Author: Weihui Liu

Application Note about Signal Processing

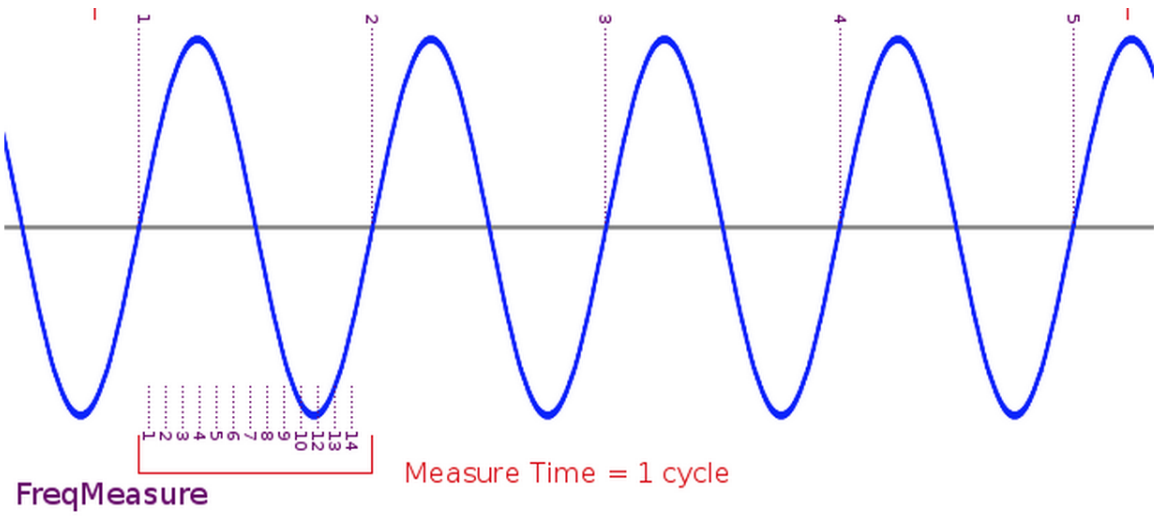
Introduction

In our team, I am responsible for using microcontroller to finish the signal processing and soldering the PCB.

This application note is focused on the signal processing portion of the project. After RF signal is received by the receive-end antenna, it goes through an amplifier, a mixer, and lastly an active low-pass filter. The output of the active low-pass filter will be fed into an ADC. For our quarter one design, we used the laptop computer's sound card as the ADC. Then we process the ADC digitized signals with matlab. In order to build an independent and portable Radar System and improve our radar system, our group decided to use the Teensy 3.1 board as the DSP device. Furthermore, the Teensy 3.1 has a proper ADC output pin, it can generate a 3.3V signal. Therefore, we used the Teensy 3.1 to generate the triangle wave as well.

The Method for Measuring Frequency

We used the excited library named FreqMeasure of Teensy 3.1 to processing the input signal. The library requires the input frequency as a digital level signal on the pin 3. The best interval for measure is within 0.1 Hz to 100 kHz. As the graph shows below, the Teensy 3.1 measures the elapsed time during a single cycle. This works well for relatively low frequencies, because a substantial time elapses. At higher frequencies, the short time can only be measured at the processor's clock speed, which results in limited resolution.



The basic commands of the Freqmeasure library

FreqMeasure.begin() -- Begin frequency measurement.
FreqMeasure.available() -- Returns the number of measurements available to read, or 0 if none are unread. If program spends time on other tasks and a relatively fast waveform is being measured, several readings may be available.

FreqMeasure.read() -- Read a measurement. An unsigned long (32 bits) containing the number of CPU clock cycles that elapsed during one cycle of the waveform. Each measurement begins immediately after the prior one without any delay, so several measurements may be averaged together for better resolution.

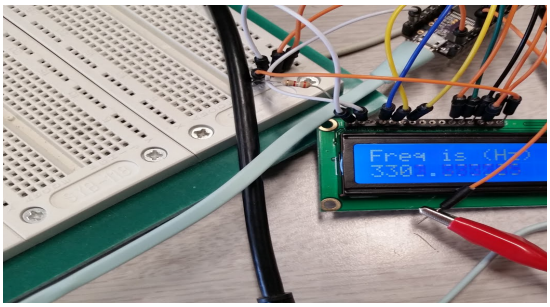
FreqMeasure.countToFrequency(count) -- Convert the 32 bit unsigned long numbers from read() to actual frequency.

FreqMeasure.end() -- Stop frequency measurement. PWM (analogWrite) functionality may be used again.

Analysis of the codes

The language of the Teensy 3.1 is similar to language C. In order to receive a reliable result, the variable “sum” is used to store thirty times of the frequencies which the Teensy measures. The average frequency of the thirty datas will be stored into the variable “frequency” variable. The while loop keeps this program run continuously. Once the teesy receive an input signal, it will record the time the signal can be detected in one cycle and store the answer into variable “sum”; and the times will be stored into variable “count”. Finally, we used the basic command “FreqMeasure.countToFrequency(sum / count) to evaluate the answer. Also, the command “ lcd.print(frequency,6)” can display a six decimal result on the LCD.

```
while (FreqMeasure.available()) {
  // average several reading together
  sum = sum + FreqMeasure.read();
  count = count + 1;
  if (count > 30) {
    float frequency = FreqMeasure.countToFrequency(sum / count);
    sum = 0;
    count = 0;
    lcd.setCursor(0,0);
    lcd.print("Freq is (Hz)");
    lcd.setCursor(0,1);
    lcd.print(frequency,6);
    delay(1000);
```



Serial_Output Example, With 3302 kHz

Based on our need, we decided to a sampling rate of 4ksps, which can detect object up to 10 meters away and with a resolution of 30Hz.

Requirements

At the end of each cycle, an interrupt routine runs. The actual capture of elapsed time is done by hardware, so some interrupt latency is acceptable. At relatively low frequencies, under 1 kHz, only minimal CPU time is used. However, as the frequency increases, the interrupt demands more CPU time. If interrupts are disabled for more than 1 cycle of the waveform, the measurement can span 2 or more cycles.