EE 113D: Digital Signal Processing Design

Lab #3
External Input and Voice Transformations

Names: Erik Hodges & Ivan Manan

Due Date: October 24, 2018

Part 1: Manipulating and Analyzing Sine Waves

Objectives

The purpose of this experiment was to analyze the delay of the LCDK processor. Another objective of this experiment was to inject an additional programmed 3 ms delay.

Results

The first part of the experiment was to analyze the delay of LCDK as it outputs the original signal from the function generator. As evident in Figure 1.1, the output is identical to the input signal from the function generator, except that the output signal has a 5.28 ms delay.

Additionally, the next part of the experiment involved programming an additional 3 ms delay. Since the LCDK was sampling at 8000 samples/second, the math below was computed in order to determine that 24 additional samples were needed in order to program the additional 3 ms delay. As evident in Figure 1.2, the delay is now 8.24 ms.

$$(\frac{3}{1000}\ second) \bullet (8000\ \frac{samples}{second}) = 24\ samples.$$

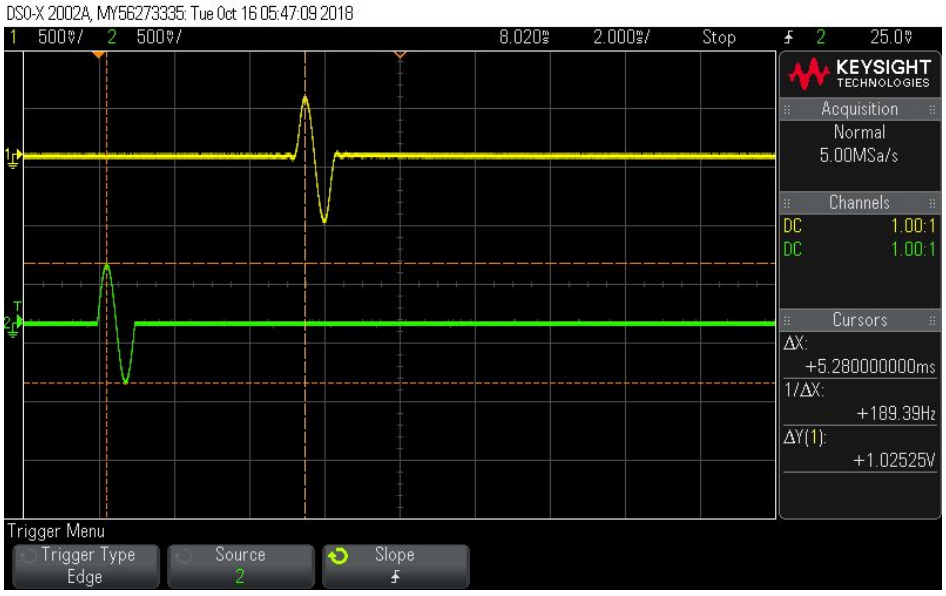

Figure 1.1 A Single Cycle of 1 kHz Sine Wave; The top waveform represents the output from the LCDK while the bottom waveform represents the signal from the function generator.

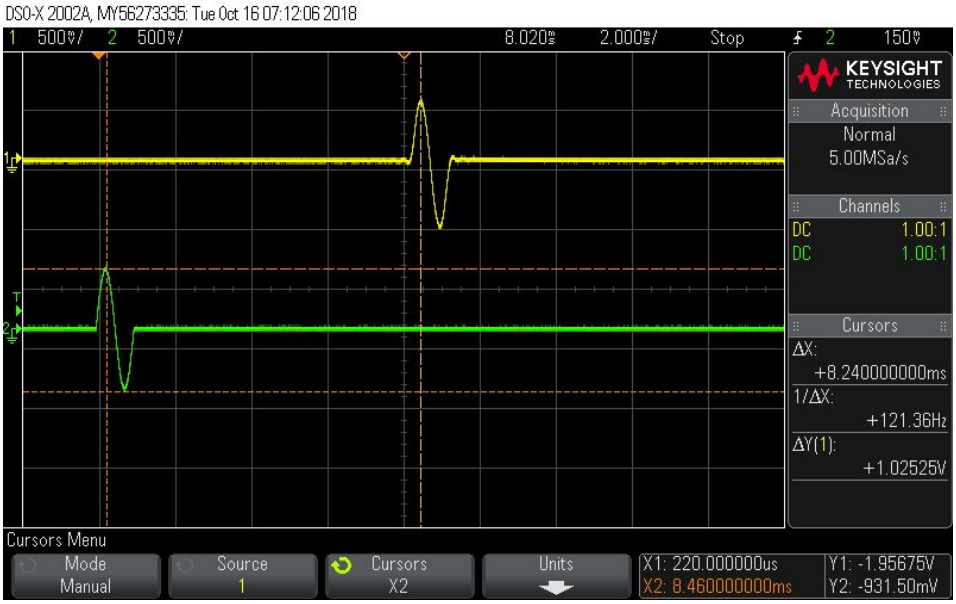

Figure 1.2 A Single Cycle of 1 kHz Sine Wave with a Programmed 3 ms Delay; The top waveform represents the output from the LCDK with a 3 ms programmed delay. The bottom waveform represents the signal from the function generator.

Code to Produce Figure 1.1

```
#include "L138_LCDK_aic3106_init.h"
#include "evmomapl138_gpio.h"

interrupt void interrupt4(void) // interrupt service routine
{
      int16_t left_sample;
      left_sample = input_left_sample();
      output_left_sample(left_sample);
      return;
}
int main(void)
{

L138_initialise_intr(FS_8000_HZ,ADC_GAIN_0DB,DAC_ATTEN_0DB,LCDK_LINE_INPUT);
      while (1);
}
```

Code to Produce Figure 1.2

```
#include "L138_LCDK_aic3106_init.h"
#include "evmomapl138_gpio.h"

int16_t FIFO[24];
int16_t FIFO_ctr = 0;

interrupt void interrupt4(void) // interrupt service routine
{
      output_left_sample(FIFO[FIFO_ctr]);
      FIFO[FIFO_ctr] = input_left_sample();
      ++FIFO_ctr;
      FIFO_ctr = FIFO_ctr % 24;
      return;
}
int main(void)
{
      int16_t i;
      for(i = 0; i < 24; i++){
            FIFO[i] = 0;
      }


L138_initialise_intr(FS_8000_HZ,ADC_GAIN_0DB,DAC_ATTEN_0DB,LCDK_LINE_INPUT);
      while (1);
}
```

Part 2: Manipulating Voice

Objectives

The purpose of this experiment was to create a program with the LCDK that manipulates voice input. The program involved storing the recorded sample into an array and processing the array in order to output the same recorded sample with either two times the original speed or half times the original speed.

Results

The experiment proved to be a success. The proof of demonstration signed by the TA is attached below. Moreover, the code to complete both the 2x and half times speed as well as the extra credit have been pasted below.

Every feature worked without much trouble. We tested the reverb feature by making a clicking sound and hearing two successive clicks played back. All the other features were tested by saying “Hello” or our names. However, there were some difficulties while developing this voice manipulation program. For instance, the microphone that we initially used turned out to be faulty, and the setup instantly started working when we replaced it. Additionally, slowing the voice input by half times speed was perplexing because the voice output was not clear when we spoke into the mic with a deep voice. This was not a software issue but simply a limitation of the speaker’s output frequency range or the frequency range of human hearing. When we spoke with a high voice, the playback was understandable. Additionally, slowing the program to 2/3 times speed was more understandable than half speed.

Scan of Proof of Demonstration

Name Erik Hudyes

Name Ivan Manan

Successfully completed Part 2 of the Week 3 lab:

Echo ✓

Twice Speed playback ✓

Half Speed Playback ✓

Successfully completed Extra Credit (1/3 slower playback) ✓

Comments Brava!

Signed Anastasios Papathanasopoulos

Code for Part 2 of the Lab

```
#include <stdint.h>
#include "L138_LCDK_aic3106_init.h"
#include "L138_LCDK_switch_led.h"
#include "evmomapl138_gpio.h"
#include <math.h>

int sw5;
int sw6;
int sw7;
int sw8;
const int one_sec = 8000; //number of cycles for 1 second
const int svty_ms = 560;  //70 ms delay = 560 samples delay
int ctr = 0;     // track time steps for each process
int LEDs_on = 0; //0 if all leds off, 1 if all leds on
int currentStep = 0; //which switch is engaged. =0 if no or multiple switches
int16_t recorded_samples[8000];
int16_t output = 0;
int n = 0;

int www = 0;

void all_on() {   //turn on all leds
      LCDK_LED_on(4);
      LCDK_LED_on(5);
      LCDK_LED_on(6);
      LCDK_LED_on(7);
      return;
}

void all_off() {   //turn off all leds
      LCDK_LED_off(4);
      LCDK_LED_off(5);
      LCDK_LED_off(6);
      LCDK_LED_off(7);
      return;
}

interrupt void interrupt4(void) // interrupt service routine
{
      sw5 = LCDK_SWITCH_state(5);
      sw6 = LCDK_SWITCH_state(6);
      sw7 = LCDK_SWITCH_state(7);
      sw8 = LCDK_SWITCH_state(8);

      www++;
      www = www % 64;


      // CASE: Check if more than one switch is ON
      if ((sw5 + sw6 + sw7 + sw8) > 1)
      {
```

```
            output_left_sample(0);

            if (ctr < one_sec/2){
                  ++ctr;
            } else { //flip all LED's after 4000 clock cycles
                  ctr = 0;
                  if (LEDs_on == 1){
                        all_off();
                        LEDs_on = 0;
                  } else {
                        all_on();
                        LEDs_on = 1;
                  }
            }
      }

      // CASE: Check only one switch is ON
      if ((sw5 + sw6 + sw7 + sw8) == 1) {

            if (sw5 == 1){ //switch 5 on

                  output_left_sample(0);

                  if (currentStep != 5){
                        ctr = 0;
                        all_off();
                        LEDs_on = 0;
                  }

                  currentStep = 5;
                  if(ctr == 0){ //turn on LED4 at start
                        LCDK_LED_on(4);
                        ++ctr;
                  }

                  if (((ctr % one_sec) != 0) && (ctr < 3*one_sec)) {
                        ++ctr;
                  }
                  else if (ctr == one_sec) { // switch to LED5
                        LCDK_LED_off(4);
                        LCDK_LED_on(5);
                        ++ctr;
                  }
                  else if (ctr == (2*one_sec)) { // switch to LED6
                        LCDK_LED_off(5);
                        LCDK_LED_on(6);
                        ++ctr;
                  }
                  else if ((ctr >= (3*one_sec)) && ctr < (4 * one_sec)) { //
switch to LED7
                        LCDK_LED_off(6);
                        LCDK_LED_on(7);
                        recorded_samples[ctr - (3 * one_sec)] =
```

```
input_left_sample();
                        ++ctr;
                  }
                  else if (ctr >= (4*one_sec)) { // turn off 7
                        LCDK_LED_off(7);
                        for (n = 0; n < 5; ++n){ // get rid of pop at start
of recording
                              recorded_samples[n] = 0;
                        }
                  }
            }

            if (sw6 == 1){ //switch 6 on

                  if (currentStep != 6){
                        ctr = 0;
                        all_off();
                        LEDs_on = 0;
                  }
                  currentStep = 6;

                  if (ctr < svty_ms){
                        output_left_sample(recorded_samples[ctr]);
                        ++ctr;
                  }
                  else if (ctr < one_sec){
                        output = recorded_samples[ctr] + 2 *
recorded_samples[ctr - svty_ms] / 3;
                        output_left_sample(output);
                        ++ctr;
                  }
                  else if (ctr < (one_sec + svty_ms)){
                        output =  2 * recorded_samples[ctr - svty_ms] / 3;
                        output_left_sample(output);
                        ++ctr;
                  }
                  else {
                        ctr = 0;
                        output_left_sample(0);
                  }

            }

            if (sw7 == 1) { // 2x speed
                  if (currentStep != 7){
                        ctr = 0;
                        all_off();
                        LEDs_on = 0;
                  }
                  currentStep = 7;

                  output_left_sample(recorded_samples[ctr]);
                  ctr += 2;
```

```
                  if (ctr >= one_sec){
                        ctr = 0;
                  }
            }

            if (sw8 == 1) { // half speed
                  if (currentStep != 8){
                        ctr = 0;
                        all_off();
                        LEDs_on = 0;
                  }
                  currentStep = 8;
                  //output each sample twice in a row
                        output_left_sample(recorded_samples[ctr >> 1]);
                  ++ctr;

                  if (ctr >= 2 * one_sec){
                        ctr = 0;
                  }
            }
      }

      // CASE: Otherwise, no switches are ON
      if ((sw5 + sw6 + sw7 + sw8) == 0) {
            ctr = 0;
            all_off();
            LEDs_on = 0;
            currentStep = 0;
            output_left_sample(0);
      }
      return;
}
int main(void)
{

L138_initialise_intr(FS_8000_HZ,ADC_GAIN_21DB,DAC_ATTEN_0DB,LCDK_MIC_INPUT);
      LCDK_GPIO_init();
      LCDK_SWITCH_init();
      LCDK_LED_init();

      // turn off LEDs
      LCDK_LED_off(4);
      LCDK_LED_off(5);
      LCDK_LED_off(6);
      LCDK_LED_off(7);

      while (1);
}
```

Extra Credit

```
#include <stdint.h>
#include "L138_LCDK_aic3106_init.h"
#include "L138_LCDK_switch_led.h"
#include "evmomapl138_gpio.h"
#include <math.h>



int sw5;
int sw6;
int sw7;
int sw8;
const int one_sec = 8000; //number of cycles for 1 second
const int svty_ms = 560;  //70 ms delay = 560 samples delay
int ctr = 0;     // track time steps for each process
int LEDs_on = 0; //0 if all leds off, 1 if all leds on
int currentStep = 0; //which switch is engaged. =0 if no or multiple switches
int16_t recorded_samples[8000];
int16_t output = 0;
int n = 0;
int sub_ctr = 0;

int www = 0;

void all_on() {   //turn on all leds
      LCDK_LED_on(4);
      LCDK_LED_on(5);
      LCDK_LED_on(6);
      LCDK_LED_on(7);
      return;
}

void all_off() {   //turn off all leds
      LCDK_LED_off(4);
      LCDK_LED_off(5);
      LCDK_LED_off(6);
      LCDK_LED_off(7);
      return;
}

interrupt void interrupt4(void) // interrupt service routine
{

      sw5 = LCDK_SWITCH_state(5);
      sw6 = LCDK_SWITCH_state(6);
      sw7 = LCDK_SWITCH_state(7);
      sw8 = LCDK_SWITCH_state(8);


      www++;
      www = www % 64;
```

```
// CASE: Check if more than one switch is ON
if ((sw5 + sw6 + sw7 + sw8) > 1)
{
	output_left_sample(0);

	if (ctr < one_sec/2){
		++ctr;
	} else { //flip all LED's after 4000 clock cycles
		ctr = 0;
		if (LEDs_on == 1){
			all_off();
			LEDs_on = 0;
		} else {
			all_on();
			LEDs_on = 1;
		}
	}
}

// CASE: Check only one switch is ON

if ((sw5 + sw6 + sw7 + sw8) == 1) {

	if (sw5 == 1){ //switch 5 on

		output_left_sample(0);

		if (currentStep != 5){
			ctr = 0;
			all_off();
			LEDs_on = 0;
		}

		currentStep = 5;
		if(ctr == 0){ //turn on LED4 at start
			LCDK_LED_on(4);
			++ctr;
		}

		if (((ctr % one_sec) != 0) && (ctr < 3*one_sec)) {
			++ctr;
		}
		else if (ctr == one_sec) { // switch to LED5
			LCDK_LED_off(4);
			LCDK_LED_on(5);
			++ctr;
		}
		else if (ctr == (2*one_sec)) { // switch to LED6
			LCDK_LED_off(5);
			LCDK_LED_on(6);
			++ctr;
		}
		else if ((ctr >= (3*one_sec)) && ctr < (4 * one_sec)) { //
```

```
switch to LED7
                        LCDK_LED_off(6);
                        LCDK_LED_on(7);
                        recorded_samples[ctr - (3 * one_sec)] =
input_left_sample();
                        ++ctr;
                  }
                  else if (ctr >= (4*one_sec)) { // turn off 7
                        LCDK_LED_off(7);
                        for (n = 0; n < 5; ++n){ // get rid of pop at start
of recording
                              recorded_samples[n] = 0;
                        }
                  }
            }

            if (sw6 == 1){ //switch 6 on

                  if (currentStep != 6){
                        ctr = 0;
                        all_off();
                        LEDs_on = 0;
                  }
                  currentStep = 6;

                  if (ctr < svty_ms){
                        output_left_sample(recorded_samples[ctr]);
                        ++ctr;
                  }
                  else if (ctr < one_sec){
                        output = recorded_samples[ctr] + 2 *
recorded_samples[ctr - svty_ms] / 3;
                        output_left_sample(output);
                        ++ctr;
                  }
                  else if (ctr < (one_sec + svty_ms)){
                        output =  2 * recorded_samples[ctr - svty_ms] / 3;
                        output_left_sample(output);
                        ++ctr;
                  }
                  else {
                        ctr = 0;
                        output_left_sample(0);
                  }

            }

            if (sw7 == 1) { // 4/3x speed
                  if (currentStep != 7){
                        ctr = 0;
                        sub_ctr = 0;
                        all_off();
                        LEDs_on = 0;
```

```
                }
                currentStep = 7;

                if (sub_ctr < 2){
                      output_left_sample(recorded_samples[ctr]);
                      ++ctr;
                      ++sub_ctr;
                } else {
                      output_left_sample(recorded_samples[ctr]);
                      ctr += 2;
                      sub_ctr = 0;
                }


                if (ctr >= one_sec){
                      ctr = 0;
                }
          }

          if (sw8 == 1) { // 2/3x speed
                if (currentStep != 8){
                      ctr = 0;
                      sub_ctr = 0;
                      all_off();
                      LEDs_on = 0;
                }
                currentStep = 8;

                //output each sample twice in a row
                if (sub_ctr < 2){
                      output_left_sample(recorded_samples[ctr]);
                      ++ctr;
                      ++sub_ctr;
                } else {
                      output_left_sample(recorded_samples[ctr]);
                      sub_ctr = 0;
                }


                if (ctr >= one_sec){
                      ctr = 0;
                }
          }
    }

    // CASE: Otherwise, no switches are ON
    if ((sw5 + sw6 + sw7 + sw8) == 0) {
          ctr = 0;
          all_off();
          LEDs_on = 0;
          currentStep = 0;
          output_left_sample(0);
    }
```

```
      return;
}
int main(void)
{

L138_initialise_intr(FS_8000_HZ,ADC_GAIN_21DB,DAC_ATTEN_0DB,LCDK_MIC_INPUT);
      LCDK_GPIO_init();
      LCDK_SWITCH_init();
      LCDK_LED_init();

      // turn off LEDs
      LCDK_LED_off(4);
      LCDK_LED_off(5);
      LCDK_LED_off(6);
      LCDK_LED_off(7);


      while (1);
}
```