```
/**
 * Simple rotary encoder tuned, I2C SSD1306 OLED, Si5351
 * 4 KHz to 150 MHz
 * V 1.0.0 ND6T  18 January 2020
 * This source file is under General Public License version 3.0
 * Interrupt driven tuning. dual PE4302 variable attenuator controls
 *
 * Pin Connections:
 *      Si5351 & Display: SDA = A4, SCL = A5
 *      Encoder: A = D2, B = D3, switch = D4
 *      Attenuator: V1 through V6; D5 through D10
 *      Selector switch = D12
 */
#include <Rotary.h> //Available at: https://github.com/brianlow/Rotary
#include <Adafruit_SSD1306.h>
#define OLED_RESET 4
Adafruit_SSD1306 display(OLED_RESET);
#include <si5351.h>  //Etherkit Si5351 library v 2.0.6
(https://github.com/etherkit/Si5351Arduino)
Si5351 si5351;

  Rotary r = Rotary(3,2);//Encoder to pins D2,D3
  unsigned char result;  //Encoder results
  int x=0;               //Utility variable
  int ind;               //Tuning position indicator
  int oldind;            //Previous tuning indicator
  int pos=2;             //Cursor position
  long attn = 10;        //Initial attenuation
  long incr = 1000;      //Initial tuning increment
  long upperLimit =15e7; //Upper frequency limit
  long oldFQ;            //Frequency change reference
  long long FQ = 7e6;    //Starting frequency
  float dbm=0;           //Result in dBm
  float atn;             //To enable floating point math
  float uv=0;            //result in microvolts
  long long post;        //Time post for sensitive delays

#define sw1 4   //Tuning increment switch
#define sw2 12  //Frequency/level selector

void setup(){
  PCICR |= (1 << PCIE2);//Interrupt setup
  PCMSK2 |= (1 << PCINT18) | (1 << PCINT19);//Matrix "state machine" decode
  r.begin();

  display.begin(0x3C);  // initialize with the OLED I2C addr

  si5351.init(SI5351_CRYSTAL_LOAD_8PF,25000830L,0); //Ref osc freq.
  si5351.set_pll(SI5351_PLL_FIXED, SI5351_PLLA);
  si5351.set_freq(FQ * 100, SI5351_CLK1);   //Program the synthesizer

  pinMode(sw1,INPUT_PULLUP);//Tuning increment sw1itch
  pinMode(sw2,INPUT_PULLUP);//Frequency/level selector

  pinMode(5,OUTPUT);   // 1 Db control pin (V1) both attenuators
  pinMode(6,OUTPUT);   // 2 dB             (V2)      "
  pinMode(7,OUTPUT);   // 4 dB             (V3)      "
  pinMode(8,OUTPUT);   // 8 dB             (V4)      "
  pinMode(9,OUTPUT);   // 16 dB            (V5)      "
```

```
  pinMode(10,OUTPUT);  // 32 dB            (V6)      "

 ////////////Splash//////////
  display.clearDisplay();
  display.setTextSize(2);
  display.setTextColor(WHITE);
  display.setCursor(0,0);
  display.print("EZgen4    V.1.0.0");
  display.display();
  delay(3000);
  post=millis();
  }

void loop(){
if(millis()-post<50) show(); //Display if changed less than 50 ms ago
  atn = attn;                //Recast to float for display
  dbm= -88 - atn;            //Assuming -88 dBm available through fixed pads
  output();                  //Write to the attenuator
  uv=(pow(10,(dbm/20)))*(100000*(sqrt(5)));    //Convert dBm to microvolts
  post=millis();             //Display results
  }

//*****************FUNCTIONS (subroutines)***********************************
void show(){    //Show 'em what you have!
  long fq=FQ;   //Re-cast to make it printable
  int mhz=(FQ/1e6);                //Truncate MHz
  long hz=FQ-(mhz*1000000);
  display.clearDisplay();
  display.setTextSize(2);
  display.setTextColor(WHITE);
  display.setCursor(0,0);
  if(mhz<10) display.print("0");

 if(FQ>=1e8)display.print(fq);          //If over 100MHz shorten display to fit
    else{
    display.print(mhz);                 //Parse at each 3 digits for easy reading
    if(FQ<1e8)display.print(" ");
     if((hz/1e3)<10) display.print("0");//Insert leading zeros if necessary
     if((hz/1e3)<100) display.print("0");
     display.print((hz)/1e3,3);         //Parse final 6 digits
    }
  display.setTextSize(2);
  display.setCursor((ind-1)*12,9);
  display.print("-");
  display.setCursor(0,17);
  display.print(uv,2);
  display.setTextSize(1);
  display.print("microVolt");
  display.setCursor(70,25);
  display.print(dbm,0);
  display.print(" dBm");
  display.display();
  si5351.set_freq(FQ * 100, SI5351_CLK1);//Program the synthesizer
 }

void output(){
   int a=attn    %2;   // calculate LSB
   int b=attn/2  %2;
   int c=attn/4  %2;
```

```
   int d=attn/8  %2;
   int e=attn/16 %2;
   int f=attn/32 %2;   // MSB

   digitalWrite(5,a);  //Write binary to control pins
   digitalWrite(6,b);
   digitalWrite(7,c);
   digitalWrite(8,d);
   digitalWrite(9,e);
   digitalWrite(10,f);
   }

ISR(PCINT2_vect){      //Encoder Interrupt service routine
 result = r.process();
 if(digitalRead(sw2)==HIGH){  //If switch is in "frequency" position
 if(digitalRead(sw1)==HIGH){     //If tuning knob is not pressed
     if(result == DIR_CW){
       FQ+=incr;                      //Clockwise. Add the increment
       if(FQ>upperLimit)FQ=upperLimit;//Unless it exceeds upper limit
    }
    if(result == DIR_CCW){            //CounterClockwise subtract
        if(((FQ)-incr)>=4000)FQ-=incr;//But keep it above 4 KHz
        }
    }
    else{     //If the tuning knob is pressed then move the cursor
        if(result == DIR_CW){ //Move cursor right
        if(pos<8)++pos;
         }
        if(result == DIR_CCW){ //Move cursor left
        if(pos>1)--pos;
        }
    }

  ind=pos;               //Correlate indicator to position

  if(FQ<1e8){            //If under 100 MHz
    if(ind>2)ind=pos+1;  //Compensate for decimal place
    if(ind>6)ind=pos+2;

  if(ind>9)incr=1;      //Set increment by cursor position.
  if(ind==9)incr=10;
  if(ind==8)incr=100;
  if(ind==6)incr=1e3;
  if(ind==5)incr=1e4;
  if(ind==4)incr=1e5;
  if(ind==2)incr=1e6;
  if(ind==1)incr=1e7;
  }
    else{                //If over 100 MHz
  if(ind==9)incr=1;      //Set increment by cursor position.
  if(ind==8)incr=10;
  if(ind==7)incr=100;
  if(ind==6)incr=1e3;
  if(ind==5)incr=1e4;
  if(ind==4)incr=1e5;
  if(ind==3)incr=1e6;
  if(ind==2)incr=1e7;
  if(ind==1)ind=2;
    }
```

```
if(oldind!=ind){         //If tuning range changed display it.
  oldind=ind;
  post=millis();
  }
}
else{                     //If switch is in "level" position
    if(result == DIR_CCW){
       ++attn;             //CounterClockwise. Add a dB
        if(attn>63)attn=63;//Unless it exceeds upper limit
     }
     if(result == DIR_CW){
       --attn;             //Clockwise. subtract a dB
        if(attn<0)attn=0;  //Unless it's below zero
      }
    }
  }
```