ESM Lab Grading Report Template

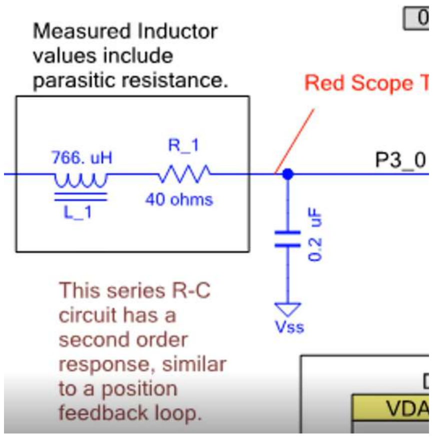
Course Number: 5
Module: Lab 6 on PID Control
Lab Report Date: 13/12/2018

Student Name(s): **Rushi James Macwan and Poorn Mehta**

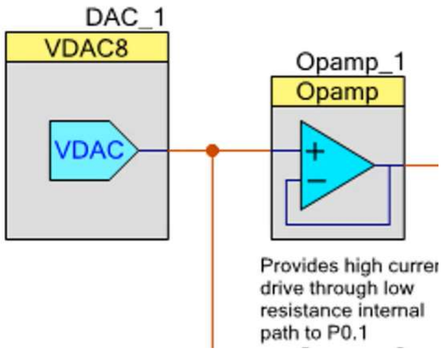
Each Section of this Lab Report Counts for 20 points. Points will be allocated as follows:
20 points: section fully meets requirements of this rubric
15 points: section mostly meets requirements of this rubric
10 points: section meets roughly half the requirements of this rubric
5 points: section does not meet requirements, but shows a weak attempt
0 points: section blank

Goal: The purpose of this lab is to learn how to construct a simple PID control system to control an LC oscillating circuit. We chose this system because it has the same 2nd order response as found in many control topologies such as robot arms, but our components are much more affordable. We will use our small DC motor from earlier labs as an inductor with resistor, and then add a 4.7 microfarad capacitor in series. All other components we will use are found inside the PSOC system.

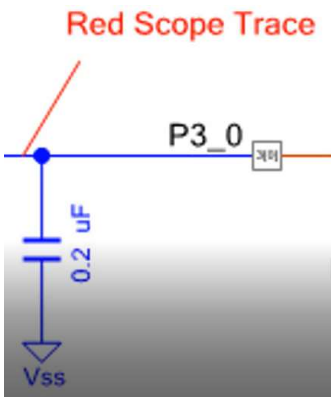
Background: In a common motion control topology a microprocessor writes a number to a circuit, and circuit is designed to force the motor current to be proportional to that number. Because torque is proportional to current, the number is proportional to acceleration times rotary inertia ($T = J\ dx^2/dt^2$). Acceleration of course, is the 2nd derivative of position, and velocity is the first derivative. Mathematically, the 2nd order equations for position control are identical to those of a series L-R-C circuit. This snippet from our schematic is for the L-R-C circuit.


Measured Inductor values include parasitic resistance.
Red Scope T
766. uH
L_1
R_1
40 ohms
P3_0
0.2 uF
Vss
This series R-C circuit has a second order response, similar to a position feedback loop.
VDA

In our circuit the input and control will come from a DAC_1, per this snippet below of the schematic.


DAC_1
VDAC8
VDAC
Opamp_1
Opamp
Provides high curre drive through low resistance internal path to P0.1

The system output be the voltage across the capacitor, as shown in this snippet of the schematic, and per the red color on our scope traces.



We will use the successive approximation (ADC SAR ADC) set to its fastest conversion speed to minimize the phase delay. Software reads the voltage from the capacitor, and calculates the proportional, derivative and integral terms, adds a feedforward team, and outputs that number to DAC_1. We will test our circuit by measuring the response of our system to a step input.

We will use all four of the PSoC high current op amps in parallel to get enough current to run this circuit.

We will show you how to tune a PID system to get a good step response, as well as how to write the code to prevent saturation and overload. We will use scope traces and our USB UART built into PSoC to send data to the Teraterm program on your PC. There is a detailed explanation in the video for how to write the code. It is important that you watch the whole video to give you guidance in code writing.

(A) Functional demonstration of your circuit to our TA. In this exercise, you schedule an appointment with your TA to show that your hardware functions as designed. For the Closed Loop C Motor Control lab, this will involve the following steps:

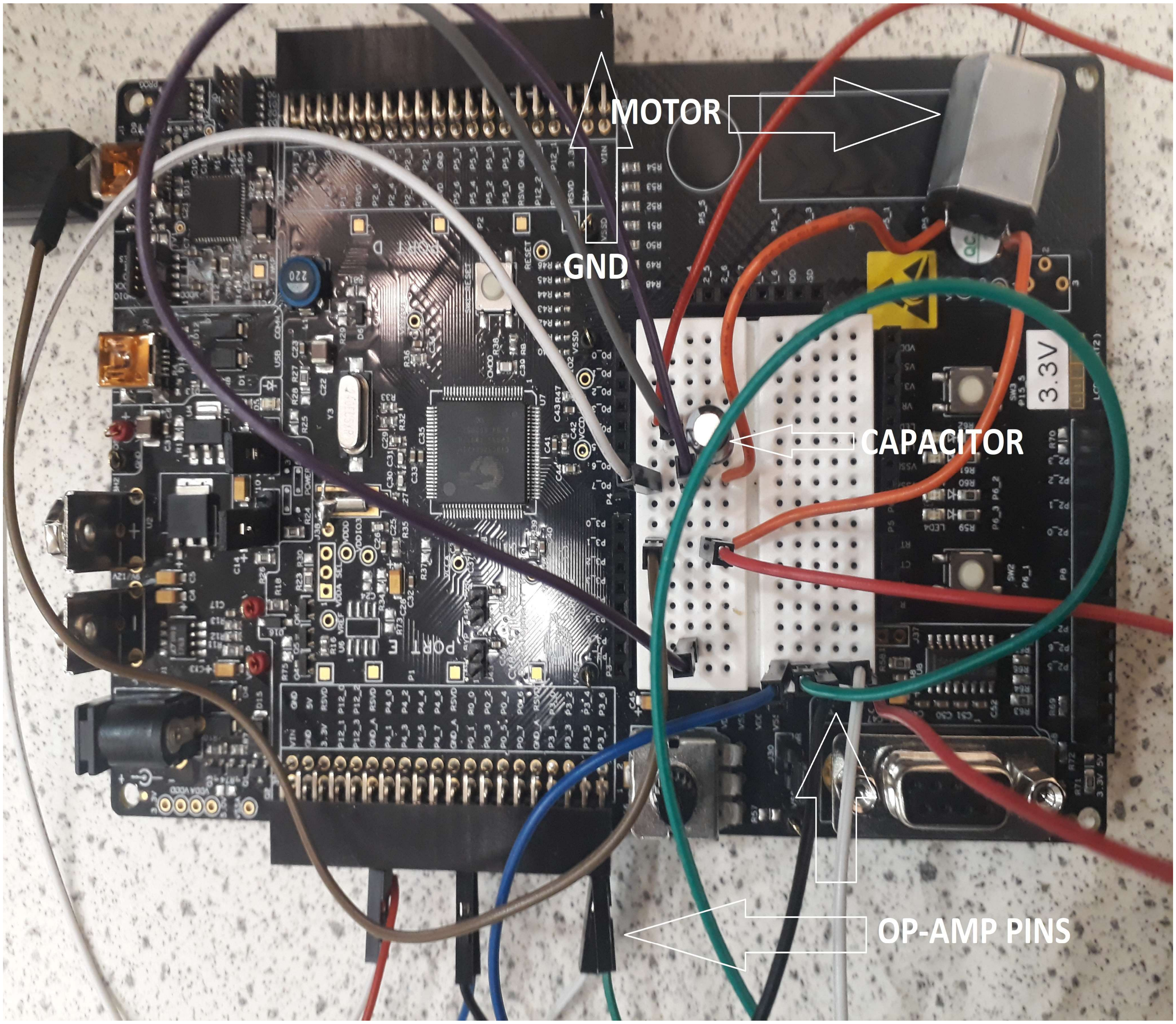
1. Show that all hardware is in place, and that your PSoC software can control the motor current.
2. Show the scope traces of your current flow, along with the desired current setting from the DAC.

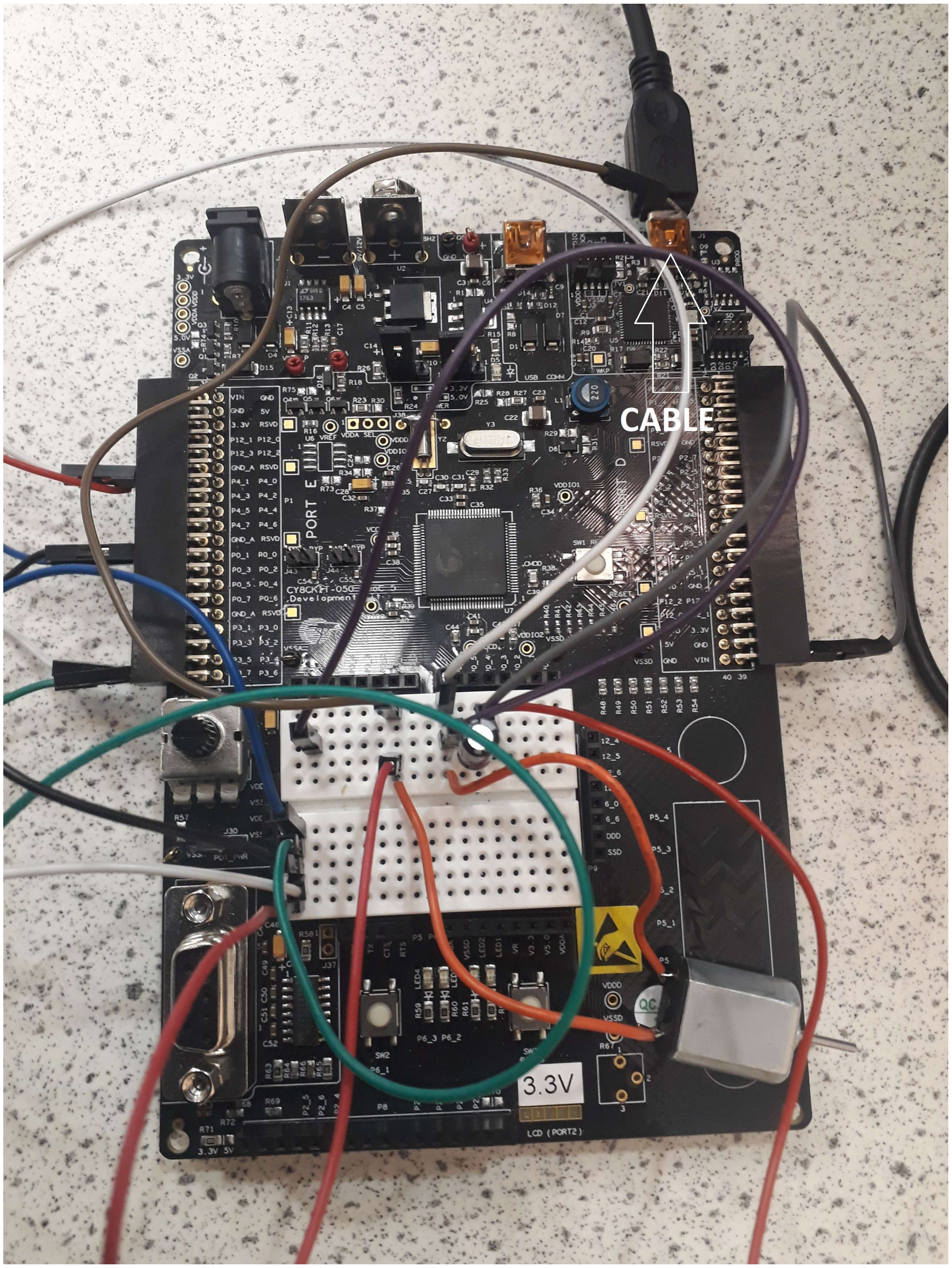
If you are an on-campus student, then show your circuit to one of our TA's during office hours.

If you are a distance learning student, make an online appointment with your TA to demonstrate your work via Zoom meeting or other Web-based meeting tool. You can use the camera on your laptop PC or suitable plug-in webcam (Logitech etc.) to demonstrate a working circuit.

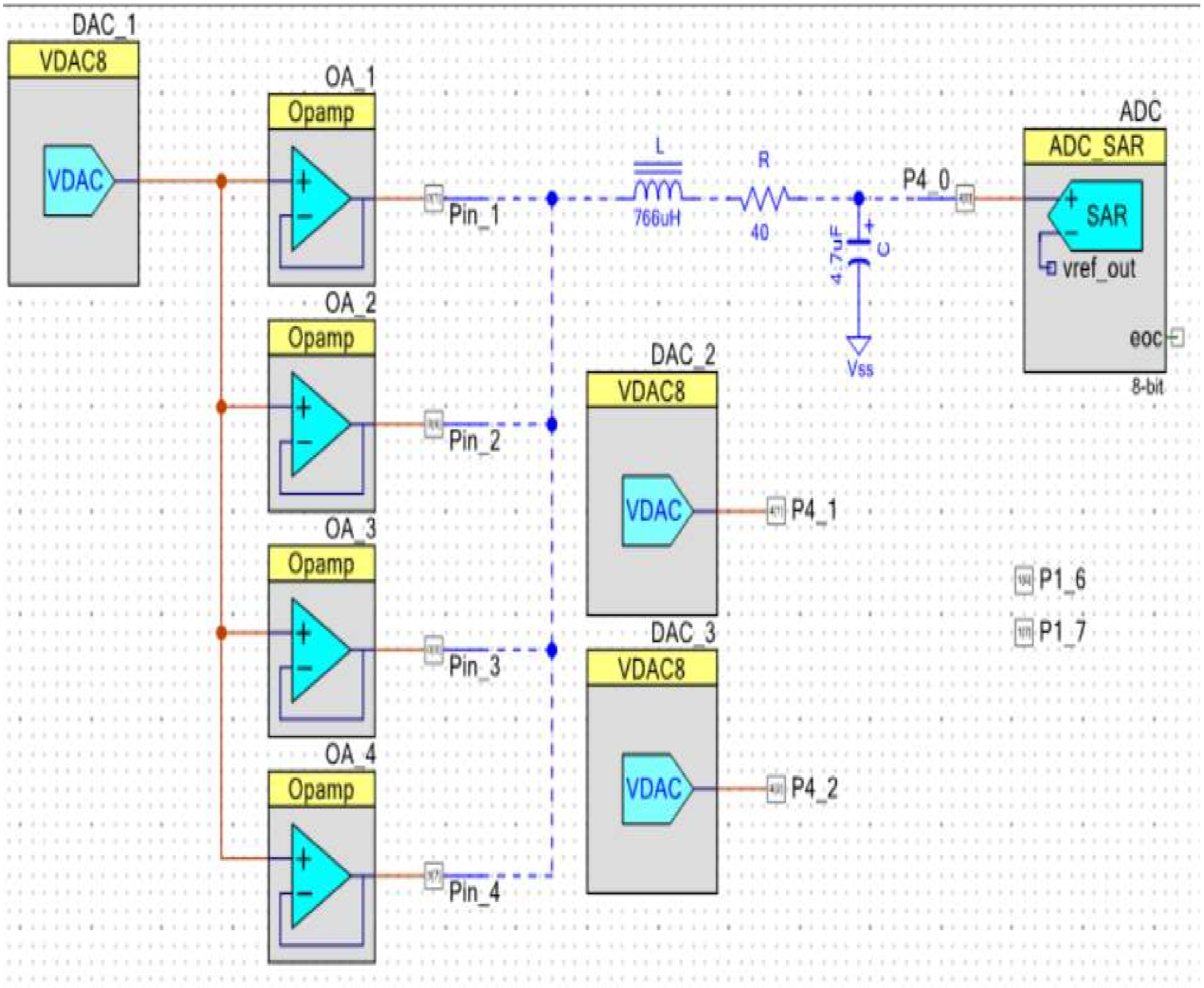
(B) Place photos here of your hardware setup, including PSoC board, connections to Oscilloscope or nScope, wiring, LCD Display, components, etc.
Label all components.

Sample photo is shown here. (Make sure to delete the sample and place your own photos here).


MOTOR
GND
CAPACITOR
OP-AMP PINS
3.3V


CABLE

(C) Place complete PSoC schematic here. This schematic must include internal components from the PSoC board (amplifier, ADC, etc.), as well as external components (power diode, motor, n-channel power FET, etc.).



(D) Complete PSoC software. This software must include calls to all internal functions, appropriate comments, and functional code that you included. We will not grade you on the exact syntax and structure, as there are numerous ways to structure the code and still provide the temperature measurement function. Instead, we will grade you on the completeness of the code relative to using the appropriate PSoC functions to gather the necessary data.

```
/* ========================================
 * ECEN 5053-003 ESA LAB 6 (PID Control)
 *
 * Created by: Rushi James Macwan and
 *             Poorn Mehta
 *
 * FALL 2018 - 12/08/2018
 *
 * Copyright YOUR COMPANY, THE YEAR
 * All Rights Reserved
 * UNPUBLISHED, LICENSED SOFTWARE.
 *
 * CONFIDENTIAL AND PROPRIETARY INFORMATION
 * WHICH IS THE PROPERTY OF your company.
 *
 * ========================================
*/

#include "project.h"
#include "stdio.h"
#include "stdlib.h"
#include "string.h"

#define SAMPLES 200

#define p 150
#define d 100
#define i 800

#define ref 100

/*................Global Variables......................*/

int8 Vmeas_raw;

uint8 dac_in;
uint8 x;

int32 Vmeas, Vmeas_last, Vmeas_delta, Verror, Vmeas_buffer[SAMPLES];
int32 Verror_sum, Verror_sum_limited = 1000;
int32 Kd_scale_factor = 150, Kp_scale_factor = 100, Ki_scale_factor =
800;
int32 Kd_term, Kp_term, Ki_term;
int32 Vout;
int32 Kp = 1000, Kd = 3000, Ki = 1000;
```

```
uint32 loop_count;

/*................Functions................*/

void func_1(int32 Vref, uint32 Z, int32 FF_term); // Vref --> reference
value; Z --> scaling factor (to evade the voltage tripping); FF_term --
> base factor

/*................MAIN....................*/

int main(void)
{
    CyGlobalIntEnable;
    ADC_Start();
    DAC_1_Start(); DAC_2_Start(); DAC_3_Start();
    OA_1_Start(); OA_2_Start(); OA_3_Start(); OA_4_Start();

      /*.......while loop........*/

    while(1)
    {
        int32 Vref = 140; int32 FF_term = 110; uint32 Z = 90;
        func_1(Vref, Z, FF_term);

        // Decreasing Vref by 20 counts; FF_term by 10 counts; Scaling
factor by 10 counts

        Vref = 120; FF_term = 100; Z = 80;
        func_1(Vref, Z, FF_term);

        // Decreasing Vref by 20 counts; FF_term by 10 counts; Scaling
factor by 10 counts

        Vref = 100; FF_term = 90; Z = 70;
        func_1(Vref, Z, FF_term);
    }
}

void func_1(int32 Vref,  uint32 Z, int32 FF_term)
{
    int count = 0;
    while(1)
    {
        count++;

        if(count >= 1000)
        {
            CyDelay(5);
            break;
        }

        DAC_2_SetValue(Z); // scaled Vref with appropriate display gain
        DAC_3_SetValue(dac_in); // Vout measured with proper 0-255
scaling without any errors

        Vmeas_last = Vmeas;

        ADC_StartConvert();
        ADC_IsEndConversion(ADC_WAIT_FOR_RESULT);
        Vmeas_raw = ADC_GetResult8();
        Vmeas = (uint8)Vmeas_raw;
        Vmeas_buffer[loop_count] = Vmeas;

            /**********Kd term************/

            if(loop_count == 1)
            {
                Vmeas_delta = 0;
            }
            else
            {
                Vmeas_delta = Vmeas - Vmeas_last;
            }

            Kd_term = Kd * Vmeas_delta / Kd_scale_factor;
            if(Kd_term > d) {Kd_term = d;}
            if(Kd_term < -d) {Kd_term = -d;}

            /**********Kd term************/
            /*****************************/
            /*****************************/
            /*****************************/
            /**********Kp term************/

            Verror = Vref - Vmeas;
            Kp_term = Kp * Verror / Kp_scale_factor;
            if(Kp_term > p) {Kp_term = p;}
            if(Kp_term < -p) {Kp_term = -p;}

            /**********Kp term************/
            /*****************************/
            /*****************************/
            /*****************************/
            /**********Ki term************/

            Verror_sum += Verror;
            if(Verror_sum > Verror_sum_limited) {Verror_sum =
Verror_sum_limited;}
            if(Verror_sum < -Verror_sum_limited) {Verror_sum = -
Verror_sum_limited;}
            Ki_term = Ki * Verror_sum / Ki_scale_factor;
            if(Ki_term > i) {Ki_term = i;}
            if(Ki_term < -i) {Ki_term = -i;}

            /**********Ki term************/
            /*****************************/
            /*****************************/
            /*****************************/
            /***********PID term**********/

            Vout = FF_term + Kd_term + Kp_term + Ki_term;

            /***********PID term**********/

        if(Vout > 230)
        {
            dac_in = 230;
            }
            if(Vout < 0)
            {
                dac_in = 0;
            }
            else
            {
                dac_in = Vout;
            }
            x = (uint8)(2*dac_in + 30)/1.25; // with proper PID scaling
            DAC_1_SetValue(x);
        }
        return;
}
```

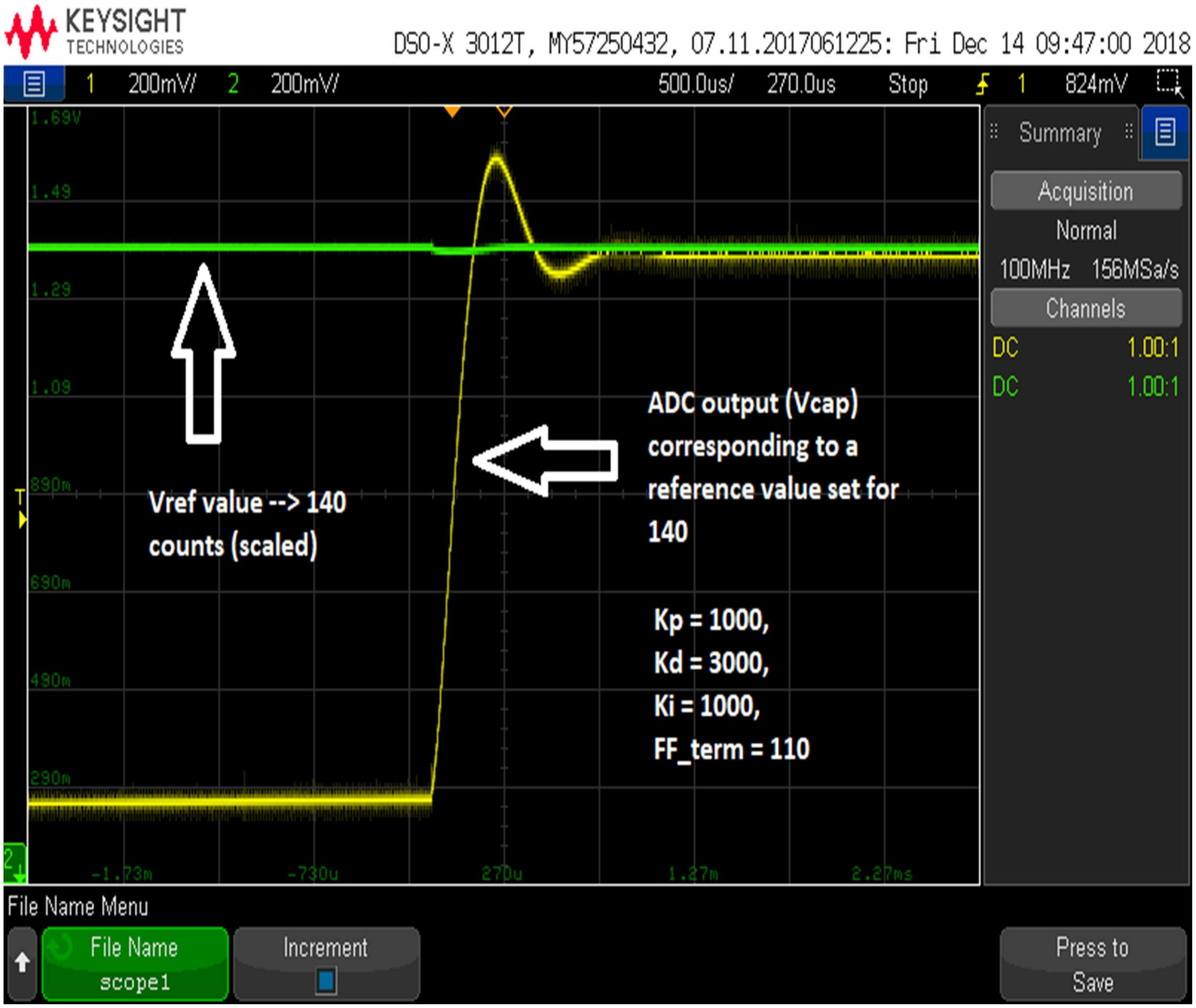
E) Place screenshots from your oscilloscope or nScope showing the output, the voltage of the capacitor, and the input DAC_1.

Vary your values for Kp, Ki, Kd, and FF_term to get Vref = 120 +/-1, a stable input value for DAC_1, and no more than 1.5 oscillations of the output before it settles at its final value. Use your Teraterm program to view Vref.

You will not get the exact values shown in our video because your motor inductance and resistance, as well as the exact value of your 4.7 microfarad capacitor will vary from what we used in our system. Your PSoC board may have varying resistances. Your code will likely execute at a different clock cycle than ours, as your goal is to make it *functionally* equivalent, *not* the same line by line. So, you will need to do your own experiment to optimize the values.

Sample screenshots are shown here. (Make sure to delete the sample and place your own screen shots here).

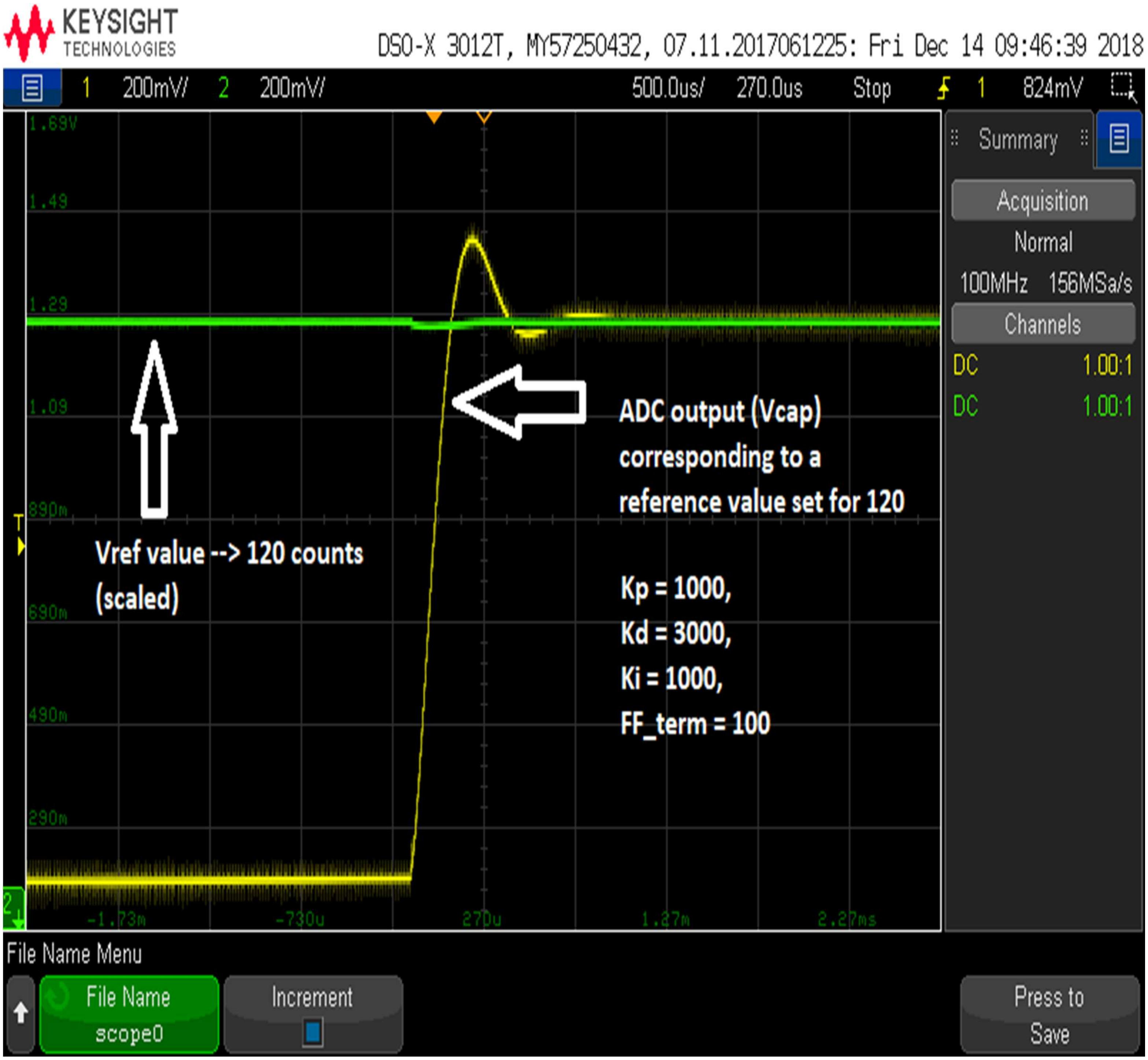
Note: The yellow dotted trace below is for Vref.



Vref value --> 140 counts (scaled)

ADC output (Vcap) corresponding to a reference value set for 140

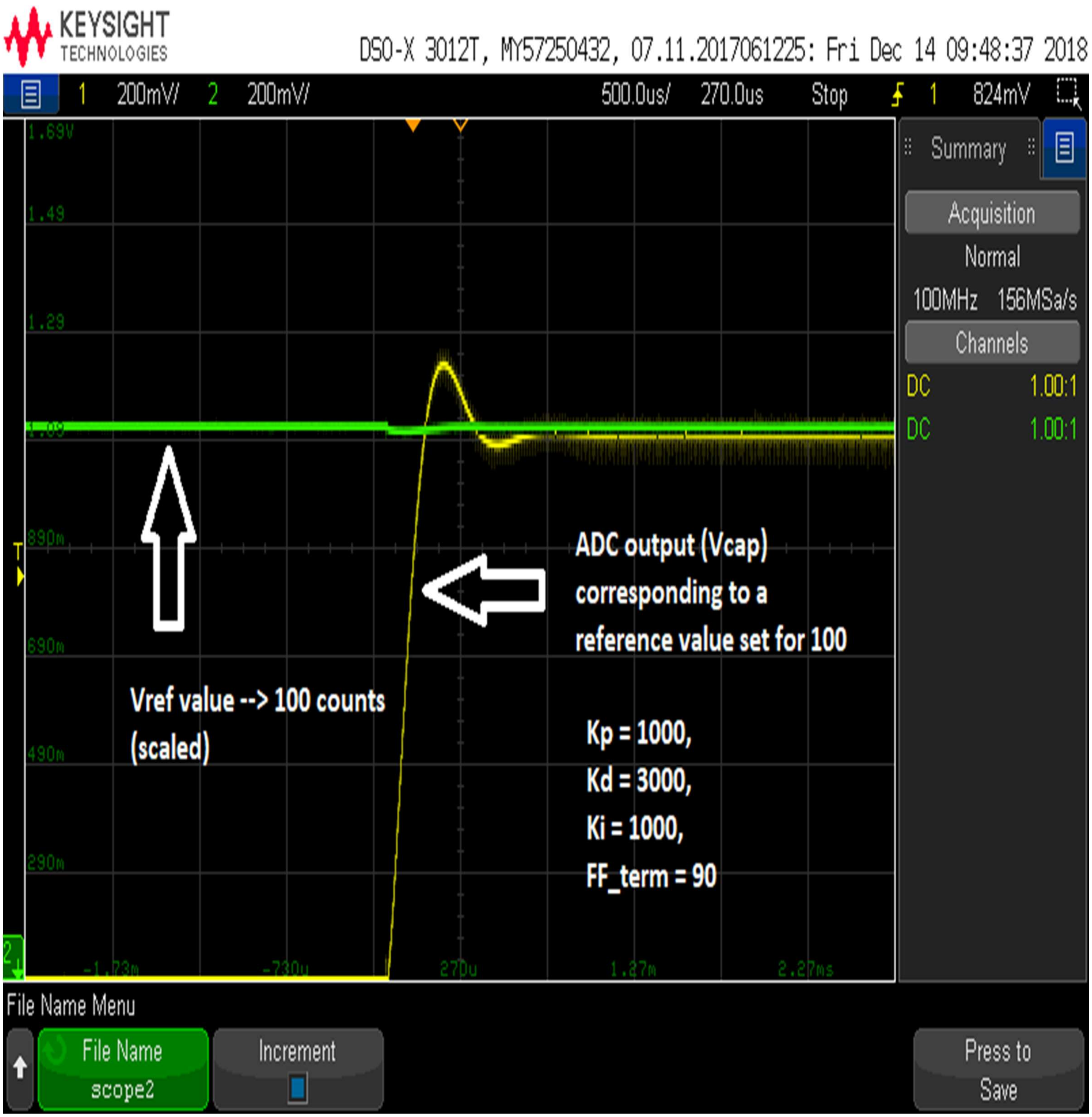
Kp = 1000,
Kd = 3000,
Ki = 1000,
FF_term = 110



Vref value --> 120 counts (scaled)

ADC output (Vcap) corresponding to a reference value set for 120

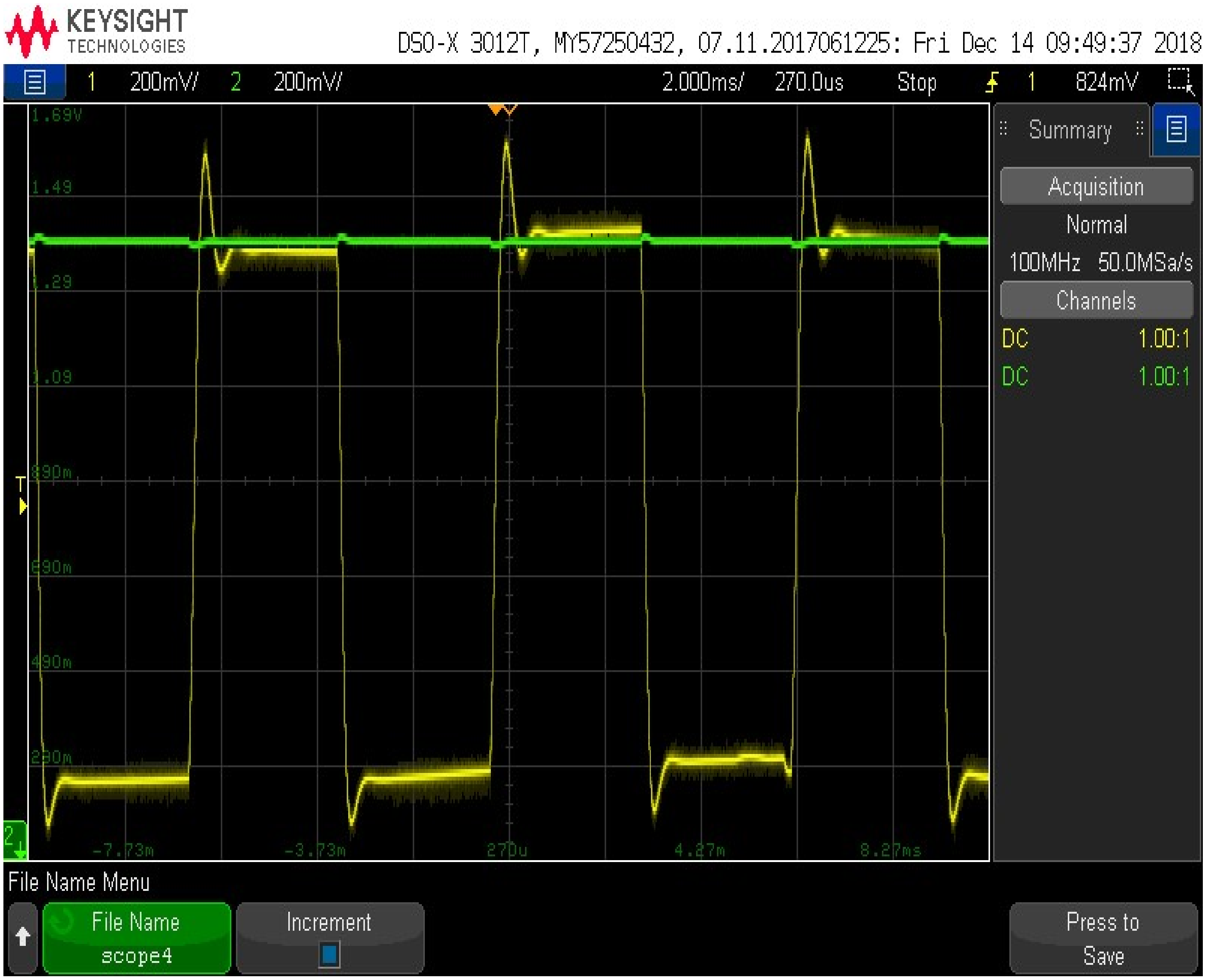
Kp = 1000,
Kd = 3000,
Ki = 1000,
FF_term = 100



ADC output (Vcap) corresponding to a reference value set for 100

Vref value --> 100 counts (scaled)

Kp = 1000,
Kd = 3000,
Ki = 1000,
FF_term = 90



KEYSIGHT TECHNOLOGIES

DSO-X 3012T, MY57250432, 07.11.2017061225: Fri Dec 14 09:49:37 2018

1 200mV/ 2 200mV/ 2.000ms/ 270.0us Stop 1 824mV

Summary

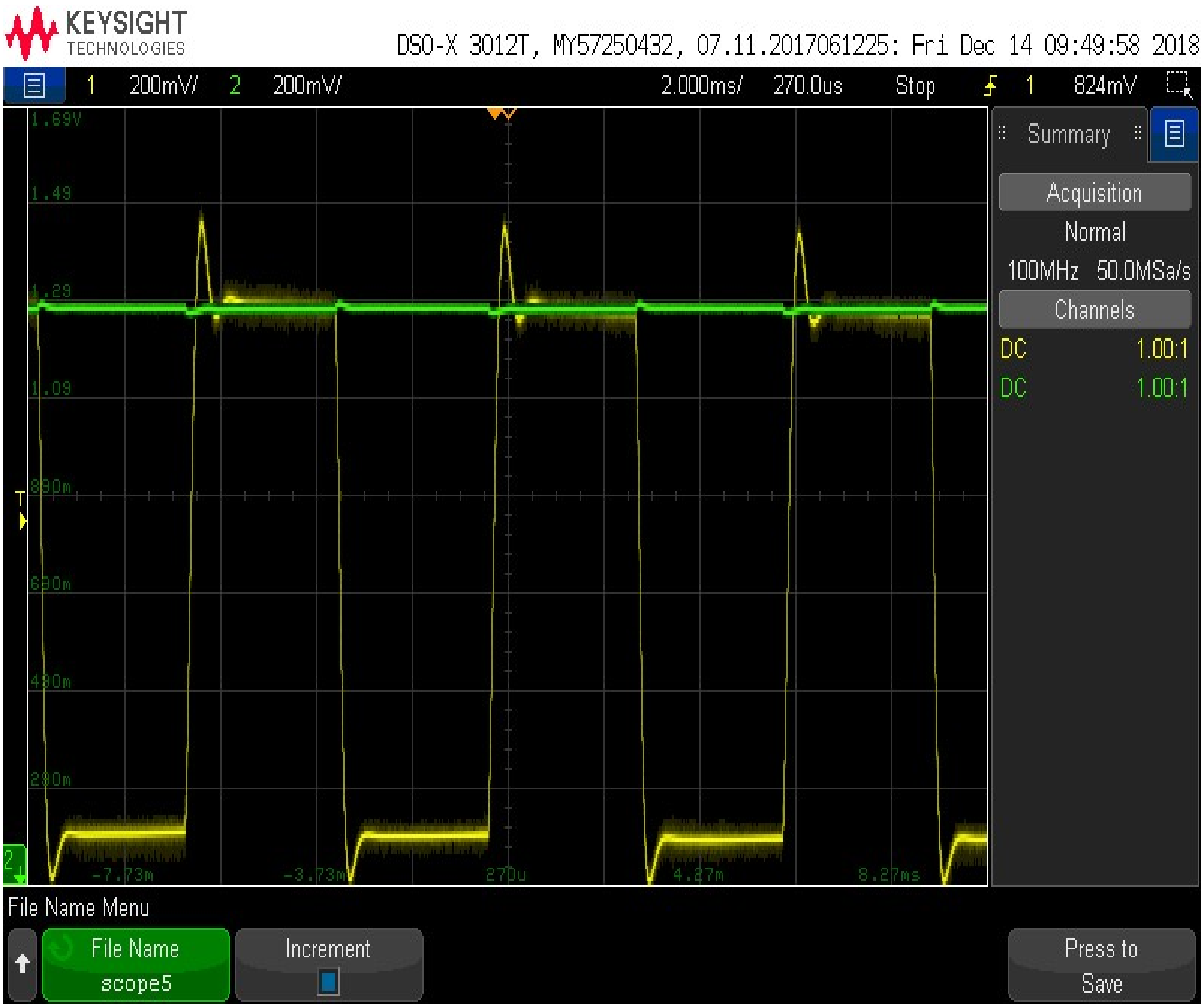
Acquisition
Normal
100MHz 50.0MSa/s

Channels

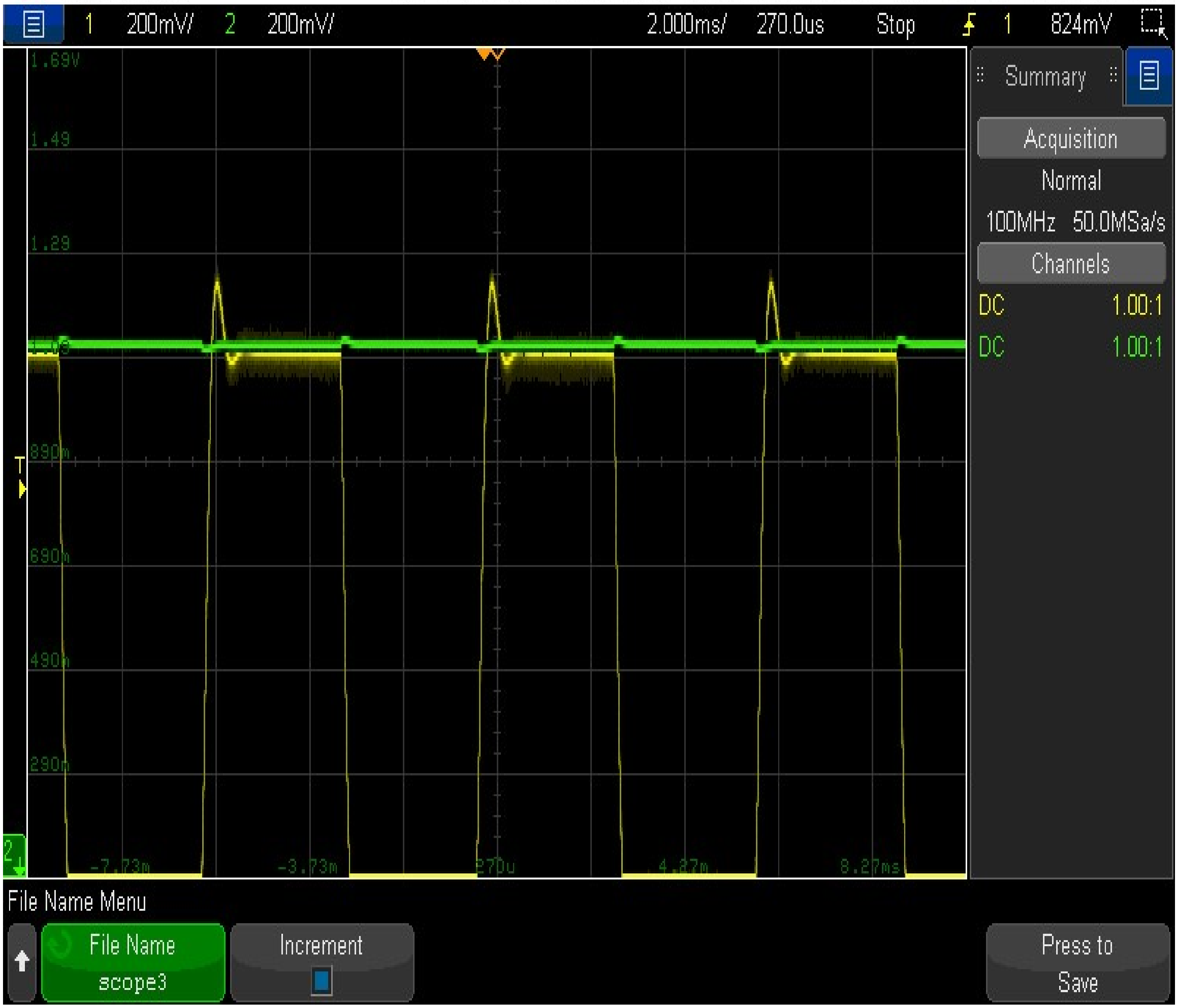
DC	1.00:1
DC	1.00:1

File Name Menu

File Name scope4 | Increment | Press to Save


KEYSIGHT TECHNOLOGIES
DSO-X 3012T, MY57250432, 07.11.2017061225: Fri Dec 14 09:49:58 2018
1 200mV/ 2 200mV/ 2.000ms/ 270.0us Stop 1 824mV
1.69V
1.49
1.29
1.09
890m
690m
490m
290m
-7.73m
-3.73m
270u
4.27m
8.27ms
Summary
Acquisition
Normal
100MHz 50.0MSa/s
Channels
DC 1.00:1
DC 1.00:1
File Name Menu
File Name
scope5
Increment
Press to Save

DSO-X 3012T, MY57250432, 07.11.2017061225: Fri Dec 14 09:49:31 2018



1 200mV/ 2 200mV/ 2.000ms/ 270.0us Stop 1 824mV

Acquisition: Normal, 100MHz 50.0MSa/s

Channels: DC 1.00:1, DC 1.00:1