MARK SIMS

FEI-5650A Comments

from TimeNuts mail list

I recently purchased some FEI FE-5650A option 58 rubidium oscillators. They are basically the same as the 5680A unit, but in a different (3x3x1.4 inch) form factor. The following applies to the oscillator versions with the Direct Digital Synthesis board that has the Analog Devices AD9830A DDS chip.

On the FE-5650A, the voltage regulator board/1PPS divider board is located on the bottom of the unit. There is a 20 pin high density ribbon cable connector. Most of the pins are either ground or +15V in. One pin is a 1 PPS (pulse per second) output. Another is an open collector oscillator lock indicator. There are a couple of other mystery pins that may remain unconnected. See the leapsecond.com site for the pinouts of this connector.

The +15V input pins are all connected to the input of a diode (located at one corner of the board) for reverse voltage protection. This diode has a very low forward voltage drop (around 0.15V after warmup). The output of the diode drives two LM2941 adjustable low dropout voltage regulator chips. They are set for a 14V output. They have around a 0.5V dropout.

The main power input to the module is specified at 15V to 18V. The regulators and base plate get rather warm at 15V input and it is difficult to attach an additional (and effective) heat sink to the unit. The rubidium physics package base plate is connected to the regulator base plate and contributes to the heat. I would not recommend running the unit at other than 15V. If you are powering the unit off of a 15V laptop wall wart power supply be aware of the voltage level at input at the unit. 15V - .2V - .5V leaves little headroom for voltage drop in the wiring. I get 14.85V at the input to the diode (14.65V during warmup) and all seems well (the regulator output voltages range from 13.90 to 13.97V on the units).

Connecting to the I/O connector is a major difficulty. The mating connector is difficult to find and work with. Instead I wired directly to the power board. The +15V is soldered directly to the input protection diode and the ground lead is soldered directly to one of the voltage regulator ground pins. If you do use the I/O connector, you must connect to all the power and ground pins or the 1.8A startup current will melt the tiny traces.

The DDS board requires an additional +5V for operation. I use a simple 7805 three terminal regulator driven from the +15V input power supply. I connect the +5V to the DDS board via the 5 pin RS-232 header.

The backside of the voltage regulator board contains a MAX913 comparator chip that converts the sine wave output from the DDS frequency synthesizer into a TTL/CMOS compatible square wave. It also contains the 23 stage binary divider chain that produces the 1 PPS output from the 8388608 Hz output. The divider chain is done with CMOS ripple counter chips... not the best way to get a 1PPS output with low jitter !

The 8388608 Hz signal from the DDS board is fed to the MAX913 comparator via a small coaxial cable. This cable is terminated on the divider board by a 50 ohm resistor to ground. If you are using the unit as a general purpose signal source, you should remove this resistor and terminate the signal at the end of your cable. Note that if you do remove the terminator, you should always remember to terminate your output connector with some kind of fairly low impedance load. The input to the MAX913 comparator chip is biased to around 1.25V with a voltage divider with a 100K ohm impedance. Without a low(ish) impedance load on the output, the sine wave output of the unit will be biased above ground.

The MAX913 comparator chip has two complementary outputs, Q and /Q. The Q output is used to drive the divider chain. I connected the /Q output (on pin 8 of the chip) to the outside world via a 50 ohm series termination resistor for use as a TTL level general purpose timing signal. I also brought out the original 1PPS output (which is only valid when the unit is programmed to 8388608 Hz). You could also bring out other taps from the binary divider chain.

The DDS board has five chips on the top side:

AD9830A DDS frequency synthesizer
PIC 16F84A microprocessor
SP233 RS-232 level converter
74AC161 counter chip used to make the PIC's clock
LM6182 dual 100 MHz current feedback op amp

and 3 chips on the bottom:

2 x 74VHC595 serial in / parallel out shift registers
74HC14 hex Schmidt trigger input inverter

You can download the data sheets for all these chips from datasheet4u.com

The AD9830A DDS frequency synthesizer chip is the heart of the board. It takes the 50.255+ MHz clock from the rubidium oscillator physics package and generates a sine wave with any frequency from DC to 25.127 MHz (actually 15MHz is a practical upper limit).

The sole purpose of the PIC microprocessor is to load the AD9830A synthesizer chip with its 32 bit frequency divisor value. It can get this value either from default values stored in its internal EEPROM or from the RS232 serial port. Additionally there are two pushbutton switches on the board that can be used to increment or decrement the stored default frequency divisor. The PIC microprocessor clock is generated by the 74AC161 counter chip. It divides the 50.255 MHz rubidium clock by 6 to make an 8.38 MHz clock.

The two 74VHC595 shift register chips on the back of the DDS board are connected as a 16 bit serial in / parallel out shift register. The PIC shifts the AD9830A divisor word into the shift register 16 bits at a time. When all 16 bits are shifted in, the PIC clocks the data into the shift register output buffer. The 16 parallel outputs of the shift register are then written to the AD9830A DDS chip. It takes two of these shift/write cycles to

update the 32 bit divisor register in the AD9830A.

Pinout usage of the PIC16F84A chip are as follows:

1 - RA2 - reads switch S401 (increase freq)
2 - RA3 - reads switch S402 (decrease freq)
3 - RA4 - flashes LED
4 - /MCLR - power on reset
5 - VSS - +5V
6 - RB0 - RS232 serial input data
7 - RB1 - shift register data output register clock
8 - RB2 - RS232 serial output data
9 - RB3 - shift register input data
10 - RB4 - AD9830A A0
11 - RB5 - AD9830A A1
12 - RB6 - AD9830A A2
13 - RB7 - AD9830A /WRITE
14 - VDD - GND
15 - OSC2 - PIC output clock
16 - OSC1 - PIC input clock from 74AC161
17 - RA0 - shift register input clock
18 - RA1 - not used???

There is a 6 pin connector that connects the DDS board to the physics package boards. Pin 1 is identified by a square pad:

1 - +14V
2 - GND
3 - +5V
4 - GND
5 - 50.255 MHz sine wave
6 - GND

The LM6182 is a dual 100 MHz current feedback amplifier. Current feedback amplifiers are magic in that their gain is not bandwidth dependent. The LM6182 is magic in that it can drive 100 mA of current.

The LM6182 contains two sections that are used to buffer and filter the output of the AD9830A DDS chip. The first section operates at a gain of 1.5X and is used to buffer the output of the 9830A chip. The second section operates at a gain of 2.5X and is used to drive the output cable. There are anti-aliasing filters on the outputs of each amplifier section. These filters can be modified to increase the maximum usable output frequency of the DDS board.

The LM6182 is designed to operate off of a split +/- voltage power supply, but on the FEI DDS board it operates off of a single +14V power supply. To do this, they created a virtual ground by using a resistive voltage divider (divide by 2) between +14V and ground. In order for this to work, the inputs and outputs of the LM6182 are capacitively coupled.

Four coupling capacitors are used. Two 0.1uF (?) caps are between the AD9830A I and /I outputs and the LM6182 first amplifier stage inputs. There is a 0.01 uF cap between the fist amplifier output and the second amplifier input and there is a 0.047 uF cap on the second amplifier output. These four coupling capacitors severely limit the lower frequency that the DDS board can operate at. They can be paralleled by much higher values to decrease the minimum frequency the board can operate at (I used 100uF tantalums). If you do this you will also have to parallel one of the DDS filter caps with a resistor (1.5K) to restore the sine wave output to its proper low distortion shape.

I have done some reverse engineering on them and have mine working with a fairly flat 4V pp output signal (2V with a 50 ohm load) from 10Hz to over 10MHz. They are usable with some signal loss from 1Hz to 15 MHz. Above 15MHz the signal drops off rapidly and gets very distorted.

According to my Tektronix AA501 audio distortion analyzer, at audio frequencies they produce a sine wave with around 0.1% total harmonic distortion (-60dbm THD). I don't have a spectrum analyzer to see what they look like at RF frequencies, but on a scope they look very clean.

When you send the "S" command to the DDS board serial port, the board responds with two values. An R=50255055.011982 reference frequency value and a F=2ABB5050621E4000 frequency divisor word (the exact values differ for each unit).

You can set the divisor word to any value with the F=xxxxxxxxxxxxxx command. The divisor word is stored to a precision of 64 bits (actually only 56 bits are set at the factory, the last 8 bits are always 00). Only the upper 32 bits are actually used by the module. If you send less than 16 hex digits, the remaining digits are not changed from their previous values.

Careful observation of the factory set values returned by several different modules has shown that THE R= REFERENCE FREQUENCY IS NOT THE FREQUENCY THAT THE PHYSICS PACKAGE IS SHIPPED TUNED TO!!! It is the frequency produced by the physics package at the minimum setting of the C-field potentiometer.

The way the factory sets up the units is to set the C-field potentiometer for minimum frequency from the physics package. They then measure this frequency against a primary standard and store it into the PIC chip. They then set the DDS synthesizer divisor word to 1 tick below the desired synthesizer output frequency (each tick of the DDS divisor word is about 0.01 Hz) and save that value as the F=divisor word. Finally, they increase the actual reference frequency to the desired value with the C-field potentiometer.

The best the DDS can offer is a frequency resolution of about 1 in 10E9. If your unit was shipped from the factory producing 8388608 Hz for dividing down to 1 PPS and you want to to produce 10.000 000 000 000 MHz, you will wind up about 0.005 Hz off unless you tweak the C-field.

If you intend to use the unit as a general purpose programmable frequency synthesizer, then for maximum accuracy of the output frequency, you should ideally measure the true

reference frequency (break out that hydrogen maser you got for Christmas). Next best is to calculate the true reference frequency from the saved (minimum C-field) R=reference frequency and F=divisor word and use that value to calculate divisor words. Third best is to just multiply the R=ref value by 1.000 000 002 150 (the average reference frequency scale factor of my units).

The notes from Don Latham and Bill Houlne published on the leapsecond.com web site, at http://www.leapsecond.com/museum/fei5650a/ have some inconsistencies that do not match my units. Also be aware that Bill and Don label pins 11..20 differently. These pins are based on Bill's drawing:

Pin 11 is not a frequency output pin
Pin 12 and Pin 14 are not Bit1 and Bit2 inputs, they are grounded
Pin 13 (lock indicator) is an open collector output... it needs a pullup resistor.

Don Latham suggests that the S401 and S402 switch inputs might be used to discipline the oscillator to a higher standard. This is probably NOT a good idea. Each push of a switch (real or simulated) causes the PIC to write a new divisor word into its internal EEPROM. The 16F84 EEPROM has a lifetime limit of around 1 million write cycles. Also the writes to the AD9830A chip are not synchronized to the output waveform. The output waveform glitches each time a new divisor word is written.

Mark Sims
Dallas, TX

I will post a semi-schematic of the changes to the DDS board. Right now I am working on adding a output level control.

****

I just posted my rather long dissertation on the FE-5650A and FE-5680A synthesizer board. A critical observation is that the F=reference frequency value returned by the "S" command is NOT the value that the unit is actually operating at as shipped from the factory. It is the value that the physics package operates at with the C-field pot set to its minimum value. If you attempt to set the output frequency value 10.000000000 MHz on a unit that was factory tuned to 8388608 Hz for a 1PPS output, your 10MHz will be around 0.005Hz off (the closest that the DDS divider can produce).

The C-field adjustment pot seems to be the one that is accessible through the hole in the side of the physics package that is close to the internal base plate... but I have not tweaked it on mine to be sure...

****

FEI FE-5650A and FE5680A code

Attached is my program for calculating the frequency divisor words for the FEI FE-5650A and FE-5680A rubidium oscillators with the AD9830A based DDS board. It compensates for the difference between the R=reference freq that the 'S' command returns (the minimum

C-field value) and the frequency that the oscillator was actually shipped tuned to.

The program does not actually read and write the serial port directly (would be a nice mod though). You will need to recompile it with the values returned by your oscillator 'S' command. There are two #defines at the start of the program.

Usage is:

```
freq 10    (for 10Hz)
freq 10 K  (for 10KHz)
freq 10 M  (for 10MHz)
```

```
#include "stdio.h"
#include "math.h"
#include "string.h"

//  This program calculates the Analog Devices 9830A DDS frequency divisor
//  values for programming the FEI FE-5650A and FE-5680A rubidium oscillator
//  units.
//
//  This program was written by Mark S Sims  May 2008.  It is relased to the
//  public domain.  Use it in peace.  Plase share with others what you do
//  with it.
//

// These are the reference freq and divisors returned by the FEI "S" command
// They vary from unit to unit.  Change these to whatever your unit returns (or
// modify this program to interrogate the unit with the "S" command.  While you
// are at it,  modify this program to send the divisor word to the oscillator.
//
#define R  50255055.011982
#define F  0x2ABB5050L

#define FF ((double) 8388608.0)      // the freq the module was calibrated for
#define M  ((double) 4294967296.0)   // 2^32

double r=R;    // working reference frequency
double f=FF;   // working DDS divisor word


void lll(double x)
{  // print divisor as a 64 bit hex value
char s[20];
int i;

   for(i=0; i<16; i++) s[i] = 0;
```

```
   i = 0;

   x = x * M * M;
   while(x>0.99) {
      s[i++] = fmod(x, 16);
      if(i > 16) break;
      x = x / 16.0;
   }

   for(i=15; i>=0; i--) {
      printf("%X", s[i]);
      if(i == 8) printf(":");
   }
   printf("\n\n");
}

main(int argc, char *argv[])
{
double d;
double x;

   if(argc > 1) {  // get desired freq from command line
      sscanf(argv[1], "%lf", &f);
      if(argc > 2) {  // if more than one command line arg,  freq is in MHz or KHz
         if(toupper(argv[2][0]) == 'K') f *= (double) 1000.0;
         else if(toupper(argv[2][0]) == 'H') f *= (double) 1.0;
         else f *= (double) 1000000.0;
      }
      else if(strchr(argv[1], 'K')) f *= (double) 1000.0;

      else if(strchr(argv[1], 'k')) f *= (double) 1000.0;
      else if(strchr(argv[1], 'M')) f *= (double) 1000000.0;
      else if(strchr(argv[1], 'm')) f *= (double) 1000000.0;
   }

   // print input data values
   printf("\nR=%19.10lf  F=%08lX   (f=%lf)\n\n",  R,F,FF);

   // calculate what freq the R= and F= values returned from the
   // oscillator would produce
   d = ((double) F) / M;
   d *= r;
   printf("R*F/(2^32)=%19.10lf  ref_scale=%.14lf\n", d, FF/d);

   // scale the R= reference freq by that value to get the actual
```

```
   // oscillator reference freq
   r = r * FF / d;
   printf("actual ref=%.10lf\n\n", r);

   // show what the 64 bit DDS divisor should be
   printf("freq=%.10lf\n\n", f);
   d = f / r;
   printf("freq/ref=%.19lf\n", d);

   printf("full divisor=");
   lll(d);


   // calculate the closest 32 bit divisor that the AD9830A DDS chip
   // will actually use
   d *= M;
   d = (double) (unsigned long) (d+0.5);


   // now show the actual generated frequency and the two closest ones
   x = r * (long) (d-1);
   x /= M;
   printf("freq(%08lX) = %.10lf  err=%13.10lf (%.3lg)\n", (long)d-1, x,  x-f, (x-f)/f);

   x = r * (long) d;
   x /= M;
   printf("freq(%08lX) = %.10lf  err=%13.10lf (%.3lg)\n", (long)d, x,  x-f, (x-f)/f);

   x = r * (long) (d+1);
   x /= M;
   printf("freq(%08lX) = %.10lf  err=%13.10lf (%.3lg)\n", (long)d+1, x,  x-f, (x-f)/f);

   printf("\n");
   printf("nearest divisor=%08lX\n", (unsigned long) d);
}
```

****

Heat sinking for the FEI FE-5650A

Decent thermal management of the FE-5650A can be a problem. It is a small, compact object dense with electronics and the rubidium physics package that runs at over 100 degrees C. There are two metal plates in the unit. One is the baseplate of the physics package. It is bolted edge on to the voltage regulator baseplate that forms the bottom (well actu1 ally one edge) of the unit.

It does not look like the FE-5650A is meant to attach to a heat sink by securing the large

flat face of the regulator baseplate to a heat sink. There are no mounting holes on it for securing it flat face down (that would also block the 20 pin cable connector).

The only mounting holes on the regulator baseplate are along the edge of the plate where it would be rather difficult to attach an effective heat sink. The holes along three edges of the baseplate are used for attaching the cover. This leaves only one edge for mounting the device to a chassis. This gives only a tiny amount of surface area for heat transfer.

The only way to get the regulator baseplate in effective contact with a heat sink is to not use the 20 pin cable/connector and to remove the two countersunk screws that secure the regulator baseplate to the edge of the physics package baseplate. By replacing these two screws with longer ones, a heat sink could be attached to the regulator baseplate. It would be even better to drill and tap a couple more screw holes so that better contact can be made to the plate. Also good thermal transfer compound should be used.

****

Operating temperatures of rubidium oscillators

I recently measured the operating (baseplate) temperatures of several different rubidium standards. These are all in free air without heat sinks. Results are as follows (degrees C over ambient, warmest point found on device):

FRK-L:	+21C
M100:	+18C
LPRO:	+24C
5650A:	+35C
PTB100:	+12C

The FRK-L mounted on a heat sink with around 80 square inches of heat sink fins had a 11C rise. The fins were in contact with a table, with the device pointing up.

The LPRO was operated with the baseplate in a vertical orientation at 24V. The FRK and M100 with the baseplate in contact with a table at 27V. The 5650A was operated at 15V with the baseplate vertical. The PTB-100 is essentially an FRK-H with a heat sink that is mounted in a Tektronix TM-500 module.

It appears all the devices can be safely operated in a room temperature environment without heat sinks, but for highest reliability and longest life one should provide some form of heat control. The FRK/M100/LPRO would probably be fine with their base plates secured to a metal chassis. The FEI-5650A is very close to its maximum specified operating temperature (65C). It would probably exceed the spec if enclosed in a case without airflow. Just a little moving air goes a long way to keeping things cool. Passive cooling with just a heat sink tends to be rather ineffective.

****

E-5680A C-field trimer ??

Actually the 5680A and 5650A units that have the DDS do have a C-field pot. The DDS is only accurate to 0.01 Hz. They set the C-field to minimum, program the DDS constant that gets the closest to the desired freq into the CPU chip (this is what gets reported by the "S" command), then they tweak the C-field to get the unit exactly on freq. The reference freq reported by the "S" command is NOT the reference freq that the unit is shipped adjusted to.

****