1/ioports.zip`
- Unzip this into a temporary directory
- execute `install.bat` from a command prompt in that directory as the Administrator

Outline

Using Andy Eager's wrapper

- See the program `ledscan.c` in `http://www.linuxivr.com/c/week1/ioports.zip` — use this as a model to see how to perform I/O
- Compile your program with the command: `g++ -Wall -lioports -o` ⟨*program*⟩ ⟨*program*⟩`.cpp`

Outline

Using inpout32.dll without Andy's wrapper

- This could (potentially) give better performance if you initialise the library once at the beginning and free the library once after all I/O is finished
  - However, Andy says the difference in speed is not detectable

  See the test program `http://www.hytherion.com/beattidp/comput/pport/test2.c`, and also the source to Andy's wrapper at `http://www.linuxivr.com/c/week1/install-io.html`, and use them as a model for your program.

References — Web I


logix4u.net.
Inpout32.dll for WIN NT/2000/XP — logix4u.
http://www.logix4u.net/inpout32.htm


Andrew Eager.
Installing the logix4u IO interface.
http://linuxivr.com/c/week1/install-io.html


logix4u.
Parallel port Interfacing Tutorial.
http://www.logix4u.net/parallelport1.htm


Joe D. Reeder.
Controlling The Real World With Computers
http://learn-c.com/


Riku Saikkonen.
Linux I/O port programming mini-HOWTO
http://www.tldp.org/HOWTO/IO-Port-Programming.html

References — Web II


P. J. Radcliffe.
Linux: A Clear Winner for Hardware I/O.
Linux Gazette, Issue 112, March 2005.
http://linuxgazette.net/112/radcliffe.html


David Chong and Philip Chong
Linux Analog to Digital Converter.
Linux Gazette, Issue 118, September 2005.
http://linuxgazette.net/118/chong.html


Craig Peacock
Interfacing the Standard Parallel Port.
http://www.beyondlogic.org/spp/parallel.htm


Jan Axelson.
The PC's Parallel Port.
http://www.lvr.com/parport.htm

References — Books


Steve Oualline.
Practical C Programming.
O'Reilly, 1993.


Paul Davies.
The Indispensable Guide to C with Engineering Applications
Addison-Wesley, 1995.


Tom Adamson and James L. Antonakos and Kenneth C. Mansfield Jr.
Structured C for Engineering and Technology, Third Edition.
Prentice Hall, 1998.


Brian W. Kernighan and Dennis M. Ritchie.
The C Programming Language.
Prentice Hall, 1988.

License covering this document