Abstract

This report describes the hardware and firmware for a differential-pressure indicator for the low-speed wind tunnel. It is essentially a two-channel high-resolution voltmeter attached to a pair of Sensirion pressure sensors but, being built around a microcontroller running FlashForth, it is programmable.
1 Hardware

Inside the box, the prototype system is built on a single strip board, with a Microchip PIC18F2520 microcontroller [1] talking to a 2-channel high-resolution analog-to-digital converter (MCP3422 [2]) and a Modtronix serial-enabled LCD[3]. Photographs of the components are shown in Fig. 1 and schematic layouts of the circuit are shown on the following pages.

Figure 1: The photograph on the left shows the main circuit board with microcontroller (centre), MAX232 serial-port level converter (lower right) and the MCP3244 analog-to-digital converter (lower left). The potentiometer in view was used to simulate a pressure sensor during firmware development. The photograph on the right shows the indicator boxed, sitting on the side of the low-speed wind tunnel, with two Sensirion SDP1108-R pressure sensors [4] mounted on the front panel. The upper sensor is attached to the pair of static-pressure rings on the contraction leading into the test-section of the wind tunnel and the lower sensor is connected to a Pitot-static probe.
Although the microcontroller is really quite capable and includes a 10-bit analog-to-digital converter, we really don’t use a lot of its capability. Instead, we use its I²C bus to glue together the analog-to-digital converter chip (which we’ve been wanting to try for some time) and the LCD display. As seen in the photograph, there are very few pins in use on the microcontroller and this is confirmed on sheet 1 of the schematics, where more than half of the MCU pins are labelled as not connected. The principal work of the microcontroller is to get the digital values from the MCP3422, do some arithmetic to convert the values from counts of voltage into units of Pascals, and display them on the LCD.

The outputs of the Sensirion SDP1108-R pressure transducers can produce up to 4 V (for 500 Pa) and need to see a fairly high input impedance (100kΩ is recommended) at the data converter. The MCP3422 has an input range of ±2.048 V at the channel input pins. To satisfy both of these requirements, a pair of 56 kΩ resistors is used as a voltage divider for each channel (Schematic sheet 3). The 1 µF capacitor provides some low-pass filtering for the incoming signal but also provides a low impedance for the high-frequency (but small) currents that the switched-capacitor input stage of the MCP3422 requires during the conversion process.

Note that the simple two-resistor voltage divider at the MCP3422 input is serviceable but not ideal. Using signed 16-bit values from the MCP3422 means that the least-significant bit corresponds to $2 \times 62.5$ microvolts at the input to the voltage divider. Even with the input to the divider tied to ground, a leakage current of 25 nA through the protection diodes of the MCP3422 results in a 16-bit converter count of about 6 rather than (the ideal value of) zero, however, a count of 6 corresponds to about 0.7 millivolts and is not a problem in this application.

Communication with the microcontroller is provided by a minimal (3-wire) RS232 serial port. This port was used for the development of the Forth firmware and is available for data logging. A suitable computer, with a serial-communications terminal program, is needed to make use of this facility.

The current-limited supply (black box in Fig. 1) shows that, with the back-light on, the LCD draws about 50 mA. In contrast, the microcontroller and data converter require only a few milliamps. The photograph shows the LCD backlight on, and this is fine when operating from a DC plugpack. If you wish to run the system from a battery, the firmware should be changed to turn the backlight off.
Low-speed wind-tunnel — power and comms

FILE: REVISION: DRAWN BY: PAGE OF

Low-speed wind-tunnel — power and comms

FILE: REVISION: DRAWN BY: PAGE OF

Low-speed wind-tunnel — power and comms

FILE: REVISION: DRAWN BY: PAGE OF
Low-speed wind-tunnel −− 18-bit data converter

MCP3422

U301

VDD

SDA

SCL

VSS

CH1+

CH1−

CH2+

CH2−

VDD 5.0V

GND

Sensor signal 0.25–4.0V

CONN_CH1

CONN_CH2

VDD 5.0V

GND

Sensor signal 0.25–4.0V

100n

10u

56k

56k

56k

1u

5V

100uH

10u

VSS

Sensor signal 0.25–4.0V

100n

VDD 5.0V

GND

Sensor signal 0.25–4.0V

56k

1u

VSS

10u

VSS

56k

56k

56k

1u

VSS

VSS
\section*{2 Firmware}

The firmware running within the microcontroller is written in Forth, with the FlashForth interpreter \cite{5} being previously programmed into the MCU. During development, interaction with the microcontroller was through the RS232 serial port and the application code was built interactively, in the four stages.

\textbf{lswt.txt}

The first stage was to talk to the MC3422, via the I2C bus, and get the converted values as integer counts. These are reported via the RS232 serial port.

\begin{verbatim}
\% lswt.txt
Low-speed wind-tunnel board using a PIC18F2520 and FlashForth
with mcp3422 and divide-by two analog input,
ready for Sensirion pressure sensors.
PJ, 09-Nov-2013

-mark-
lswt

: mcp3422-go-1 ( -- )
  \$d0 is default mcp4322 address for writing
  16-bit one-shot conversion of ch 1
  \$d0 i2cws %10001000 i2c ! spen
  ;

: mcp3422-go-2 ( -- )
  \$d0 i2cws %10101000 i2c ! spen
  ;

: mcp3422@2byte ( -- n f )
  Read the 16-bit result as 2 bytes
  \$d1 i2cws i2c@ak $8 lshift \ MSB
  i2c@ak or \ LSB into same cell
  i2c@nak $80 and 0= \ leave true if result is latest
  ;

: report-counts ( n f -- )
  if ." new " else ." old " then
  ." counts "
  ;
\end{verbatim}
The second stage was to talk to the Modtronix LCD, via the same I2C serial bus, and display text.

```plaintext
: run-with-counts ( -- )
   decimal
   i2cinit
begin
   mcp3422-go-1 #100 ms mcp3422@2byte
   cr ." chan 1 " report-counts
   mcp3422-go-2 #100 ms mcp3422@2byte
   cr ." chan 2 " report-counts
   #800 ms
   key? until
   hex
;
```

**lcd2s.txt**

The second stage was to talk to the Modtronix LCD, via the same I2C serial bus, and display text.

```plaintext
\ lcd2s.txt
\ Words for the Modtronix LCD2S display
\ Peter J. 26-Oct-2013

Assume that i2c_base has been loaded.

-lcd2s
marker -lcd2s

: lcdc! ( c -- ) $50 i2cws i2c! spen ;
: remember ( -- ) $8d lcdc! ;
: blight-off ( -- ) $20 lcdc! ;
: blight-on ( -- ) $28 lcdc! ;
: lcd-clear ( -- ) $8c lcdc! ;
: lcd-curs-pos ( row col -- )
  swap \ now have S: col row
  $50 i2cws $8a i2c! i2c! i2c! spen
;
: lcd-go-line-2 ( -- )
```

8
lswt-1.txt

We now need to convert from integer counts to voltage knowing that the sensitivity is 125 $\mu$V for the least significant bit of the ADC result. Since we would like to do the calculation with integers on the microcontroller, it is convenient to work with scaled units of tenths of a millivolt and write the conversion as

$$V_{out} \left[ \frac{mV}{10} \right] = count \times \frac{125}{100}$$

To avoid losing too much precision, this calculation is done with the scale operator for mixed-precision operands, $m*/$. Once we have the voltage value in tenths of a millivolt, the output word, (d.1), displays the integer value in a format that puts a decimal point in front of the last digit, thus making the displayed result appear as a fractional number.
: mv10 ( n -- d )
  \ Convert 16-bit count to tenths of millivolts.
  \ With the divide-by-two resistors on the analog input,
  \ the least-significant bit corresponds to 62.5*2 microvolts
  s>d #125 #100 m*/

: (d.1) ( d -- )
  swap over dabs
  # [char] . hold #s rot sign #>

: report-mv ( n f -- ) \ Assuming decimal, print millivolt value
  if ." new " else ." old " then
  mv10 (d.1) type space ." mV "

: report-mv-lcd ( n -- ) \ Display millivolt value to LCD
  mv10 (d.1) lcd-type s" mV" lcd-type

: run-with-volts ( -- )
  decimal
  i2cinit
  begin
  mcp3422-go -1 #100 ms mcp3422@2byte 2dup
  cr ." chan 1 " report-mv drop \ S: ch1
  mcp3422-go -2 #100 ms mcp3422@2byte 2dup
  cr ." chan 2 " report-mv drop \ S: ch1 ch2
  swap \ S: ch2 ch1
  lcd-clear s" ch1 " lcd-type report-mv-lcd
  lcd-go-line -2 s" ch2 " lcd-type report-mv-lcd
  #500 ms
  key? until
  hex

;
The final stage of the firmware needs to do a bit more arithmetic to convert from voltage to pressure. If we continue to work in tenths of a millivolt \( \left( \frac{mV}{10} \right) \) and tenths of a Pascal \( \left( \frac{Pa}{10} \right) \), we can write the quadratic voltage-pressure relationship for the transducer [4] as

\[
\Delta p \left[ \frac{Pa}{10} \right] = \left( \frac{V_{out} \left[ \frac{mV}{10} \right] - 2500 \left[ \frac{mV}{10} \right]}{3750} \right)^2 \times 5000 \left[ \frac{Pa}{10} \right]
\]

In the Forth code below, we do the calculation in two steps:

\[
dV = V_{out} \left[ \frac{mV}{10} \right] - 2500 \left[ \frac{mV}{10} \right]
\]

and, using two scale operations,

\[
\Delta p \left[ \frac{Pa}{10} \right] = (dV \times \frac{5}{375}) \times \frac{dV}{3750}
\]
: report-pa-lcd ( n -- ) \ Display Pascal value to LCD
mv10 mv10-to-pa10 (d.l) lcd-type » Pa» lcd-type
;
: run-with-pa ( -- )
decimal
i2cinit
begin
mcp3422-go-1 #100 ms mcp3422@2byte 2dup
.cr ." chan 1 " report-mv drop dup \\ S: n1 n1
report-pa \ S: n1
mcp3422-go-2 #100 ms mcp3422@2byte 2dup
.cr ." chan 2 " report-mv drop dup \\ S: n1 n2 n2
report-pa \ S: n1 n2
swap \ S: n2 n1
lcd-clear s" ch1 " lcd-type report-pa-lcd
lcd-go-line-2 s" ch2 " lcd-type report-pa-lcd
#800 ms
key ? until
hex
;
References


