**RTDSP Lab2 Report**

**Group member: Jiabo Zhou, Hao Ding**

**Part 1. Quiz Answer**

|  |  |
| --- | --- |
| **Index of Y** | **Value** |
| 0 | 0.1376302 |
| 1 | 0.7976802 |
| 2 | 0.9904491 |
| 3 | 0.6030129 |
| 4 | -0.1376683 |
| 5 | -0.7977034 |
| 6 | -0.9904438 |
| 7 | -0.6029822 |

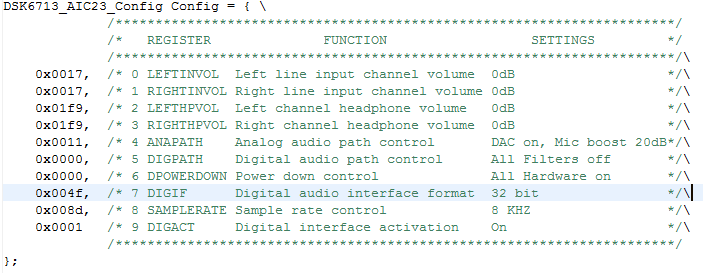
**Question1**

According to trace table above, there are eight samples per cycle. After 8 samples, the reading will wrap to the beginning of the trace table.

**Question2**

Since the sampling frequency is 8 kHz and sample number recorded per cycle is eight, the actual output frequency is 8000/8 = 1000 Hz.

**Question3**



According to reading from register 7 (0x004f),

**Part 2. Code explanation**

**float** sinegen(**void**)

{

/\*This code produces a fixed sine of 1KHZ (if the sampling frequency is 8KHZ)using a digital filter.

Using a look up table instead of a filter.\*/

// temporary variable used to output values from function

**float** wave;

**float** inteval;

//take sample from look up table

wave=table[(**int**)i];

//interval between each sample

inteval=SINE\_TABLE\_SIZE\*sine\_freq/sampling\_freq;

//increment sample number

i=i+inteval;

//keep i below table size and generate data continuously

**if**(i>=SINE\_TABLE\_SIZE){

i=i-SINE\_TABLE\_SIZE;

} `

**return**(wave);

}

**void** sine\_init(){

**int** i;

**for**(i=0;i<SINE\_TABLE\_SIZE;i++){

table[i]=sin(2\*PI\*i/SINE\_TABLE\_SIZE);

}

}

//I is previously defined as global integer

**float** i=0**;**

Without increasing SINE\_TABLE\_SIZE, we prefer to wrap around it. Once it exceeds its maximum value, we use an offset to maintain the value within its table size. Code is shown below:

**if**(i>=SINE\_TABLE\_SIZE){

i=i-SINE\_TABLE\_SIZE;

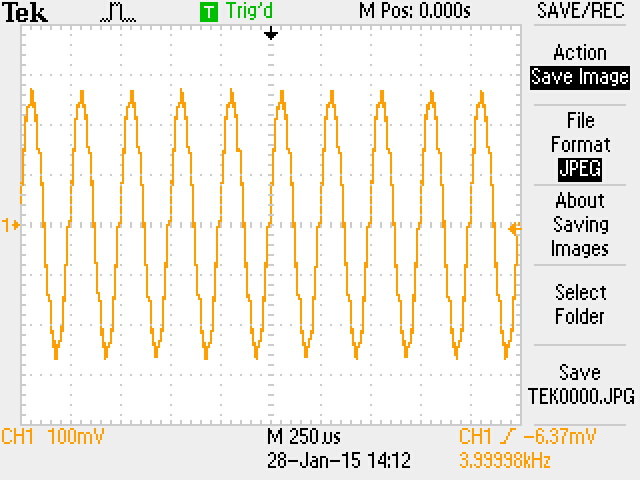
}

**Part 3. Scope trace and code performance**

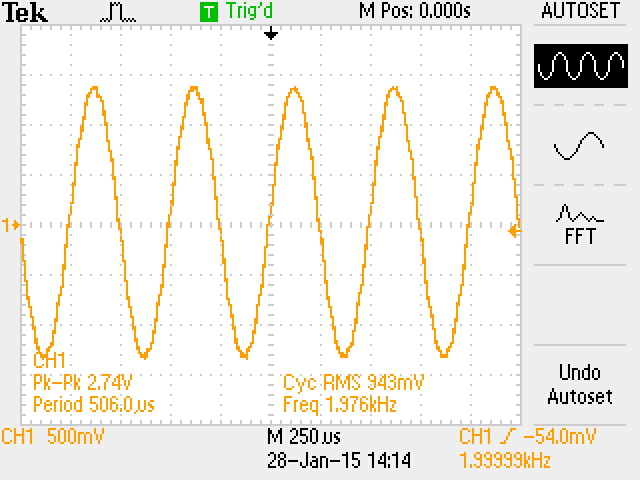
Our algorithm is able to show clear waveform from below 10 Hz to Nyquist Frequency. No distortion or harmonics for all applicable frequencies.

Waveforms at various frequencies are shown below:

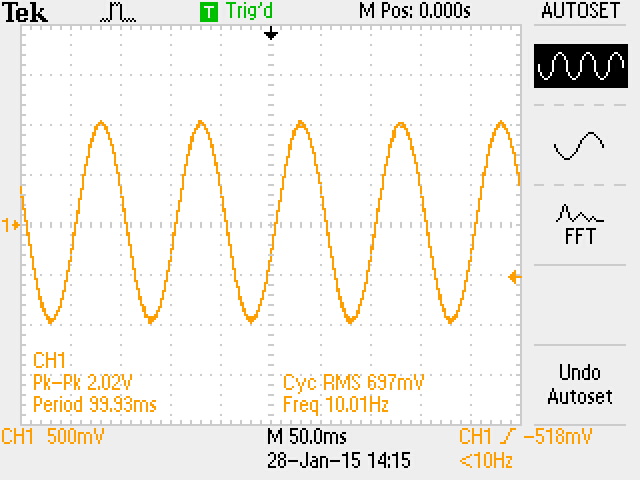
(Sampling Frequency: 8000 Hz)

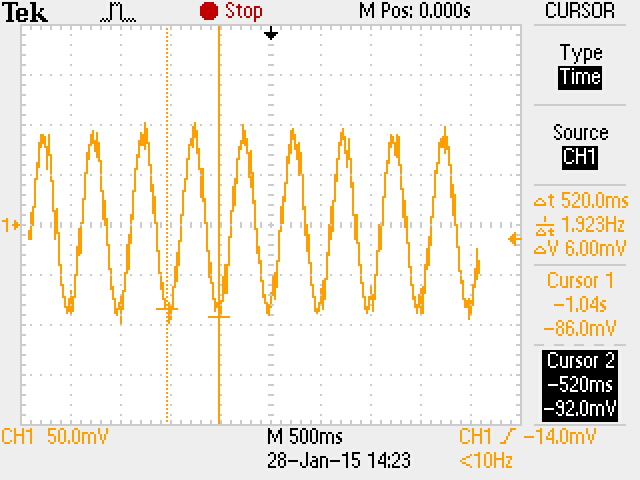
4000Hz:

2000Hz:



10 Hz:



2Hz (At a relatively low frequency, program runs well and a clear but attenuated waveform is shown):

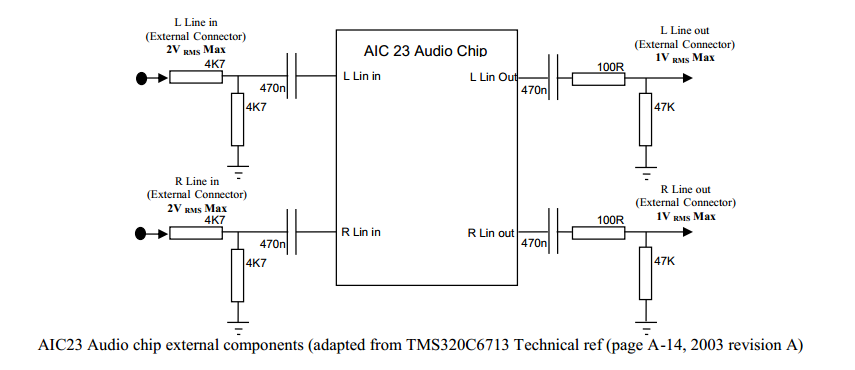
**Part 4. Limitation of boundary frequency**

Frequency is upper bounded by Nyquist Rate, above which aliasing happens:

Fmax = FS / 2

As shown in Part 3, sampling frequency is 8000 Hz, maximum frequency obtained after sampling is: 8000 / 2 = 4000 Hz.

Minimum frequency is determined by output port of AIC 23 Audio Chip.

Each output ports of audio chip is connected to a high-pass filter with components C = 470nf, R1 = 100 Ohm, R2 = 47 KOhm.

Corner frequency calculated is 7.1 Hz, below which signal is attenuated. Our observation in Part 3 has shown it.

**Part 5. Appendix**