```
/*

AD9833 DDS Waveform Generator by Glen Popiel - KW5GP



 This program is free software: you can redistribute it and/or modify
 it under the terms of the version 3 GNU General Public License as
 published by the Free Software Foundation.

 This program is distributed in the hope that it will be useful,
 but WITHOUT ANY WARRANTY; without even the implied warranty of
 MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
 GNU General Public License for more details.

 You should have received a copy of the GNU General Public License
 along with this program.  If not, see <http://www.gnu.org/licenses/>.



Based on AD9837 Pro Generator sample code written in Arduino 1.0.1 for an
Arduino Pro Mini, 5V, 16MHz
by Pete Dokter, 9/2/12, Remixed by Anne Mahaffey, 10/8/12, ReRemixed by
sinneb, 15th of april 2013

The connections to the AD9833 board are:

FNC -> 2 (FSYNC)
CLK -> 13 (SCK)
DAT -> 11 (MOSI)
VCC -> VCC
GND -> GND
OUT -> Ouput
REF - N/C

*/

#include <SPI.h>  // SPI Library
#include <Wire.h>    //I2C Library
#include <LiquidCrystal_I2C.h>  // Liquid Crystal I2C Library

#define FSYNC 2  // Define FSYNC Pin as Digital Pin 2
#define Square_wave 3  // Define Square Wave Switch Input as Digital Pin
3
#define Triangle_wave 4  // Define Triangle Wave Switch Input as Digital
Pin 4
#define min_freq 50  // Define the minimum frequency as 50Hz
#define max_freq 10000  // Define the maximum frequency as 10 Khz
#define frequency_pot A0

long debounce = max_freq/1000;  // debounce the frequency control pot
long freq; //32-bit global frequency variable
long previous_frequency = 0, desired_frequency ;  // variables for
frequency selection
```

```
int wave_type, previous_type = 0, wave_data = 0x2000;  // variables for
waveform selection

const int lcd_end = 16; // set width of LCD
const int lcd_address = 0x27; // I2C LCD Address
const int lcd_lines = 2; // Number of lines on LCD

LiquidCrystal_I2C lcd(lcd_address,lcd_end,lcd_lines);  // set the LCD I2C
address to 0x27 for a 16 chars and 2 line display

void setup()
{
  lcd.init();  // initialize the LCD
  lcd.backlight();  // Turn on the LCD backlight
  lcd.home();  // Set the cursor to line 0, column 0

  lcd.print(" KW5GP Waveform");  // Display the startup screen
  lcd.setCursor(3,1);
  lcd.print("Generator");
  delay(3000);
  lcd.clear();

  lcd.print("Freq: ");  // Set up the LCD display
  lcd.setCursor(0,1);
  lcd.print("Sine Wave");

  pinMode(FSYNC, OUTPUT); // Set the FSYNC Pin as an Output
  pinMode(Square_wave, INPUT);  // Set the Square_wave Control Pin as an
Input
  pinMode(Triangle_wave, INPUT);  // Set the Triangle_wave Control Pin as
an Input
  digitalWrite(Square_wave, HIGH);  // Enable the Internal Pullup
Resistor on the Square wave Control Pin
  digitalWrite(Triangle_wave, HIGH);  // Enable the Internal Pullup
resistor on the Triangle wave Control Pin

  digitalWrite(FSYNC, HIGH);  // Set FSYNC High - diables input the the
AD9833

  SPI.setDataMode(SPI_MODE2); // required SPI Mode for AD9833
  SPI.begin();  // Start the SPI bus

  delay(100); //A little set up time, just to make sure everything's
stable

}  // End Setup Loop

void loop()
{
  wave_type = 0;  // Default to Wave Type 0 - Sine Wave
  if (digitalRead(Square_wave) == LOW)  // Check the Square Wave switch
pin
  {
    wave_type = 1;  // Set the Wave Type to 1 for a Square wave
```

```
  }
  if (digitalRead(Triangle_wave) == LOW)  // Check the Triangle Wave
switch pin
  {
    wave_type = 2;  // Set the Wave Type to 2 for a Triangle Wave
  }
  desired_frequency =
map(analogRead(frequency_pot),1,1020,min_freq,max_freq);  // Read the
frequency pot to determine desired frequency

  // Update the DDS frequency if we've changed frequency or wave type
  if (desired_frequency > (previous_frequency+debounce) ||
desired_frequency < (previous_frequency - debounce) || (wave_type !=
previous_type))
  {
    WriteFrequencyAD9833(desired_frequency);  // Call the Function to
change frequency and/or waveform type
    previous_frequency = desired_frequency;  // Update the frequency
variable
    previous_type = wave_type;  // update the wave type variable
    lcd.setCursor(0,1);  // Display the Wave Type on the LCD
    switch(wave_type)
    {
      case 0:  // Type 0 = Sine Wave
        lcd.print("Sine Wave    ");
        break;

      case 1:  // Type 1 = Square Wave
        lcd.print("Square Wave  ");
        break;

      case 2:  // Type 2 = Triangle Wave
        lcd.print("Triangle Wave");
        break;
    }
  }
}  // End Main Loop

void WriteFrequencyAD9833(long frequency)  // Function to change the
frequency and/or waveform type
{

  //
  int MSB;  // variable for the upper 14 bits of the frequency
  int LSB;  // variable for the lower 14 bits of the frequency
  int phase = 0;  // variable for phase control

  //We can't just send the actual frequency, we have to calculate the
"frequency word".
  //This amounts to ((desired frequency)/(reference frequency)) x
0x10000000.
  //calculated_freq_word will hold the calculated result.
  long calculated_freq_word;  // variable to hold the calculated
frequency word
```

```
  float AD9833Val = 0.00000000;  // variable used to caluclate the
frequency word

  AD9833Val = (((float)(frequency))/25000000);  // Divide the desired
frequency by the DDS Reference Clock
  if (wave_type == 1)  // Divide the frequency by 2 if we're generating
Square waves
  {
    AD9833Val = AD9833Val * 2;
  }
  calculated_freq_word = AD9833Val*0x10000000;  // Finish calculating the
frequency word

  lcd.setCursor(6,0);
  lcd.print(String (frequency) + " Hz   ");  // Display the current
frequency on the LCD


  //Once we've got the calculated frequency word, we have to split it up
into separate bytes.
  MSB = (int)((calculated_freq_word & 0xFFFC000)>>14); // Upper 14 bits
  LSB = (int)(calculated_freq_word & 0x3FFF);  // Lower 14 bits

  LSB |= 0x4000; // Set control bits DB15 ande DB14 to 0 and one,
respectively, to select frequency register 0
  MSB |= 0x4000; // You have to do this for both frequency words

  phase &= 0xC000;  // Set the Phase Bits (defaults to 0)

  WriteRegisterAD9833(0x2000);  // Set the Control Register to receive
frequency LSB and MSB in consecutive writes

  //Set the frequency
  WriteRegisterAD9833(LSB); //lower 14 bits  // Write the lower 14 bits
to the Frequency Register

  WriteRegisterAD9833(MSB); //upper 14 bits  // Write the upper 14 bits
to the Frequency Register

  WriteRegisterAD9833(phase); //mid-low  // Write the phase bits to the
Phase Register

  switch (wave_type)  // Select the correct Register settings for the
desired Waveform
  {
    case 0:  // Sine Wave
    wave_data = 0x2000;
    break;

    case 1:  // Square Wave
      wave_data = 0x2020;
      break;

    case 2:  // Triangle Waqve
```

```
      wave_data = 0x2002;
      break;
  }
  WriteRegisterAD9833(wave_data);  // Write the Waveform type
}

//This is the guy that does the actual talking to the AD9833
void WriteRegisterAD9833(int dat)  // Function to write the data to the
AD9833 Registers
{
  digitalWrite(FSYNC, LOW); //Set FSYNC low - Enables writing to the DDS
Registers

  SPI.transfer(highByte(dat)); // Send the High byte of data
  SPI.transfer(lowByte(dat)); // Send the Low byte of data

  digitalWrite(FSYNC, HIGH); //Set FSYNC high  - Disable writing to the
DDS Registers
}
```