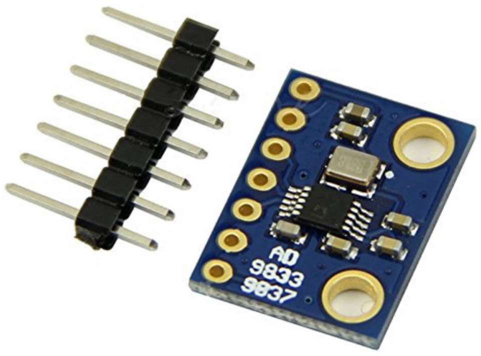
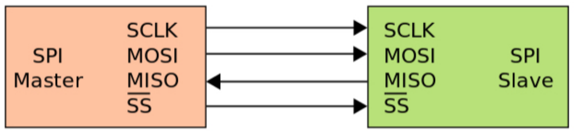
31. Application - AD9833 Audio SIggen

Using a very similar signal generation to the AD9851, the small AD9833 can generate triangle, square or sine wave outputs with two frequency registers, and phase registers.



The AD9833 module uses an SPI bus interface, rather than the I2C we have seen previously. The SPI (Serial Peripheral Interface) bus is a faster one than I2C, but needs more connections (4 vs 2). These are:

SCLK - a serial data transfer clock
MOSI - the Master Out Slave In data
MISO - the Master In Slave Out data
/SS - a chip select (allowing more than one device on the bus)



When used with the Arduino UNO specific pins are used for the clock and data, and any additional pin for the chip select.

MOSI - 11
MISO - 12
SCLK - 13

and for a single connection it is convenient to use 10 as /SS chip select.

Connections

VCC	+5V
DGND	GND
SDATA	MOSI (12)
SCLK	SCLK (13)
FSYNC	/SS (10)
AGND	GND
OUT	Audio output (use serial cap to remove DC level)

Setup

```
#include "AD9833.h"                                      // include library
#include "Rotary.h"                                      // include library
#include "LiquidCrystal_I2C.h"

#define CLK 2                                            // encoder connnections for interrupt
#define DT 3                                             // reverse 2&3 if wrong direction
#define SW 4                                             // step switch
#define SS 10                                            // SPI chip enable pin

#define LCDADDR 0x3F                                     // LCD I2C address

#define FREQMIN 10                                       // 10Hz to 100kHz
#define FREQMAX 100000

AD9833 audio(SS);                                        // create audio object, SS chip select
Rotary enc = Rotary(CLK, DT);                            // create encoder object
LiquidCrystal_I2C lcd(LCDADDR, 16, 2);                   // create lcd object

float freq = 1000.0;                                     // init 1kHz
float freqStep = 100.0;                                  // init step 100Hz
```

The AD9833, Rotary Encoder and Liquid Crystal display libraries are used. Rotary encoder pin connections and the chip enable for the AD9833 are defined, together with the LCD display I2C address (0x3F - which could be different for your display). Next the three device objects are created. And two global variables defined for the audio frequency and tuning step.

setup()

This is quite simple, pin connections are made for the encoder and push button. The audio and lcd are initialised, and the audio output is chosen as a sine wave output, and set to frequency, using the AD9833 Register 0 for storage of the value.

```
void setup() {
  pinMode(CLK, INPUT_PULLUP);                            // encoder inputs, with pull-ups
  pinMode(DT, INPUT_PULLUP);
  pinMode(SW, INPUT_PULLUP);                             // step switch

  audio.Begin();                                         // init audio
  lcd.begin();                                           // init LCD
  audio.ApplySignal(SINE_WAVE, REG0, freq);              // output sine wave, using REG0

  dispUpdate();                                          // display freq & step
}
```

Loop()
The loop is almost the same as that for RF generator using the AD9851, two sections detect a button push and change the step, and the second detects rotation of the encoder and changes the frequency

```
void loop() {
  int result;

  if (digitalRead(SW) == LOW) {                          // step button push
    while (!digitalRead(SW));                            // wait release
    if (freqStep == 10) freqStep = 10000;                // update step Hz, min 10Hz, max 10kHz
    else freqStep = freqStep / 10;                       // step down
    dispUpdate();
  }

  result = enc.process();
  if (result == DIR_CW && freq < FREQMAX) {              // freq up
    freq += freqStep;
    dispUpdate();
  }
  if (result == DIR_CCW && freq > FREQMIN) {             // freq down
    freq -= freqStep;
    dispUpdate();
  }
}
```

Display
The display is extremely simple. Just the frequency (in Hz) and the step.