EENG 3910: Project V – Digital Signal Processing System Design
Assignment 4
Due 29 February 2016
Nathan Ruprecht
UNT – Electrical Engineering

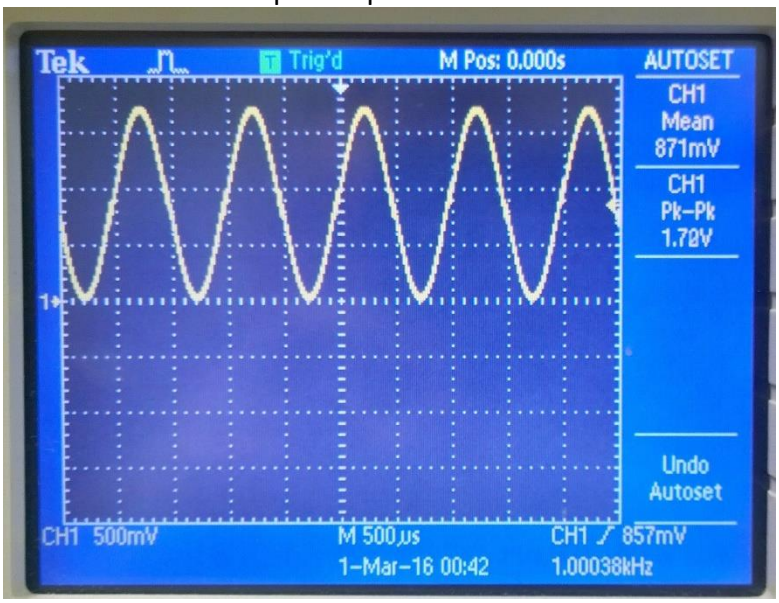
Introduction

The purpose of assignment 4 was to use SSI/SPI along with ADC/DAC. We were able to see the effects of different sample frequencies on the signal by using an oscilloscope as well as a LPF to clean it up.

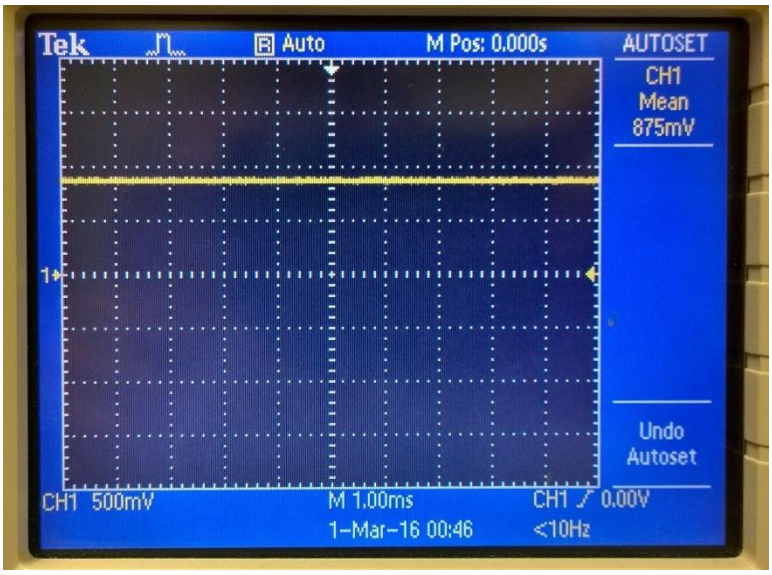
Results and Discussion

Problem 1: Generate a single-tone sine signal from LaunchPad with a SPI DAC chip

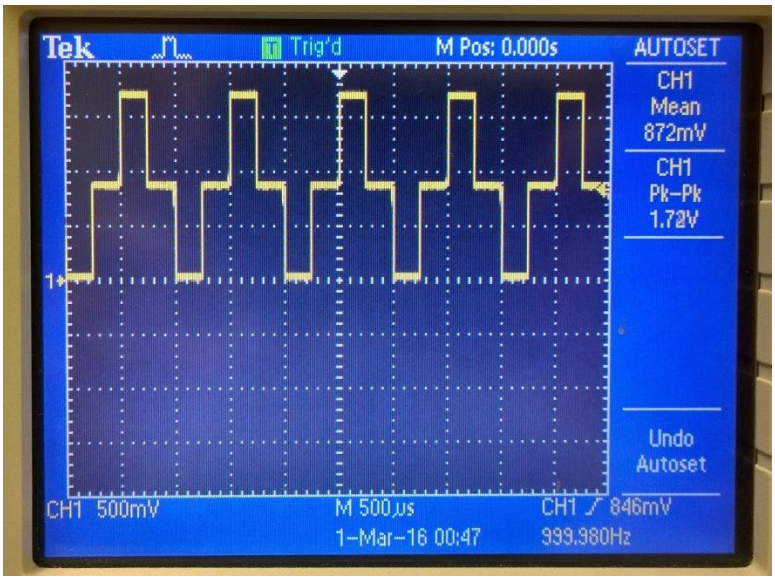
The code for problem 1 was given to us. The first set of pictures are to show the code running as is at different sample frequencies.



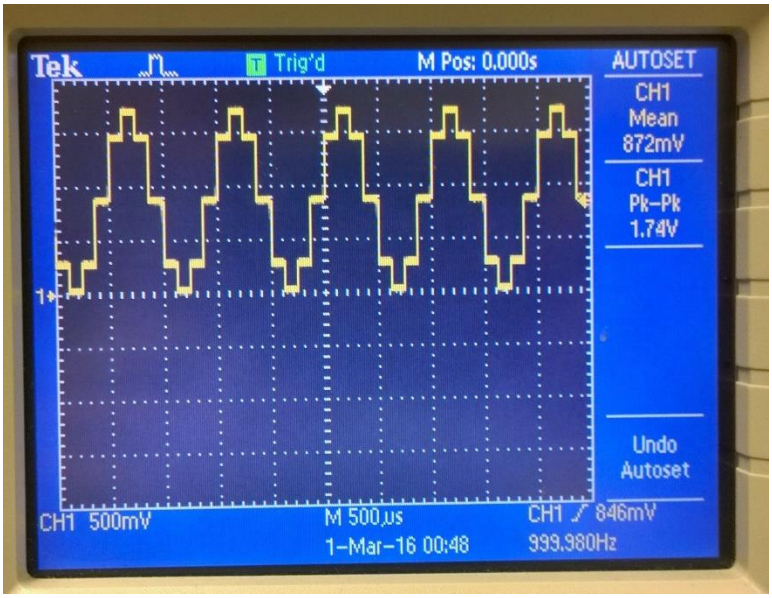
Output with SAMP_FREQ at 48kHz



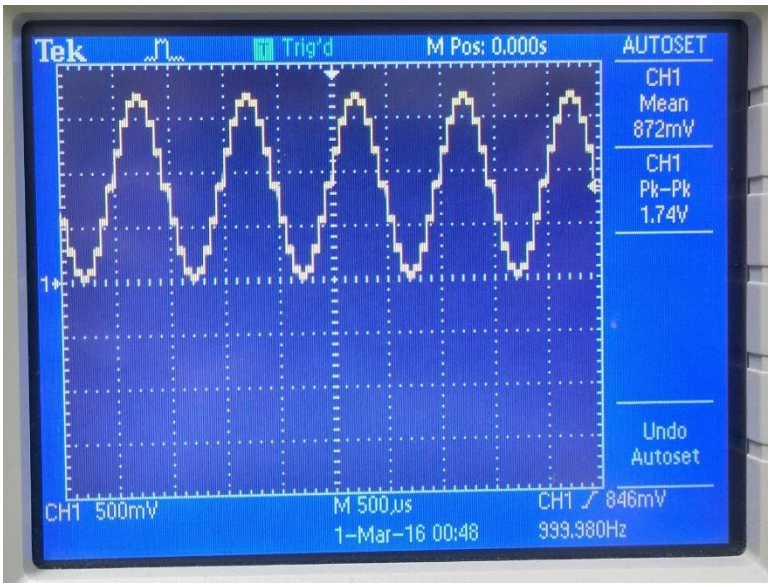
Output with SAMP_FREQ at 2 kHz



Output with SAMP_FREQ at 4 kHz

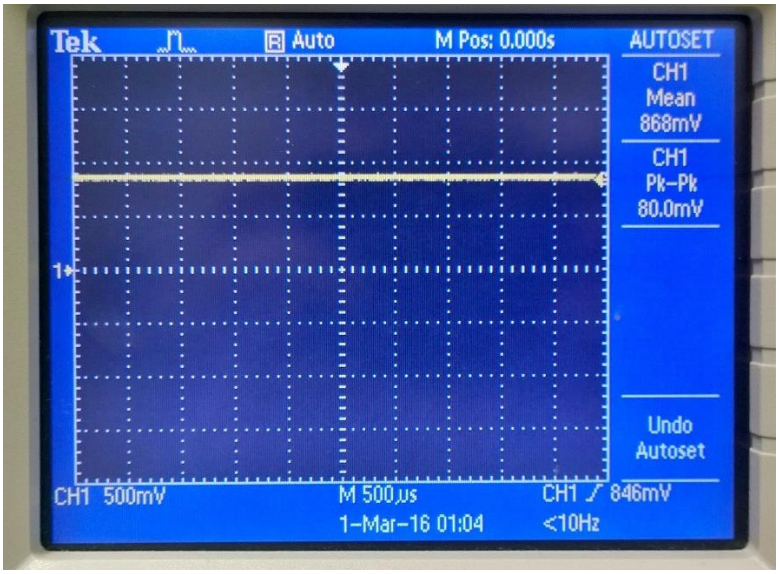


Output with SAMP_FREQ at 8 kHz

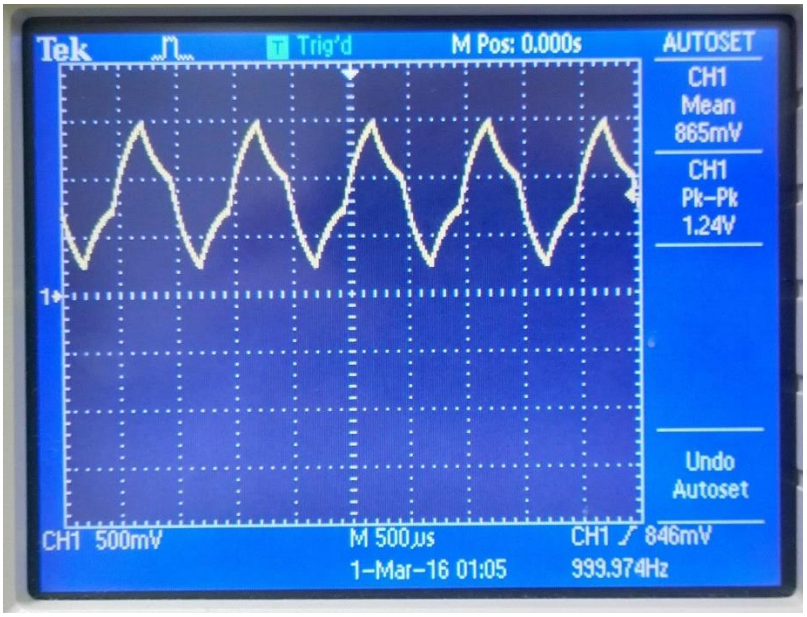


Output with SAMP_FREQ at 16 kHz

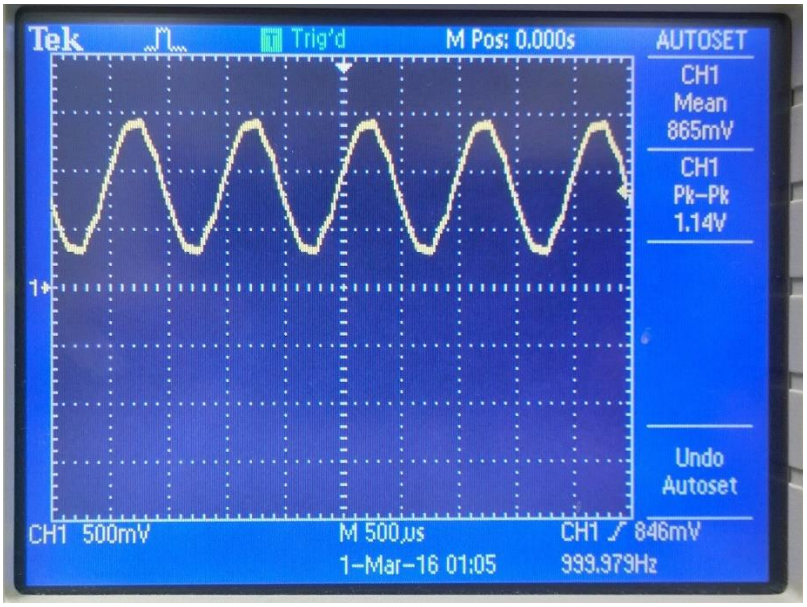
Using the same code and same sample frequencies, the second part of problem 1 was to send that same signal through a low pass filter to smooth it out. In problem 1, the signal is a simulated sinusoid that goes through the ADC which is why it shows steps. Sending it through a circuit with a capacitor cleans it up to look more like a smooth sine wave.



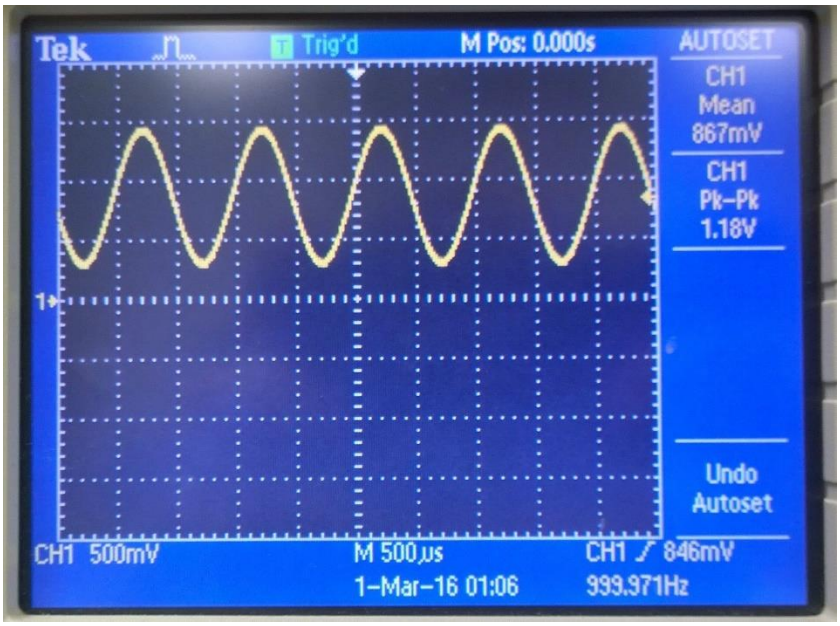
Output of LPF with SAMP_FREQ at 2 kHz



Output of LPF with SAMP_FREQ at 4 kHz



Output of LPF with SAMP_FREQ at 8 kHz



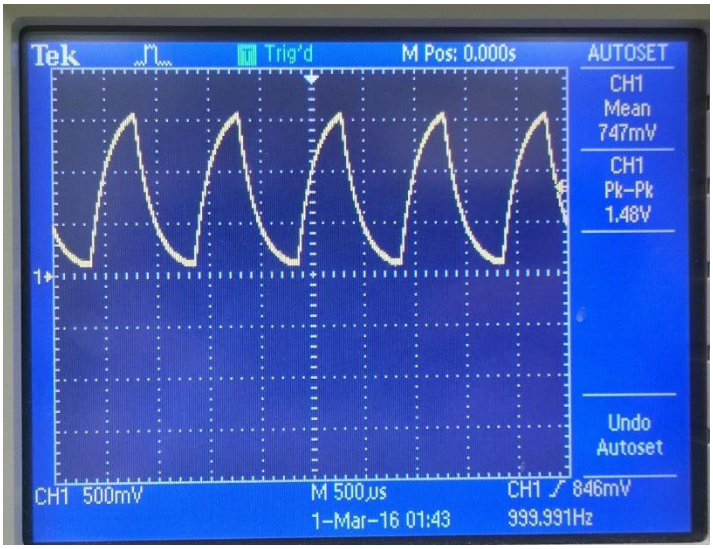
Output of LPF with SAMP_FREQ at 16 kHz

Problem 2: Signal pass through with ADC and DAC

In problem 2, now we have an actual signal instead of the code generating one. The same idea applies where we need to sample the signal are certain intervals. How that shows in code is making an array of signal length, then sampling that signal and storing those values into the array. I referenced my Lab3_5 to make sure I set up the pins and ADC the same way. The completely new is the if-else statement in the Timer1 ISR to deal with the signal which is just used from the Lab4_1 generated signal.

```
while(!ADCIntStatus(ADC0_BASE, ADC0_SEQ_NUM, false)) {
}

ADCIntClear(ADC0_BASE, ADC0_SEQ_NUM);
ADCSequenceDataGet(ADC0_BASE, ADC0_SEQ_NUM, &sig[x]);
if(x < SIG_LEN-1){
	sig[x] = SPI_CTRL_MASK | (SPI_DATA_MASK & sig[x]);
	x++;
}else{
	x=0;
}
```



Output of Lab4_2 with SAMP_FREQ at 16 kHz

Problem 3: Signal pass through with ADC, DAC and UART user command
I just took problem 2 and added in two conditions in the switch statement (like the problem specified). So now the "s" starts timer1 while "p" stops it.

```
case 'S':
case 's':
      //Start timer1
      TimerEnable(TIMER1_BASE, TIMER_A);
      break;
case 'P':
case 'p':
      //Stop timer1
      TimerDisable(TIMER1_BASE, TIMER_A);
      break;
```

Summary and Conclusion

The part of the assignment that gave me the greatest difficulty was understanding how the signal was being stored in an array. Or how the code was sampling and saving into the array. The actual code was a matter of rearranging what was given to us in Lab4_1 so it was less of knowing how to code, but more of knowing what the intent is and how to make it happen. Since everyone knows what a sine wave looks like, 4_1 was good to actually see the signal after it goes through the ADC with varying sampling frequencies. So if we took an infinite amount of sampling, we'd have the analog signal. So the challenge is to take enough samples and smooth it out to get as close to the analog signal as possible.

Modified Source Code:

Changes to get Lab4_2

```
//initialization of ADC
void init_adc(void)
{
	SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOD); //enable port D
	GPIOPinTypeADC(GPIO_PORTD_BASE, GPIO_PIN_0); //Sets PD0 as ADC0 input

	SysCtlPeripheralEnable(SYSCTL_PERIPH_ADC0);   //enable ADC0
	ADCSequenceConfigure(ADC0_BASE, ADC0_SEQ_NUM, ADC_TRIGGER_PROCESSOR, 0);
	ADCSequenceStepConfigure(ADC0_BASE, ADC0_SEQ_NUM, 0,
ADC_CTL_IE|ADC_CTL_CH7|ADC_CTL_END);
	ADCSequenceEnable(ADC0_BASE, ADC0_SEQ_NUM);


// Timer1 interrupt service routine
void Timer1_ISR(void)
{
	// Clear the timer interrupt.
	TimerIntClear(TIMER1_BASE, TIMER_TIMA_TIMEOUT);

	//ADC Read
	ADCProcessorTrigger(ADC0_BASE, ADC0_SEQ_NUM);

	while(!ADCIntStatus(ADC0_BASE, ADC0_SEQ_NUM, false)) {
	}

	ADCIntClear(ADC0_BASE, ADC0_SEQ_NUM);
	ADCSequenceDataGet(ADC0_BASE, ADC0_SEQ_NUM, &sig[x]);
	if(x < SIG_LEN-1){
		sig[x] = SPI_CTRL_MASK | (SPI_DATA_MASK & sig[x]);
		x++;
	}else{
		x=0;
	}

	// Send data to SPI.
	SSIDataPut(SSI0_BASE, sig[sig_index]);

	if(sig_index < SIG_LEN-1)
		sig_index++;
	else
		sig_index = 0;

}
```

Changes to get Lab4_3

```
const char *disp_text[NUM_DISP_TEXT_LINE] = {
		"\n",
		"UART and LED Demo\n",
		"H: help, R: red, G: green, B: blue. S: start sampling.  P: pause
sampling. \n",
```

```
"> " };


case 'S':
case 's':
      //Start timer1
      TimerEnable(TIMER1_BASE, TIMER_A);
      break;
case 'P':
case 'p':
      //Stop timer1
      TimerDisable(TIMER1_BASE, TIMER_A);
      break;
```