Printer port data acquisition system: A/D input

Eric Auer <eric -at- coli.uni-sb.de>
(this is an excerpt from a term project of mine)

17th July 2004

1 System outline

This text presents a simple, effective and inexpensive 12 bit A/D converter device. This device connects to the parallel printer port and exploits the fact that starting with 386 based PCs (maybe even earlier) printer ports became bidirectional. This is not to be confused with ECP or EPP printer port functions: ECP / EPP allows you to transfer data to and from the printer in a very fast and heavily hardware assisted way. My device is already happy if the data bus between PC and printer works in reverse direction.

For that, you tell the port hardware to stop sending a data byte. Then you can read back data on the bus which is sent to the PC. This input byte is of course not wide enough for the 12 bit A/D conversion result (0 to 4095 range) which the PC should be able to read from the device. Therefore the device uses two monolithic serial A/D converters. The other 6 bits of the byte after subtracting two bits for the serial X and Y data are free for other signals.

(...) first set the data port to input (towards PC) direction before it will trigger connection of the device. Then it will read the digital data from the port. The clock for the serial data transmission of the X and Y coordinates is sent to the device through the STROBE line. The INIT line also serves as a signal to start A/D conversion. Both converters will send their results at the same time, and both conversions will happen at the same time. This allows a simpler driver and speeds up data readout.

I recommend to use the LTC1286 12 bit monolithic A/D converter for this program: The clock for the serial data transmission can be up to 200kHz, so reading the coordinate data only takes a fraction of a millisecond, which is more than enough speed (...) The selected converter chips are a cost-effective solution (less than 15 Euros each) and allow enough speed and accuracy for our purposes.

2 Portability

For better portability, and to make full use of 32 bit CPUs, the software is written for the DJGPP compiler. DJGPP is the DOS version of the well-known gcc/g++ compilers which are popular among Linux programmers. It is open source and freely available. The C library of the DJGPP platform is modelled after the GNU C libary of Linux, but the software itself is not meant to work in Linux!

Small parts of the software are written in inline assembly language: A tiny delay which must not be optimized away by the C compiler in the A/D interface readout and the code which reads the Time Stamp Counter. This inline assembly is processed with the GNU AS assembler which is part of the DJGPP compiler suite. Note that AS uses AT&T syntax which has some important differences from classic Intel Assembly language syntax (as used in MASM, TASM and NASM assemblers, for example).

If you ever want to create a Linux version of the software, you will probably need root privileges to run it. Some important points: This is real-time software, so for a Linux version, you have to make sure that no other tasks can take CPU time at the wrong moments. Only root is allowed to reserve the CPU for a single task. We also access hardware directly. You can rewrite most of this, but you will still need privileges to access the corresponding devices. Used devices are, among others, the time stamp counter and the printer port.

3 More things about the device

I decided to use 2 monolithic LTC1286 converters with 12 bit resolution (4096 steps) and up to 12500 samples per second. Speed depends on the used serial data transmission clock speed. Please check the LTC1286 data sheet for details about the timing and communication protocol. An alternative would have been a single LTC1298 converter with builtin multiplexer: You can read up to 11500 samples per second with that chip and select for each sample which of the two input signals has to be sampled. As the LTC1286 are only 15 Euros each, I decided to use two of them. This allows to read both the X and the Y voltage at the same moment and reduces the data transfer time because both values are read in parallel.

Using the printer port in the input direction (possible since PS/2) allows you to read eight digital signals in parallel. The output data drivers in the PC are usually open collector drivers to pull low and 4700 Ohm resistors to pull high – the resistors stay active while the port is in printer-to-PC mode. (...) The A/D chips have builtin logics to disconnect their output signals from the bus while the printer port is in output mode. We use the $\overline{\text{AutoFeed}}$ signal is to tell the device when the printer port is in input direction.

Although the A/D converters consume very little power, you have to get a 5 Volt power supply for the device. You can either connect to a floppy power connector in the PC and have a cable to the device from there. The 5 Volt supply of the joystick port is often too weak. Other possibilities are the serial port control lines (can be programmed to either positive or negative 12 Volts and can supply 20 mA each, far more than needed for the device) and the keyboard connector. Using 12 Volts and regulating down to 5 Volts allows better stabilized power supply to the A/D device. Make sure that the device is not connected to the printer port while not being connected to the power supply – it could damage the chips to have data lines connected but no power present.

The X and Y data stream from the A/D chips is read through the data port bits 5 and 4 respectively (masks 32 and 16). Other bits can be used for button inputs (you can also connect 4 digital signals to the joystick joystick port). For non-bidirectional printer ports, you can use status lines to transmit the serial data stream.

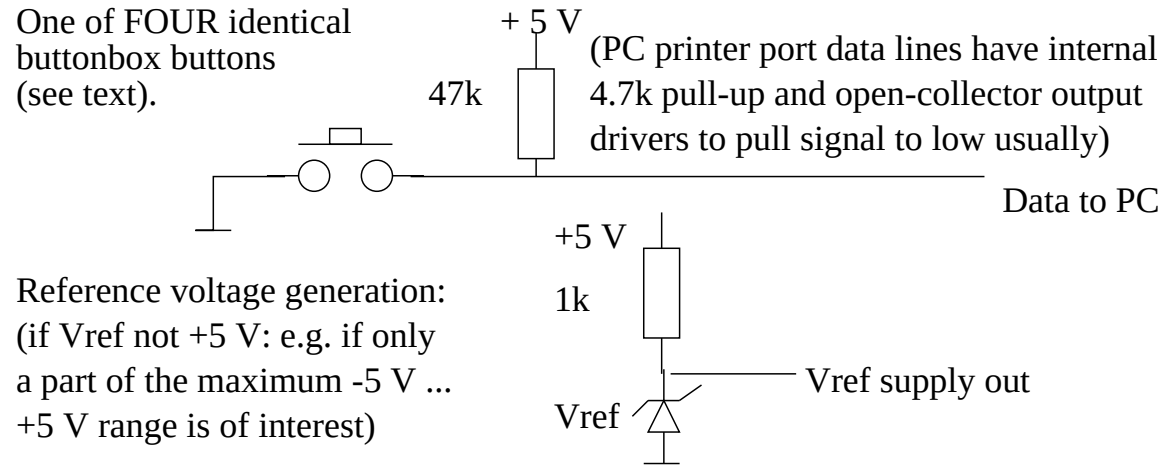
Pin assignment information: Drive power plugs use black for ground, yellow for 12 Volts and red for 5 Volts. The joystick port pins are: Pin 2, 7, 10 and 14 for buttons 1...4 (normally open buttons, connected to ground, which is at pins 4, 5 and 12). Pin 3, 6, 11 and 13 for X1, Y1, X2 and Y2 respectively. Pin 1 and 9 for 3$\dot{3}$ or 5 Volt supply to which the X/Y potentiometers are connected. Pin 12 and 15 are sometimes used as midi output / input.

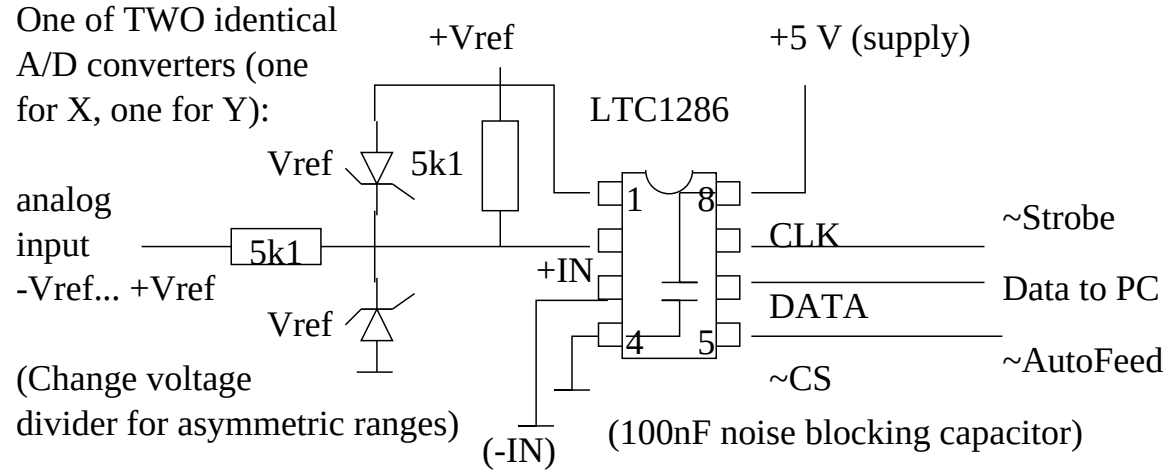
Printer port pins – at the sub D 25 pin connector – are: Data bit 0...7 are pins 2...9. Pin 1 is $\overline{\text{Strobe}}$ and pin 10...17 are, respectively, $\overline{\text{Ack}}$, Busy, PaperEnd, Select, $\overline{\text{AutoFeed}}$, $\overline{\text{Error}}$, $\overline{\text{Init}}$ and $\overline{\text{SelectIn}}$. Pins 18...25 are ground.

4 References

GNU General Public License: http://www.gnu.org/licenses/gpl.html
NASM Assembler: http://nasm.sourceforge.net/
DJGPP Compiler by DJ Delorie: http://www.delorie.com/djgpp/
FreeDOS, a DOS operating system: http://www.freedos.org/
Printer port pinout: http://www.hamhud.net/ka9mva/regpins.html
Joystick port: http://www.epanorama.net/documents/joystick/pc_joystick.html
Ralf Brown's Files (RBIL,...):
http://www-2.cs.cmu.edu/afs/cs/user/ralf/pub/WWW/files.html

Circuity for printer port data acquisition module





The Zener diodes are optional protective clamping.
Use sockets for the LTC1286s.

Voltage regulation circuit, to get 5 Volts from a 12 Volt supply: