

# **The Networked Aircraft - Design, Construction And Flight Testing Of A Low-Cost, High-Performance Primary Flight Data Display**

by Bill Parodi

Principal Engineer, Massana Technologies

gparodi@massana.com

Since I started flying I have been impressed by how expensive aviation instruments are. Most primary instruments (airspeed, altitude, vertical speed) are mechanical instruments using archaic technology: The cockpit of a typical general aviation aircraft seems to have taken no advantage of the impressive cost reductions and reliability improvements that electronics have offered – unlike cars.

Recently, I decided to build an experimental aircraft. Being an electrical engineer, I decided to try and build, and install, the electronic equivalents to traditional aviation instruments and on my plane. This would help me find out if, as I suspected, we could take advantage of the last 20 years of electronic revolution for general aviation use.

The instrumentation system that I envisioned would cover most of the needs of the average VFR pilot. A primary flight data display (FDD) would give the pilot airspeed, altitude, and attitude information, a GPS moving map display would give the pilot situational awareness, and a graphical engine monitoring system would provide complete engine information. This report describes the design, construction and flight testing of the FDD.

## **System Requirements**

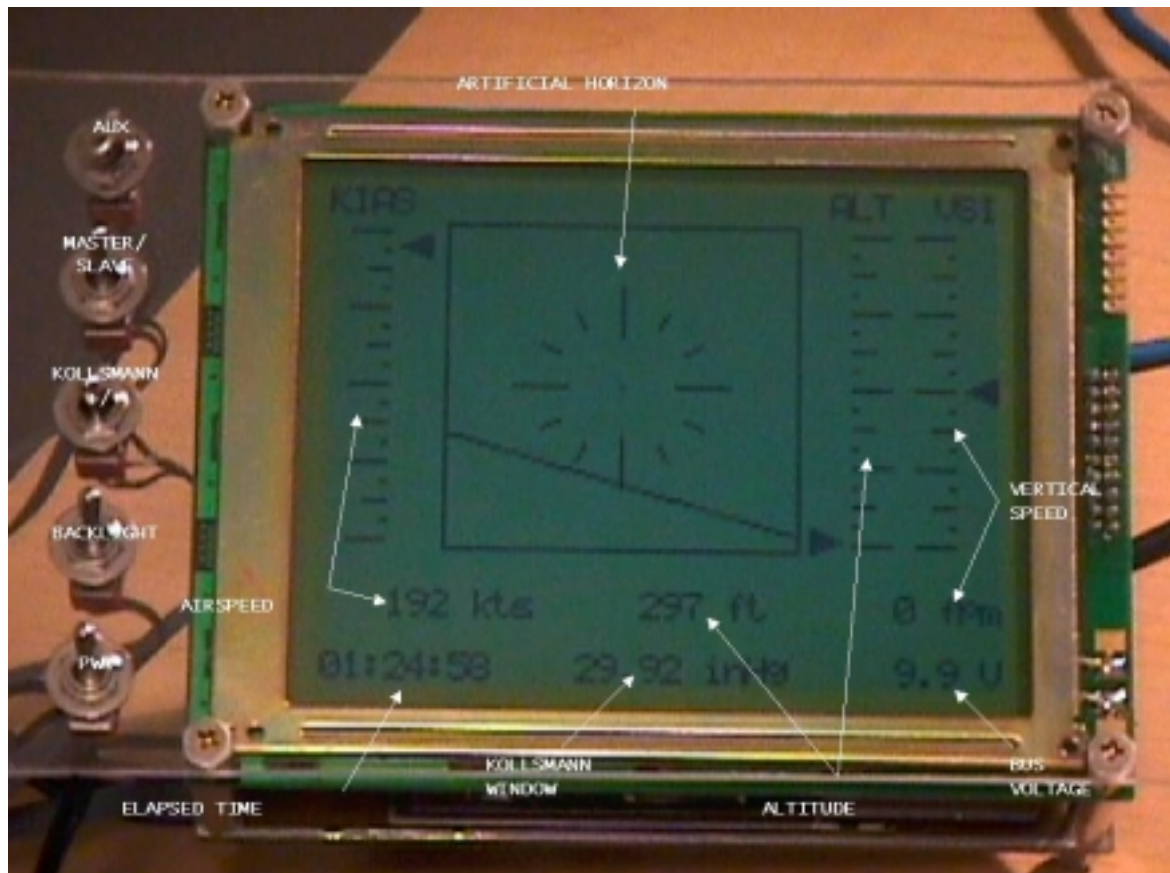
The most immediate requirement that the FDD has to satisfy is high reliability. The data displayed can be, under severe circumstances, of critical importance. And inaccuracy or operational failures are dangerous possibilities. The system would operate in a harsh environment, exposed to a wide temperature range, mechanical vibration, electromagnetic noise, direct sunlight, and other challenging external stimuli. Additionally, the system would have to be powered by the aircraft's unfiltered 12-V electrical system (it was deemed useful to have the FDD measure and display the bus voltage as well.)

As will be described later on in this report, the attitude indication was provided by a separate MEMS based solid state gyroscope (Crossbow DMU-HDX.) The location of the gyroscope is critical for its accuracy since it must be installed close to the centre of mass of the aircraft, with long data lines then routed to the FDD. I decided to use fiber optics to connect them. The data rate is far below that of which fiber optics are capable, but its low flammability, electromagnetic noise immunity, and short-circuit-safe nature rendered them ideal for the safety requirements of aircraft cabling.

Unlike the EFIS systems that are now becoming available for general aviation, this was supposed to be a cost-effective alternative to traditional aircraft instruments with its price nowhere near the four and five digit prices of the newer systems. The target cost was \$200 (excluding the gyro unit.)

## Primary (Visual) Flight Data Display

This picture shows what the FDD looks like and the information it displays:

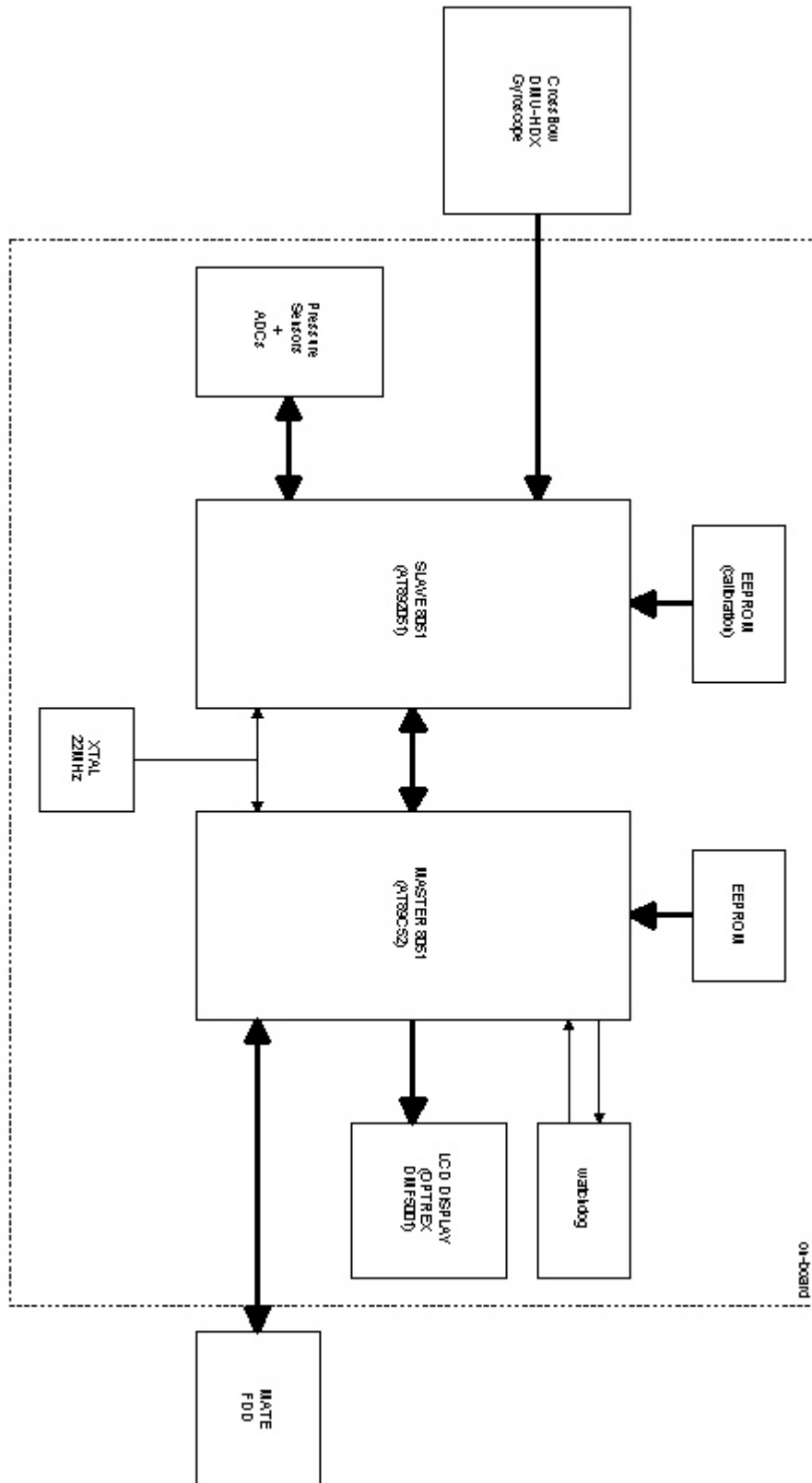


The FDD is located on the instrument panel, directly in front of the pilot. It would be ideal to overlay it to the pilot's view with a head-up display system, but I decided to leave that for a future upgrade!

The artificial horizon occupies the center of the screen. A grid shows the standard horizontal, 30°, 60°, and 90° left/right bank angles. The horizon itself is represented as a simple line. The airspeed is shown on the left side, both graphically (tape) and by number. The altitude is shown on the right side, inwards, also graphically and by number. The vertical speed is shown also on the right side graphically and by number. The Kollsman window (used for altimeter calibration) is shown on the bottom center (a panel switch is used to adjust this number.) Both an elapsed-time clock (time since instrument power-up) and bus voltage indicator are shown in the left and right bottom corners respectively.

### Architecture

The following figure shows the top-level architecture of the FDD. It contains two 8051-like processors (chosen for their ruggedness); one handles all the sensor data (slave), and the other handles the high level functions (master), like user interaction and graphics. The idea behind separating the two function groups was to give the master processor the freedom to obtain data from the slave processor on the same FDD or on a mate one (installed for the co-pilot) – as a sort of cross redundancy.

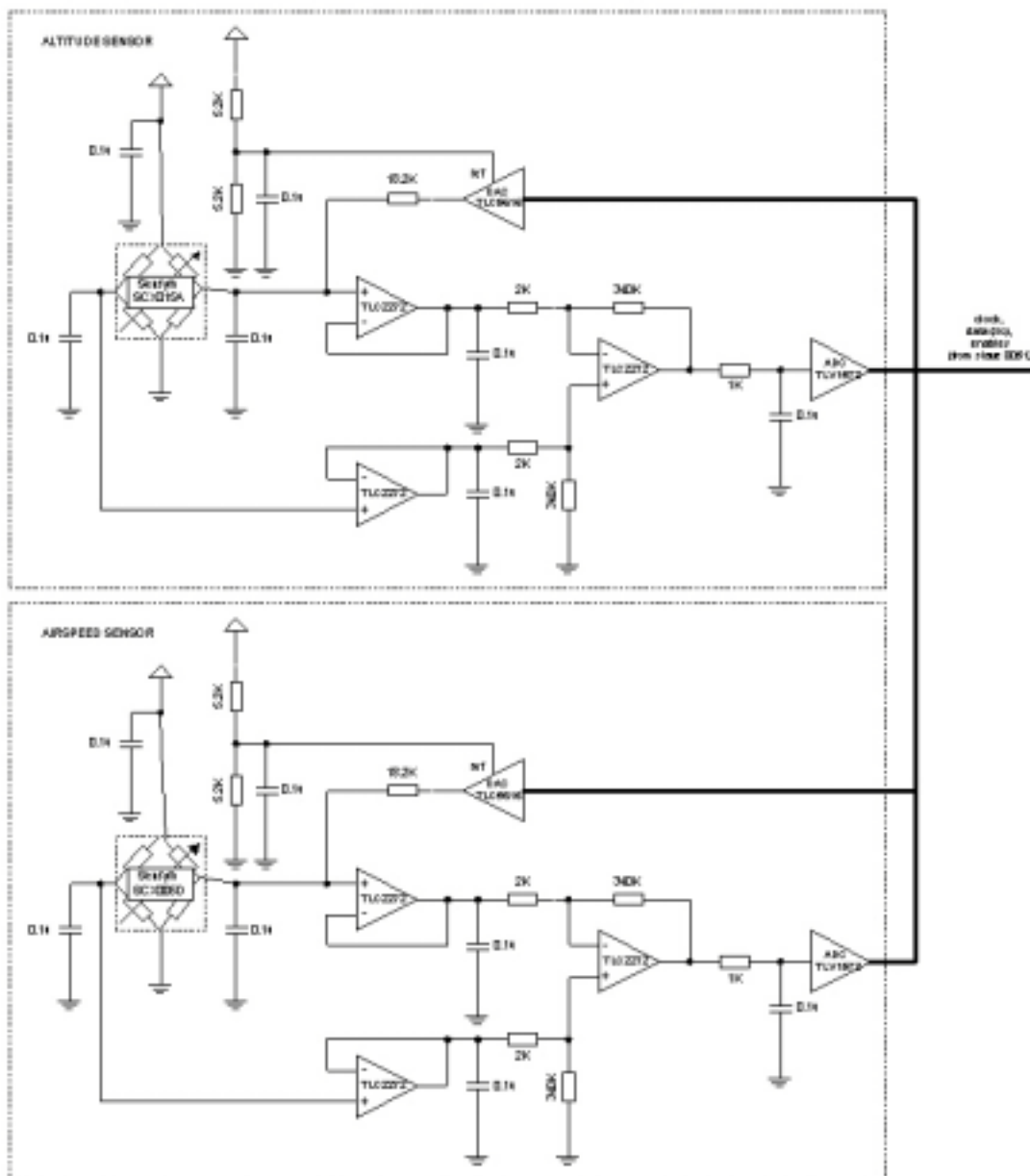


The master and slave 8051 processors communicate via a 2-wire serial link. The master processor polls the slave bit-by-bit in order to obtain a full flight data frame. Each line contains a header, followed by the 16-bit values of KIAS, pressure altitude,

roll angle, pitch angle, side slip (in g's), bus voltage, and a checksum (XOR) byte. The following table shows the flight data frame with indexed byte order.

1	2	3	4	5	6	7	8	9	10	11	12	13	14
HD R	KIAS		PRESS. ALT		ROLL		PITCH		SIDE SLIP		BUS VOLT.		CH K

The following figure shows the schematics of the pressure sensors subsystem of the FDD. Two pressure transducers are used to calculate the airspeed and altitude. An absolute pressure sensor with 0-15 psi range was used for altitude calculation, and a 0-5 psi differential sensor was used for calculating airspeed. It should be noted that at the time of writing this article, the airspeed indicator was not yet calibrated, but the altitude indicator was fully calibrated and showing acceptable precision.



The slave 8051 processor uses EEPROM calibration values for both, biasing the analog circuitry via DACs, and converting the ADC's outputs to standard altitude and airspeed measurements. As the slave 8051 processor is not aware of the user-supplied Kollsman adjustment value, the altitude that it calculates (and passes to the master 8051) is the pressure altitude (in feet.) The indicated airspeed calculated from the ADC measurement is an absolute number that is passed to the master 8051 without modification (in Knots.)

### **Artificial Horizon**

The artificial horizon is the most complex portion of the display. Fortunately, the Crossbow unit provides the attitude information in a very simple format. The FDD polls the gyro 8 times per second to have a smooth refresh on the screen. It should be noted that low-cost LCDs don't show motion very well, but with a careful adjustment of the contrast, the quality is acceptable in pretty much all lighting conditions.

### **Software**

The software on the main 8051 (master) handles user-interface functions, such as graphic output, Kollsman adjustment, and elapsed-time clock. It runs a main SW loop 8 times per second (8 Hz) for flight data update (including artificial horizon.) Two slower loops handle user input (4 Hz), and the elapsed-time clock (1Hz.) The main loop polls the slave processor for flight data. If the slave processor does not answer, or if the checksum fails on the reported flight data, the master resets the slave. Additionally, there is an on-board watchdog that the master has, to tag to prevent being reset. Events such as serial data reception, or counters reaching target values (used for loop timing) are handled by raising flags, the flags then being picked up by the main program loop. I chose this technique, rather than handling events with large event management traps, to ease implementation & debugging. If the traps are large, using several variables, the possibility of getting hit by an interrupt while servicing a prior one and corrupting the state is large, and the headaches indescribable.

The slave 8051 biases the analog circuitry for the pressure sensors, reads the sensor outputs (including the gyro), converts it to standard units (with interpolation to solve sensor non-linearity), and prepares the flight data for the master processor.

The following table shows part of the contents of the slave 8051 EEPROM. This section of the memory is used for the 17-entry altitude interpolation table. A similar table is used for KIAS interpolation.

ADDRESS	DESCRIPTION	DEFAULT VALUE	CURRENT VALUE
0x020,0x021	ALT interpolation table, entry 0	0x00,0x00	0xFE,0xF8
0x022,0x023	ALT interpolation table, entry 1	0x00,0x40	0xFE,0xF8
0x024,0x025	ALT interpolation table, entry 2	0x00,0x80	0xFE,0xF8
0x026,0x027	ALT interpolation table, entry 3	0x00,0xC0	0xFE,0xF8
0x028,0x029	ALT interpolation table, entry 4	0x01,0x00	0xFE,0xF8
0x02A,0x02B	ALT interpolation table, entry 5	0x01,0x40	0x04,0xD8
0x02C,0x02D	ALT interpolation table, entry 6	0x01,0x80	0x0A,0xF0
0x02E,0x02F	ALT interpolation table, entry 7	0x01,0xC0	0x11,0x30
0x030,0x031	ALT interpolation table, entry 8	0x02,0x00	0x1E,0xDC
0x032,0x033	ALT interpolation table, entry 9	0x02,0x40	0x26,0x70
0x034,0x035	ALT interpolation table, entry 10	0x02,0x80	0x2E,0x7C
0x036,0x037	ALT interpolation table, entry 11	0x02,0xC0	0x36,0xEC
0x038,0x039	ALT interpolation table, entry 12	0x03,0x00	0x3F,0xF2
0x03A,0x03B	ALT interpolation table, entry 13	0x03,0x40	0x3F,0xF2
0x03C,0x03D	ALT interpolation table, entry 14	0x03,0x80	0x3F,0xF2
0x03E,0x03F	ALT interpolation table, entry 15	0x03,0xC0	0x3F,0xF2
0x040,0x041	ALT interpolation table, entry 16	0x04,0x00	0x3F,0xF2

This code fragment shows how the slave uses these 17-point table interpolation tables:

```

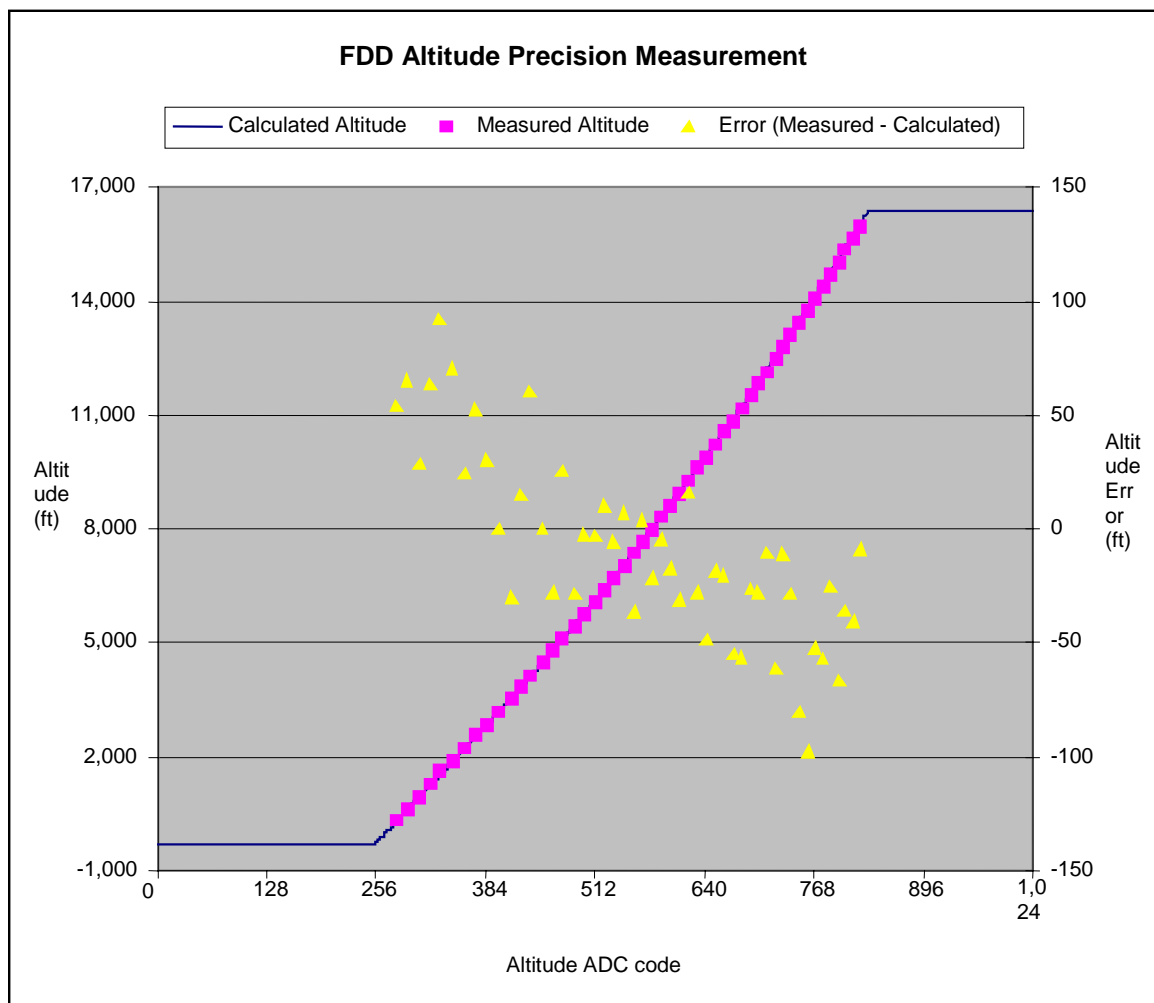
;-----
; interpolate: Interpolates a given 16 bit quantity with an EEPROM table.
;           Table contains 17 entries, each 16 bits, starting at address
;           'interp_tbl_addr'.
; Params:   [interp_in:interp_in+1] : value to adjust (altitude, airspeed, or voltage)
;           interp_tbl_addr         : base address in EEPROM of interpolation table
; Return:   [interp_out:interp_out+1] : interpolated value
; Stack:    2
; Notes:    uses ACC, R0, R1, R2, and globals ns1 and n12 w/o restoring original
values
;-----
interpolate:   mov   r0,#interp_in
              inc   r0
              mov   a,@r0
              dec   r0
              anl   a,#$3F
              rl   a
              rl   a
              mov   r0,#n11
              lcall conv_u8_to_u16    ; n11 = offset (=(ival%64)*4)/256)
              mov   r0,#interp_in
              mov   a,#5
find_entry:   lcall right_16
              djnz  a,find_entry

```

```

inc r0
mov a,@r0 ; a = table entry (=ival/64)*2
anl a,#$FE
add a,interp_tbl_addr
mov r0,a
mov r1,#n12
lcall read_eeprom_16 ; n12 = left value of table entry
inc r0
inc r0
mov r1,#interp_out
lcall read_eeprom_16 ; interp_out = right value of table entry
mov r0,#interp_out
mov r1,#n12
lcall subtract_16 ; interp_out = slope of table entry
mov r1,#n11
lcall mult_i16_fxp ; interp_out = value gain (offset * slope)
mov r1,#n12
lcall add_16 ; interp_out = result (left value + gain)
ret

```



This graph shows the precision obtained with the described altimeter. Note that the altimeter is within 100 ft. accuracy across the entire operating range.

## Flight Testing

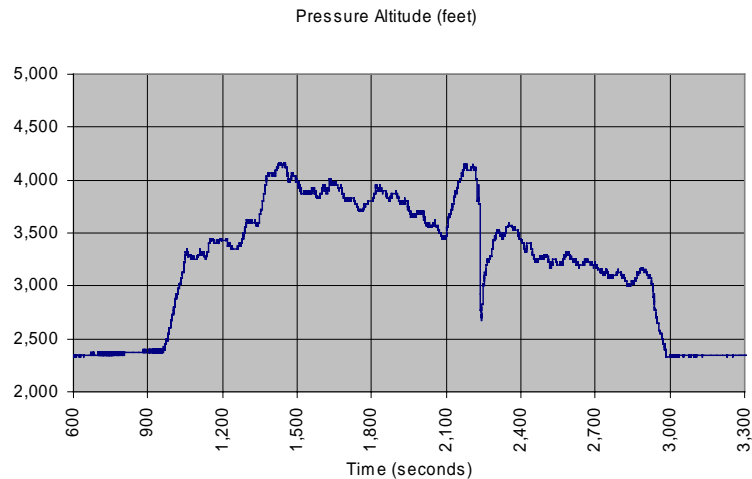
In order to validate the FDD we decided to conduct a series of flight tests. Both flight tests consisted of taxi, take-off, traffic pattern departure, normal flight manoeuvres (<math>60^\circ</math> bank), unusual attitude manoeuvres, entering traffic pattern, and landing. The first flight test was for qualitative analysis, to assess the “feeling” of the FDD and its viability for aircraft use. A more thorough second flight test was performed at a later date in which FDD data was logged into a laptop computer for further analysis.

The first flight test consisted of a qualitative assessment of the FDD, and it was conducted on a Lancair 360 (experimental) in Princeton Airport, NJ. The result of the first flight test was satisfactory, although several technical issues were identified. While on the ground, taxiing for takeoff, the gyroscope seemed to jump to a wrong attitude reading a couple of times, but that is understandable given the high-frequency high-amplitude vibration stimuli typical of taxiing (specially when in a bumpy taxiway.) This issue seemed to disappear once in flight. The refresh rate of 8 Hz was deemed marginally adequate, bearing in mind that the Lancair is a fast-response plane (roll rate exceeding 100 %/s.) Trying to test the gyroscope range we performed a full-aileron roll, but when we got back to straight and level flight, the gyro seemed to have lagged by about  $90^\circ$ , giving an incorrect indication of  $90^\circ$  bank. This was an early FDD prototype, and the problem must have been in the FDD rather than in the gyro itself because the second flight test yielded better results.

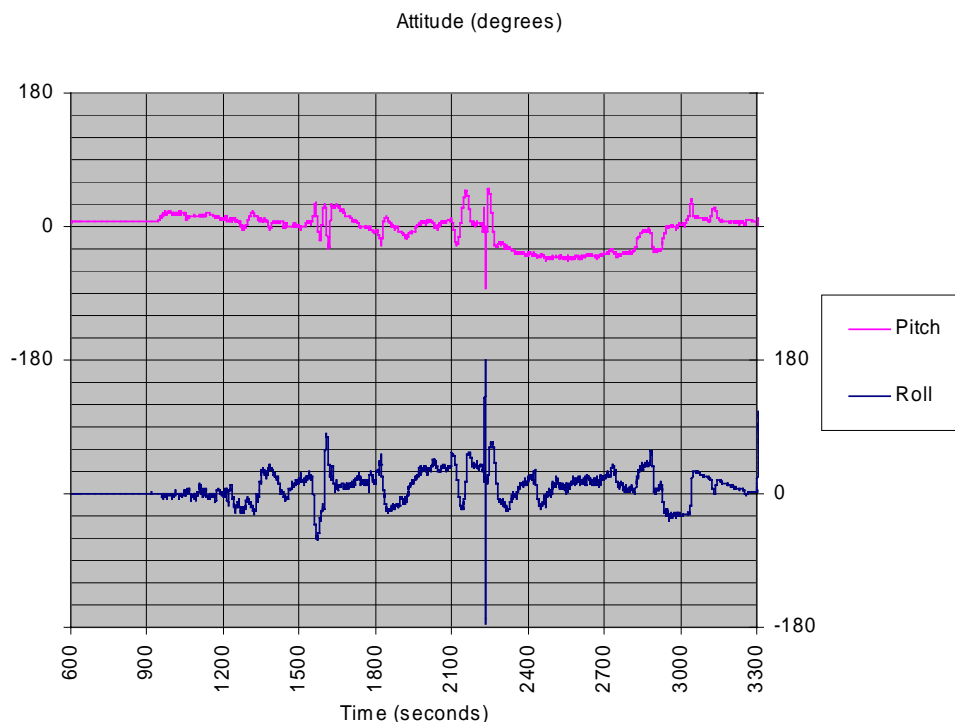
The second flight test was performed on a Piper Warrior PA28, in Cuatro Vientos, Madrid Spain. The plane used is shown in the picture below:



This second flight test consisted more of a quantitative analysis of the FDD. The altitude reading as well as pitch and roll angles were logged at a 1 Hz rate. Several technical issues were improved for this second flight test and a better response was noticed. The flight consisted on the steps mentioned before (taxi, takeoff, pattern departure...) for a total duration just short of 1 hour, including engine run-up time. The following graph shows the pressure altitude throughout the complete flight.



The flight was initiated at an MSL altitude of 2260 ft and 3200 ft was the altitude for entering and departing the pattern, and 4000 ft was chosen as the altitude for performing most of the test manoeuvres. The following graph shows both pitch and roll angles during the flight test. Correlation can be seen between pitch-up attitudes, like the ones in [1000 s, 1300 s], and positive increments in altitude, or pitch-down, like [2300 s, 2800 s] and altitude decrements. Power settings and speeds can account for the non-correlation between pitch and altitude (changes), but unfortunately airspeed was not logged as it was not calibrated at the moment of the flight test.



The time frame around 2250 s is particularly relevant because I attempted a gentleman's roll (constant +1g) at that time. Unfortunately, my lack of experience in PA28 unusual-attitude manoeuvres gave a less-than-elegant result, causing a significant loss in altitude due to a major pitch dive (90°); the manoeuvre ended more like the

bottom end of a loop than a roll. Lack of styles aside, the full  $-90^\circ$  pitch, and  $\pm 180^\circ$  roll range was tested in the gyro. As can be seen in the above graph, the gyro responded correctly. Notice the correlation of the  $+180^\circ$  to  $-180^\circ$  roll swing (corresponding to the inverted portion of the roll), and the  $-90^\circ$  pitch point (corresponding to finishing the roll as a loop from the inverted position.) Cleaner segments show the reliability of the gyro, like the left-banking roll angle to enter the right-hand pattern downwind leg (time segment [2900 s, 3050 s]), and the two right-banking roll angles for the turns to base and final legs (time segments [3050 s, 3100 s] and [3100 s, 3200 s] respectively.)

## **Conclusion**

We presented a low-cost Flight Data Display alternative for general (experimental) aviation. The key component (gyro) is still pricey, but being manufactured with MEMS technology, it should benefit from economy of scale and drop substantially in price as larger volumes are manufactured. The FDD prototype built was flight tested and proven to give satisfactory situational awareness, including altitude and attitude.

Further work remains to be done, including the calibration of the airspeed indicator and the implementation of the vertical speed indicator software. This FDD is currently being installed in a RANS S-16 experimental aircraft, due to fly for the first time this winter.

*For full flight logs and comparisons between the results given by the electronic FDD and conventional instruments contact Lee Goldberg at [lgoldberg@green-electronics.com](mailto:lgoldberg@green-electronics.com)*