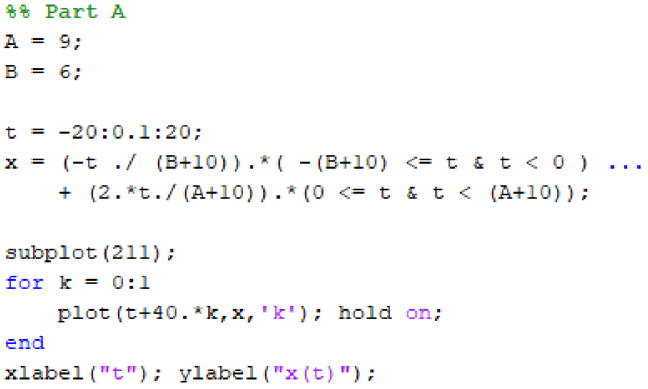
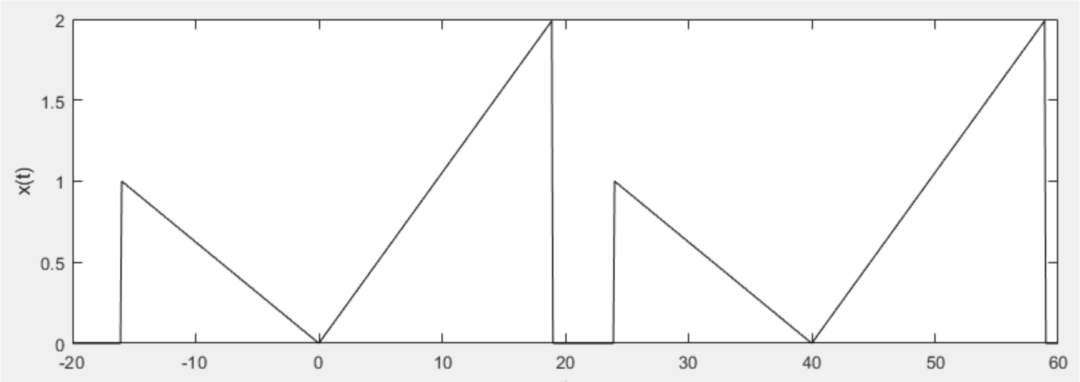
5a) Student ID = 1148496

In MATLAB,



```
%% Part A
A = 9;
B = 6;

t = -20:0.1:20;
x = (-t ./ (B+10)).*( -(B+10) <= t & t < 0 ) ...
    + (2.*t./(A+10)).*(0 <= t & t < (A+10));

subplot(211);
for k = 0:1
    plot(t+40.*k,x,'k'); hold on;
end
xlabel("t"); ylabel("x(t)");
```



5b) From book on page 43 the "Gibbs phenomenon" is explained. Where because of jump discontinuities (which we have), the Fourier series will overshoot the top by 9% for any order K of the series.

So, we must give 9% tolerance on the top and bottom. If we just scale our current x(t) so that it has a range from 0V→3.3V then we will overshoot to a range of -0.297→3.597V, which is not possible for our hardware to achieve. So, we must scale x(t) so it is in the range of 0.272→3.028.

This will give us two equations with two unknowns. One for our top max voltage, and one for our bottom min voltage:

$$eq1) \qquad (T-B)*1.09 \leq 4095$$

$$eq2) \qquad B = 4095 - T$$

Then,

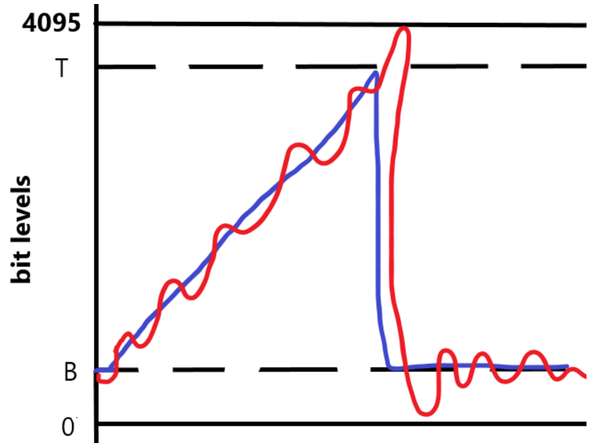
$$(T - 4095 + T)*1.09 \leq 4095$$

$$2T - 4095 \leq \frac{4095}{1.09}$$

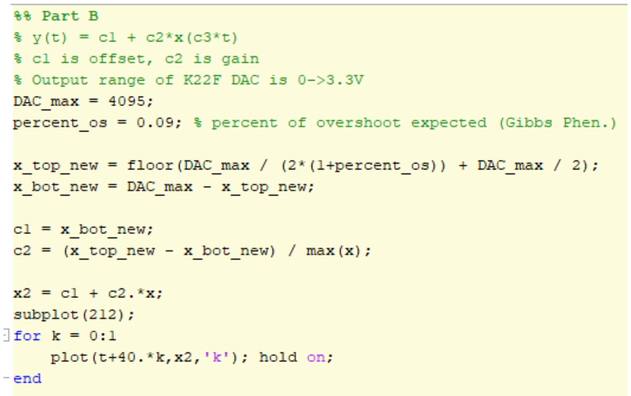
$$T \leq \frac{4095}{2.18} + \frac{4095}{2} = 3925.9 \rightarrow 3925$$

$$B = 170$$

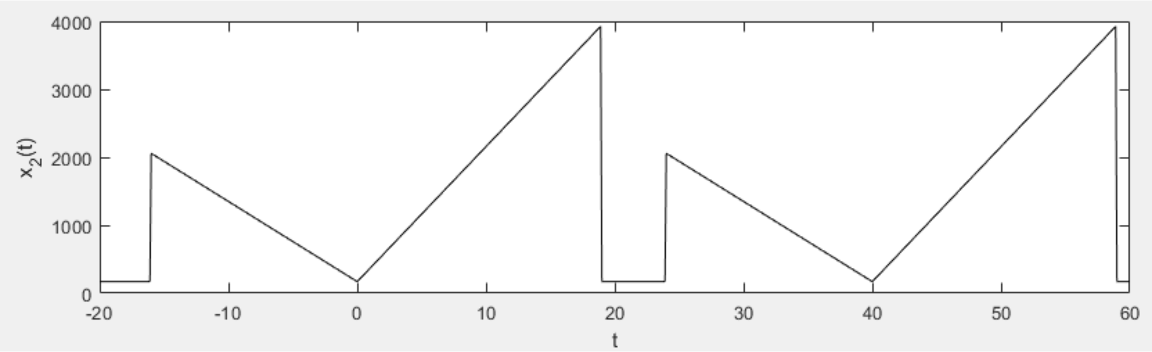
$$c_1 = B = 170 \qquad c_2 = \frac{3925-170}{\max(x(t))} = 1877.5$$



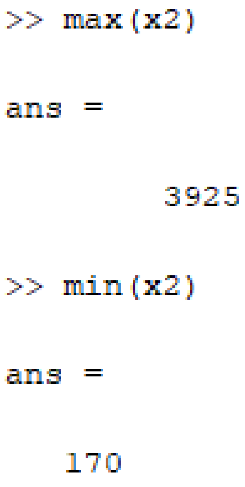
In MATLAB,



```
%% Part B
% y(t) = c1 + c2*x(c3*t)
% c1 is offset, c2 is gain
% Output range of K22F DAC is 0->3.3V
DAC_max = 4095;
percent_os = 0.09; % percent of overshoot expected (Gibbs Phen.)

x_top_new = floor(DAC_max / (2*(1+percent_os)) + DAC_max / 2);
x_bot_new = DAC_max - x_top_new;

c1 = x_bot_new;
c2 = (x_top_new - x_bot_new) / max(x);

x2 = c1 + c2.*x;
subplot(212);
for k = 0:1
    plot(t+40.*k,x2,'k'); hold on;
end
```



Where now:



```
>> max(x2)

ans =

        3925

>> min(x2)

ans =

   170
```

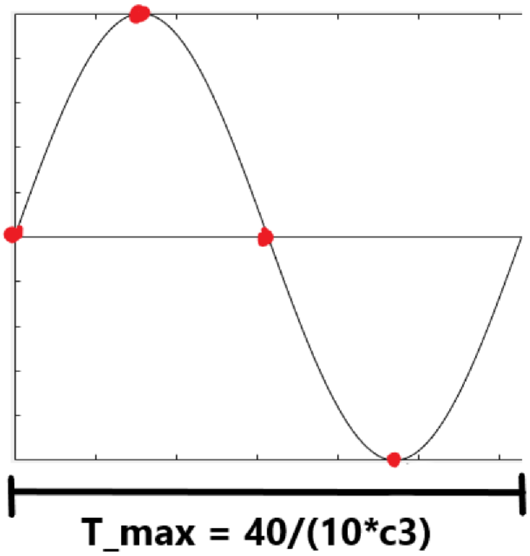
5c)

Requirement: $T_{DAC\ Output} = 0.0001\ sec$

Currently: $T_{x(t)} = 40\ sec$

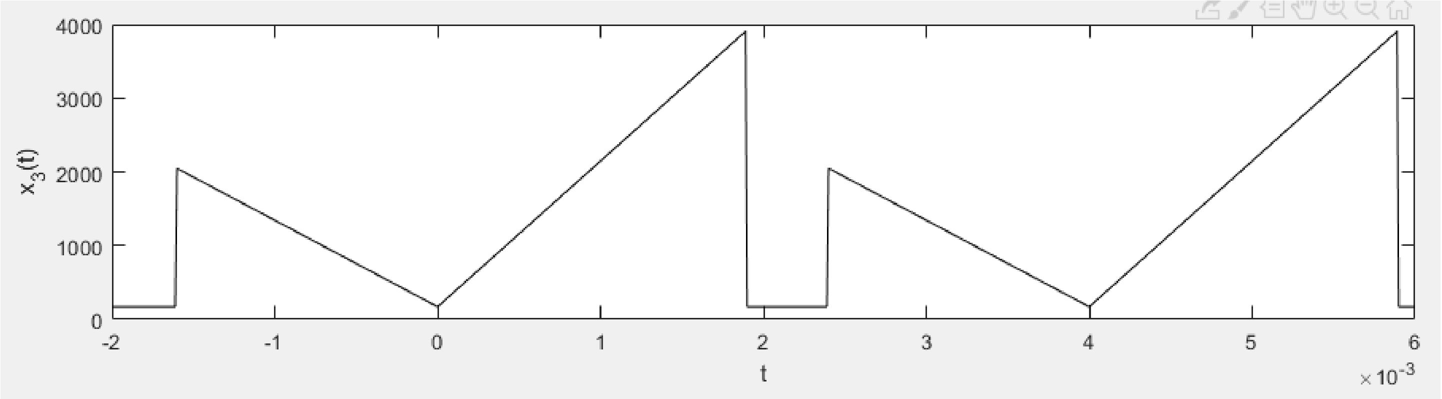
Relationship: $\#\ of\ outputs\ per\ period = \frac{T_{x(t)}/_{10c_3}}{T_{DAC\ Output}}$

Let's have our ideal number of points at the smallest period component (highest frequency) be four. This seems like it would be a decent number. This looks like:



So, $4 = \frac{\left(\frac{40}{10c_3}\right)}{0.0001} \to 0.0004 = \frac{4}{c_3} \to c_3 = 10,000$

Then, $x_3(t) = c_1 + c_2 x(c_3 t)$



d)

$$Y_k = \frac{1}{T_3}\int_{T_3} \left(c_1 + c_2\, x(c_3 t)\right) e^{-jkw_3 t}\, dt$$

$$= \underbrace{\frac{c_1}{T_3}\int_{T_3} e^{-jkw_3 t}\, dt}_{A} + \underbrace{\frac{c_2}{T_3}\int x(c_3 t)\, e^{-jkw_3 t}\, dt}_{B}$$

$$A: \frac{c_1}{T_3}\left[\frac{e^{-jkw_3 t}}{-jkw_3}\right]_0^{T_3}$$

$$= \frac{c_1}{T_3}\left[\cancelto{0}{\frac{e^{-jk2\pi}}{-jkw_3} - \frac{1}{-jkw_3}}\right] = 0$$

$$B: \frac{c_2}{T_3}\left(\underbrace{\int_{\frac{-B-10}{c_3}}^{0} \frac{-1}{B+10}\, t\, e^{-jkw_3 t}\, dt}_{B_1} + \underbrace{\int_{0}^{\frac{A+10}{c_3}} \frac{2}{A+10}\, t\, e^{-jkw_3 t}\, dt}_{B_2}\right)$$

recall $\int u\,dv = uv - \int v\,du$

let $u = t \qquad dv = e^{-jkw_3 t}\, dt$

$du = dt \qquad v = \dfrac{e^{-jkw_3 t}}{-jkw_3}$

$$\int t e^{-jkw_3 t}\, dt = \frac{t e^{-jkw_3 t}}{-jkw_3} + \int \frac{e^{-jkw_3 t}}{jkw_3}\, dt$$

$$= \frac{jt e^{-jk\omega_3 t}}{k\omega_3} + \frac{e^{-jk\omega_3 t}}{k^2\omega_3^2}$$

$$B1: \frac{-1}{B+10}\left[\left(\frac{-1}{k^2\omega_3^2}\right) - \left(\frac{j\left(\frac{-B-10}{c_3}\right)e^{-jk\omega_0(-B-10)}}{k\omega_3} + \frac{e^{-jk\omega_0(-B-10)}}{k^2\omega_3^2}\right)\right]$$

$$= \frac{-1}{B+10}\left[\frac{1}{k^2\omega_3^2} + \frac{j(B+10)e^{jk\omega_0(B+10)}}{kc_3\omega_3} - \frac{e^{jk\omega_0(B+10)}}{k^2\omega_3^2}\right]$$

$$= \frac{-1}{B+10}\left[\left(\frac{1 - e^{jk\omega_0(B+10)}}{k^2\omega_3^2} + j\frac{(B+10)e^{jk\omega_0(B+10)}}{kc_3\omega_3}\right)\right]$$

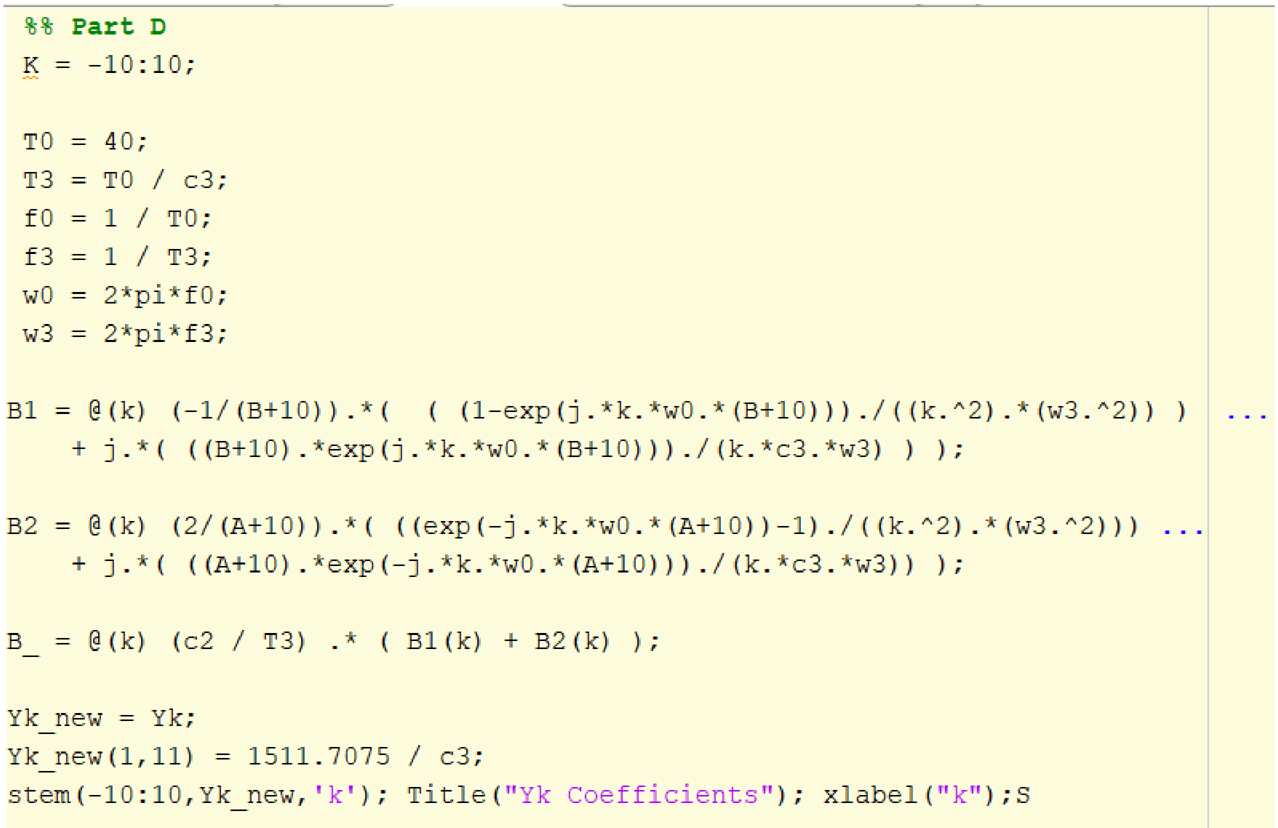
$$B_2: \frac{2}{A+10}\left[\left(\frac{j(A+10)e^{-jk\omega_0(A+10)}}{kc_3\omega_3} + \frac{e^{-jk\omega_0(A+10)}}{k^2\omega_3^2}\right) - \left(\frac{1}{k^2\omega_3^2}\right)\right]$$

$$= \frac{2}{A+10}\left[\frac{e^{-jk\omega_0(A+10)} - 1}{k^2\omega_3^2} + j\frac{(A+10)e^{-jk\omega_0(A+10)}}{kc_3\omega_3}\right]$$

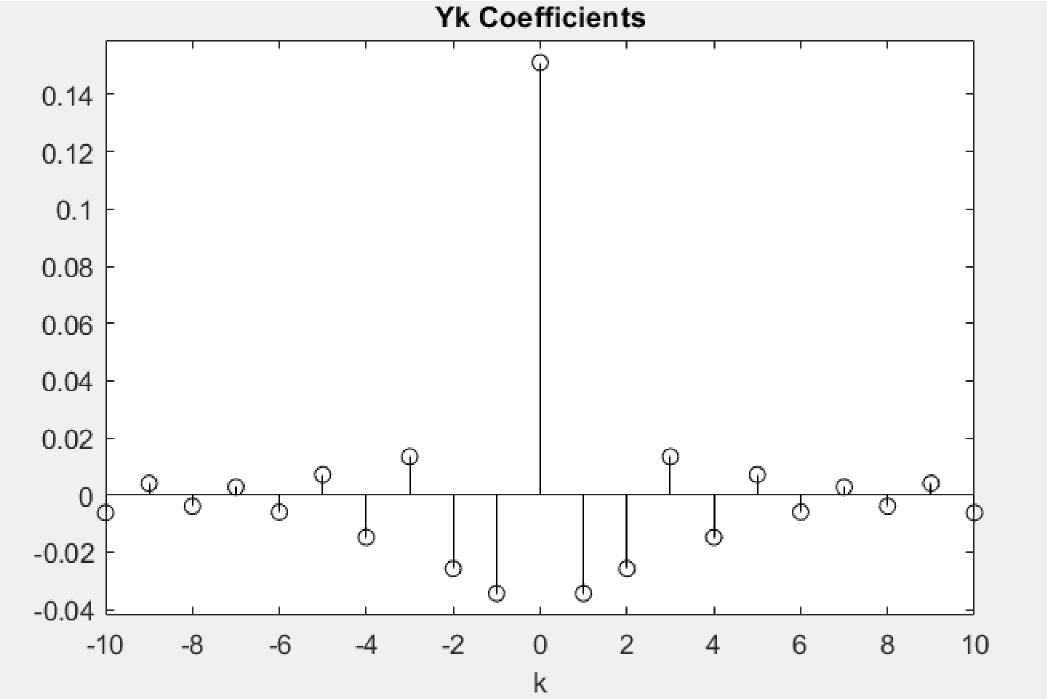
$$B = \frac{c_2}{T_3}\left(B_1 + B_2\right)$$

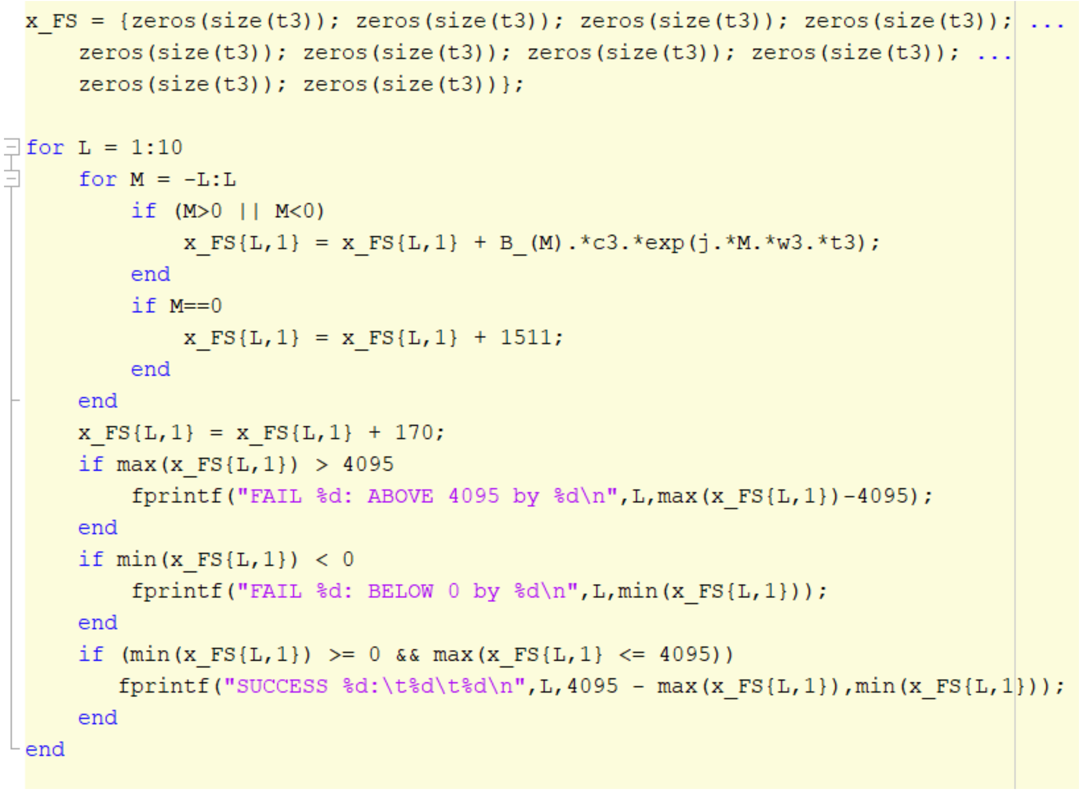
$$Y_k = B$$

In MATLAB,

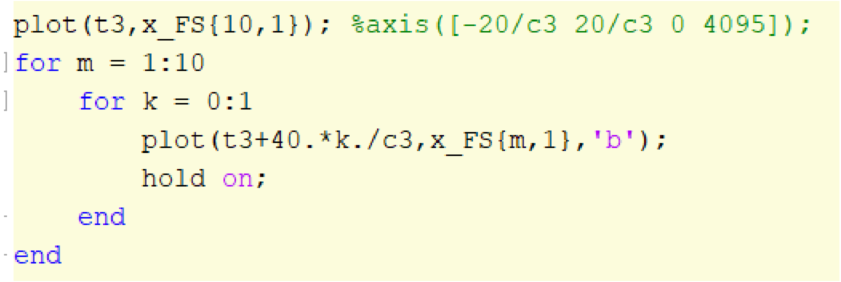


```
 %% Part D
 K = -10:10;

 T0 = 40;
 T3 = T0 / c3;
 f0 = 1 / T0;
 f3 = 1 / T3;
 w0 = 2*pi*f0;
 w3 = 2*pi*f3;

B1 = @(k) (-1/(B+10)).*(  ( (1-exp(j.*k.*w0.*(B+10)))./((k.^2).*(w3.^2)) )  ...
    + j.*( ((B+10).*exp(j.*k.*w0.*(B+10)))./(k.*c3.*w3) ) );

B2 = @(k) (2/(A+10)).*( ((exp(-j.*k.*w0.*(A+10))-1)./((k.^2).*(w3.^2))) ...
    + j.*( ((A+10).*exp(-j.*k.*w0.*(A+10)))./(k.*c3.*w3)) );

B_ = @(k) (c2 / T3) .* ( B1(k) + B2(k) );

Yk_new = Yk;
Yk_new(1,11) = 1511.7075 / c3;
stem(-10:10,Yk_new,'k'); Title("Yk Coefficients"); xlabel("k");S
```

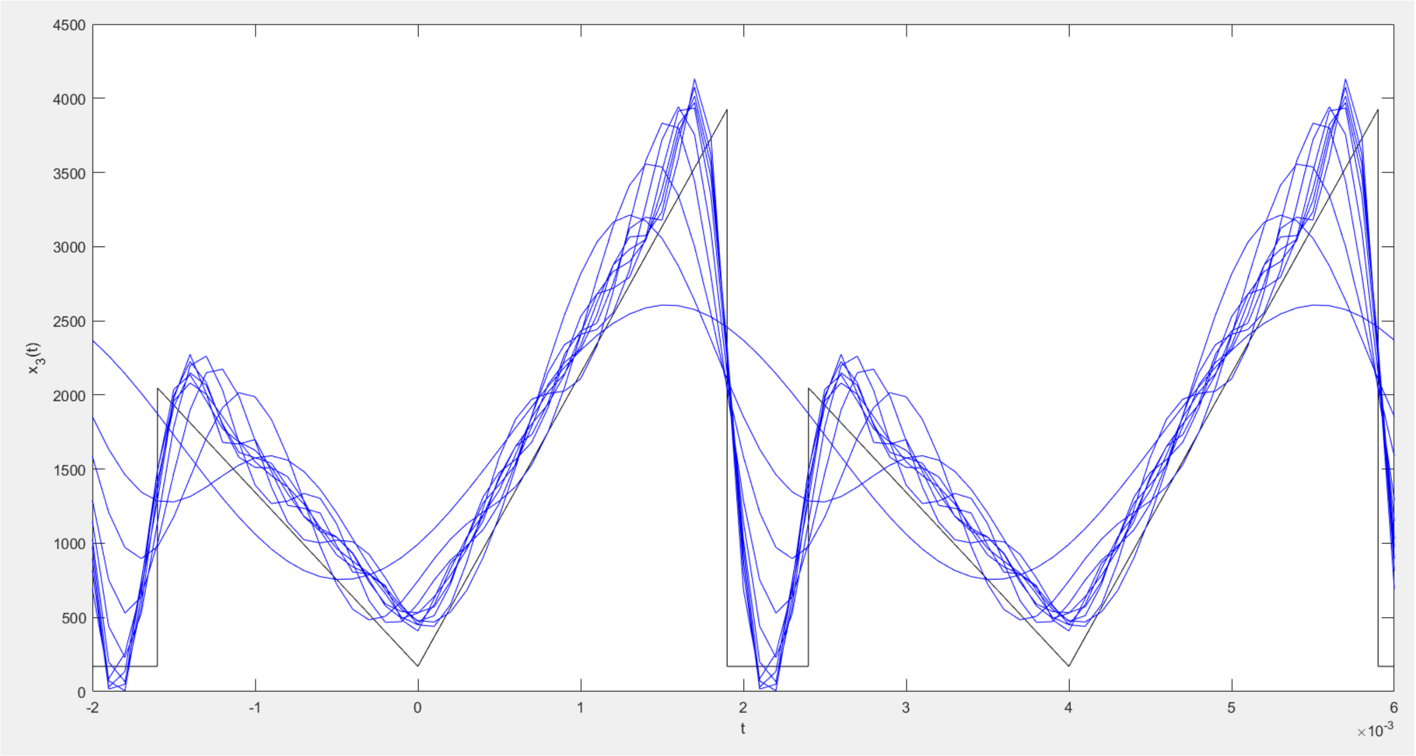


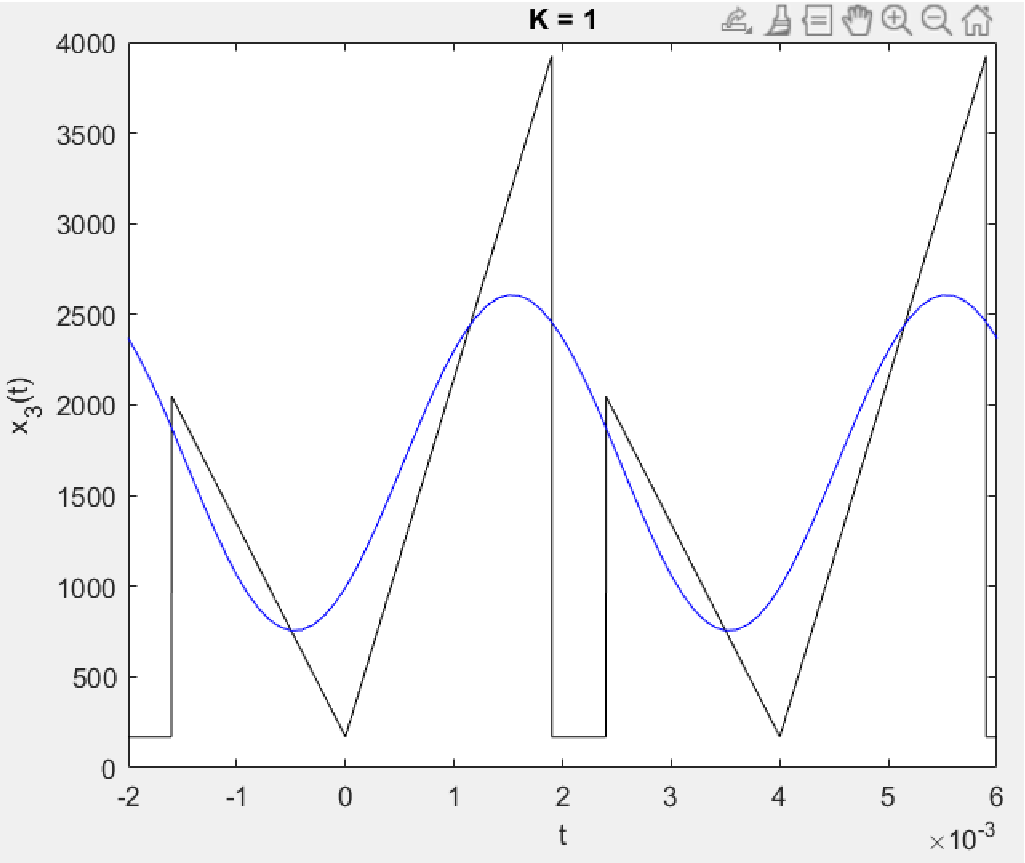


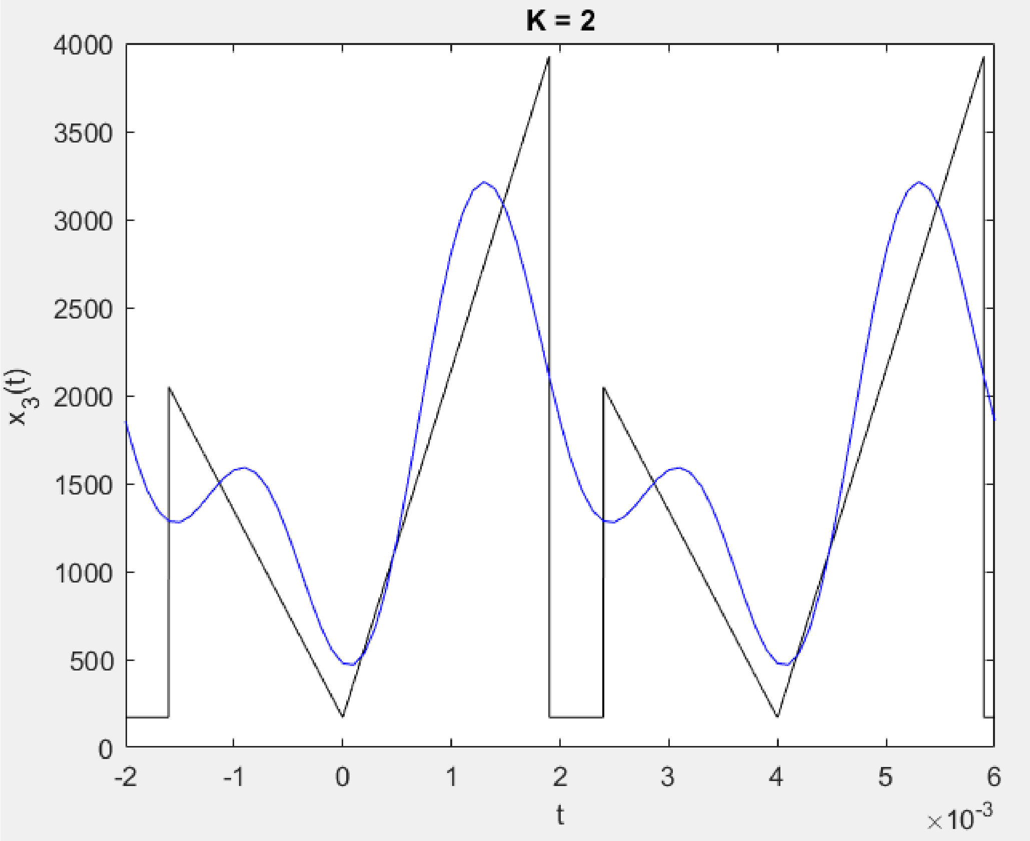
```
x_FS = {zeros(size(t3)); zeros(size(t3)); zeros(size(t3)); zeros(size(t3)); ...
    zeros(size(t3)); zeros(size(t3)); zeros(size(t3)); zeros(size(t3)); ...
    zeros(size(t3)); zeros(size(t3))};

for L = 1:10
    for M = -L:L
        if (M>0 || M<0)
            x_FS{L,1} = x_FS{L,1} + B_(M).*c3.*exp(j.*M.*w3.*t3);
        end
        if M==0
            x_FS{L,1} = x_FS{L,1} + 1511;
        end
    end
    x_FS{L,1} = x_FS{L,1} + 170;
    if max(x_FS{L,1}) > 4095
        fprintf("FAIL %d: ABOVE 4095 by %d\n",L,max(x_FS{L,1})-4095);
    end
    if min(x_FS{L,1}) < 0
        fprintf("FAIL %d: BELOW 0 by %d\n",L,min(x_FS{L,1}));
    end
    if (min(x_FS{L,1}) >= 0 && max(x_FS{L,1} <= 4095))
       fprintf("SUCCESS %d:\t%d\t%d\n",L,4095 - max(x_FS{L,1}),min(x_FS{L,1}));
    end
end
```

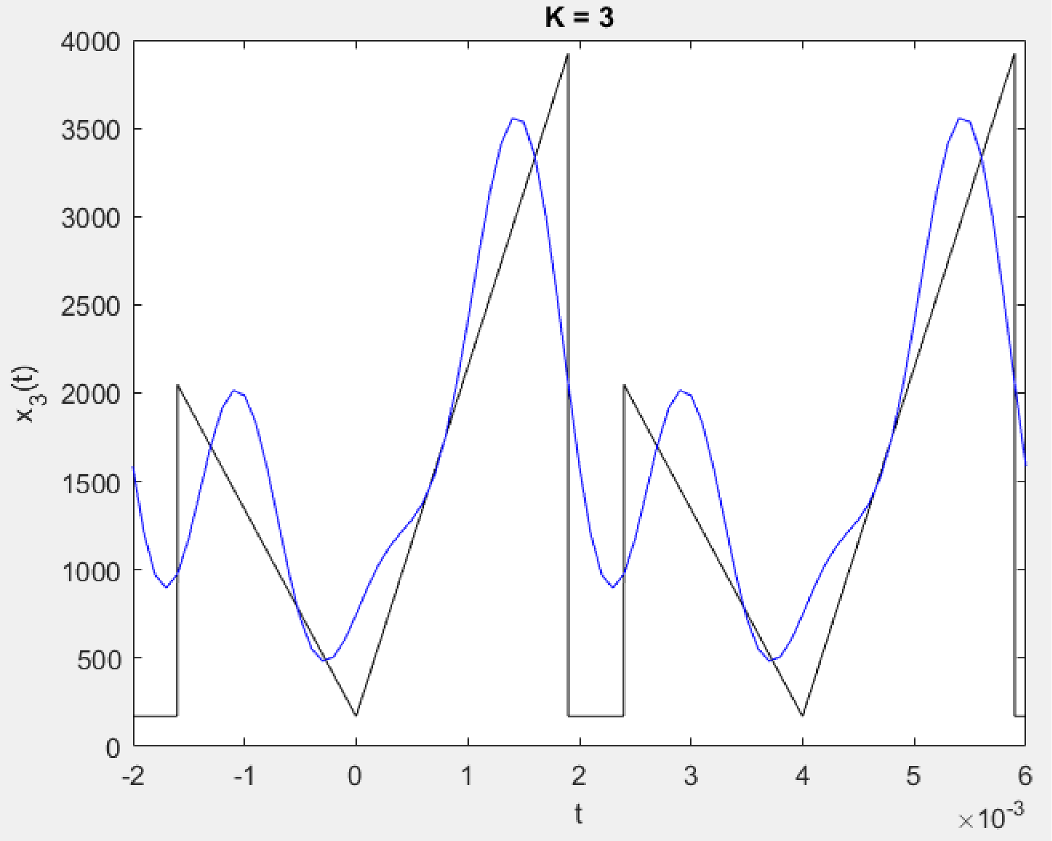


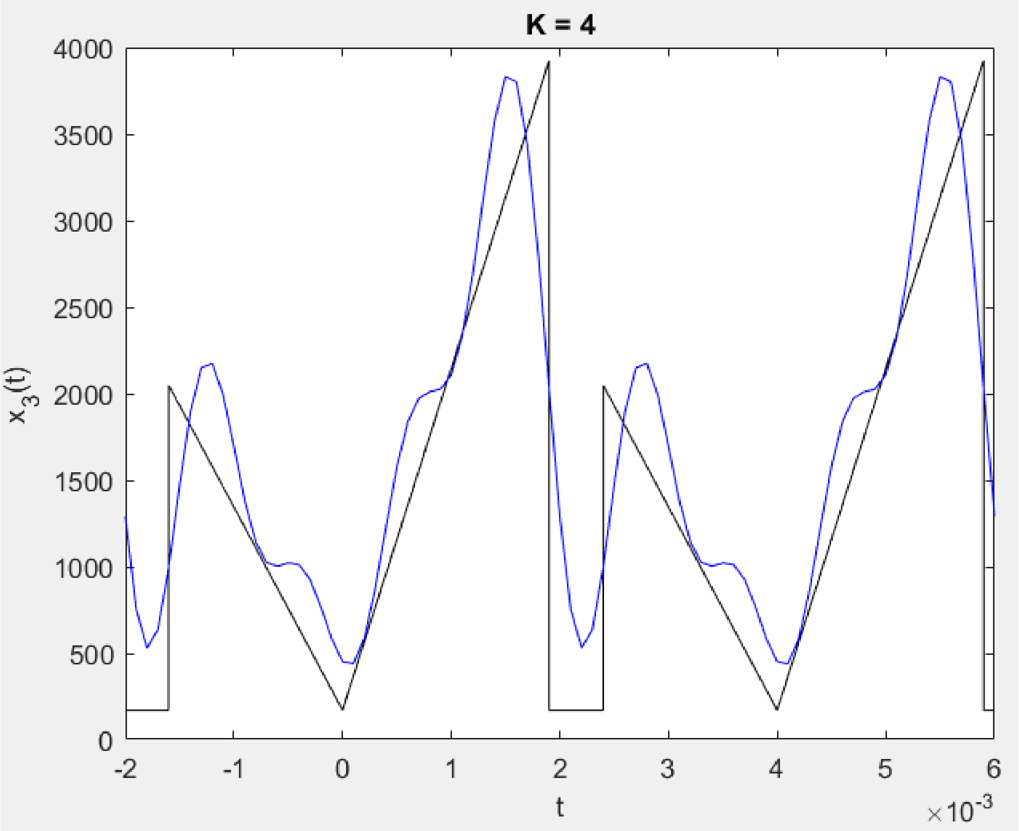
```
plot(t3,x_FS{10,1}); %axis([-20/c3 20/c3 0 4095]);
for m = 1:10
    for k = 0:1
        plot(t3+40.*k./c3,x_FS{m,1},'b');
        hold on;
    end
end
```

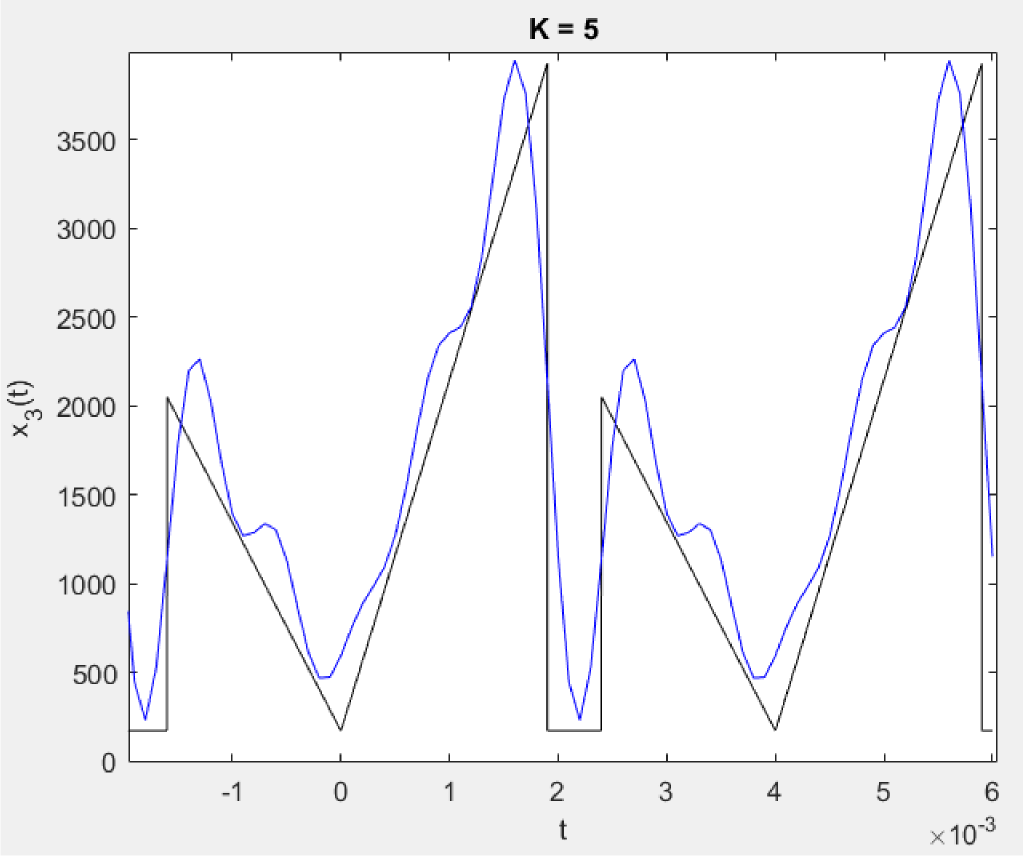


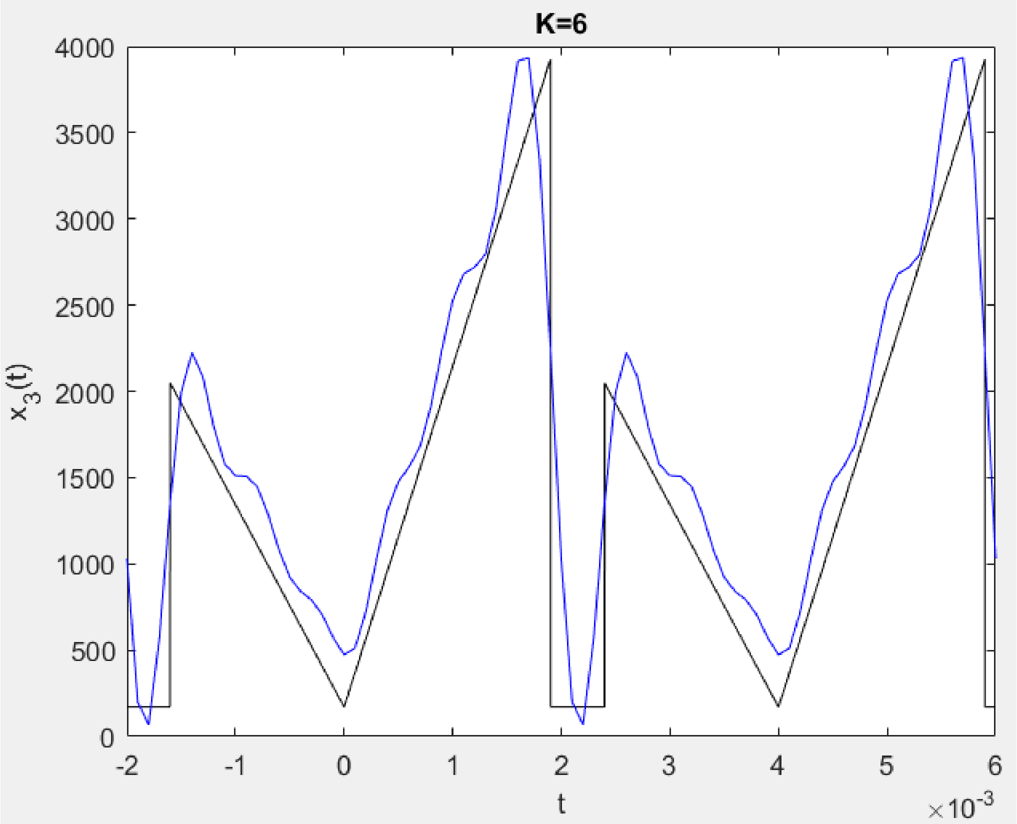

K = 1
4000
3500
3000
2500
2000
1500
1000
500
0
$x_3(t)$
-2
-1
0
1
2
3
4
5
6
t
$\times 10^{-3}$

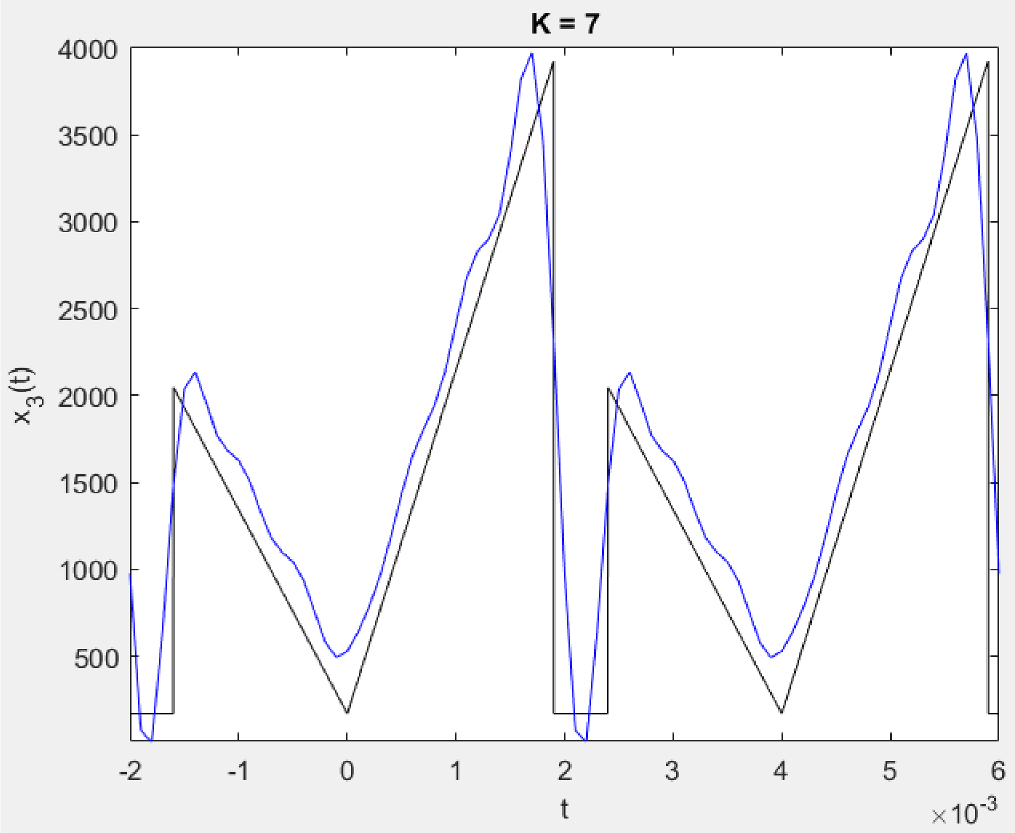

K = 2
4000
3500
3000
2500
2000
1500
1000
500
0
$x_3(t)$
-2
-1
0
1
2
3
4
5
6
t
$\times 10^{-3}$

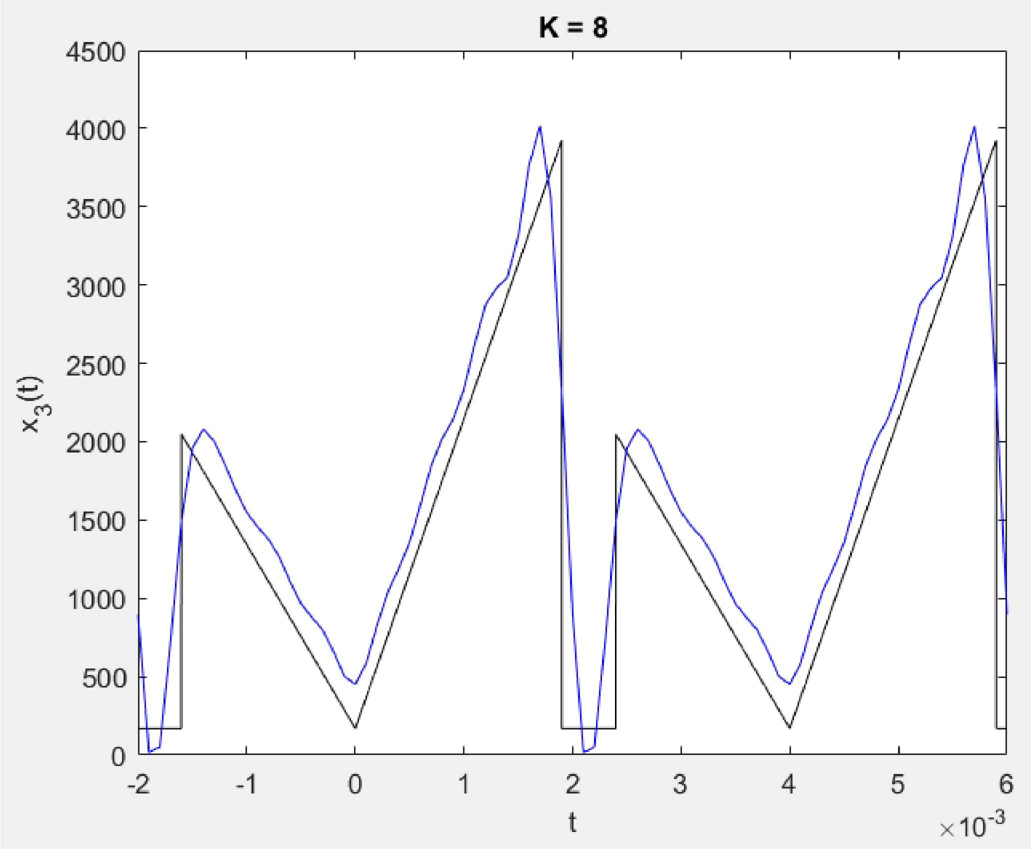

K = 3
4000
3500
3000
2500
2000
1500
1000
500
0
$x_3(t)$
-2
-1
0
1
2
3
4
5
6
t
$\times 10^{-3}$

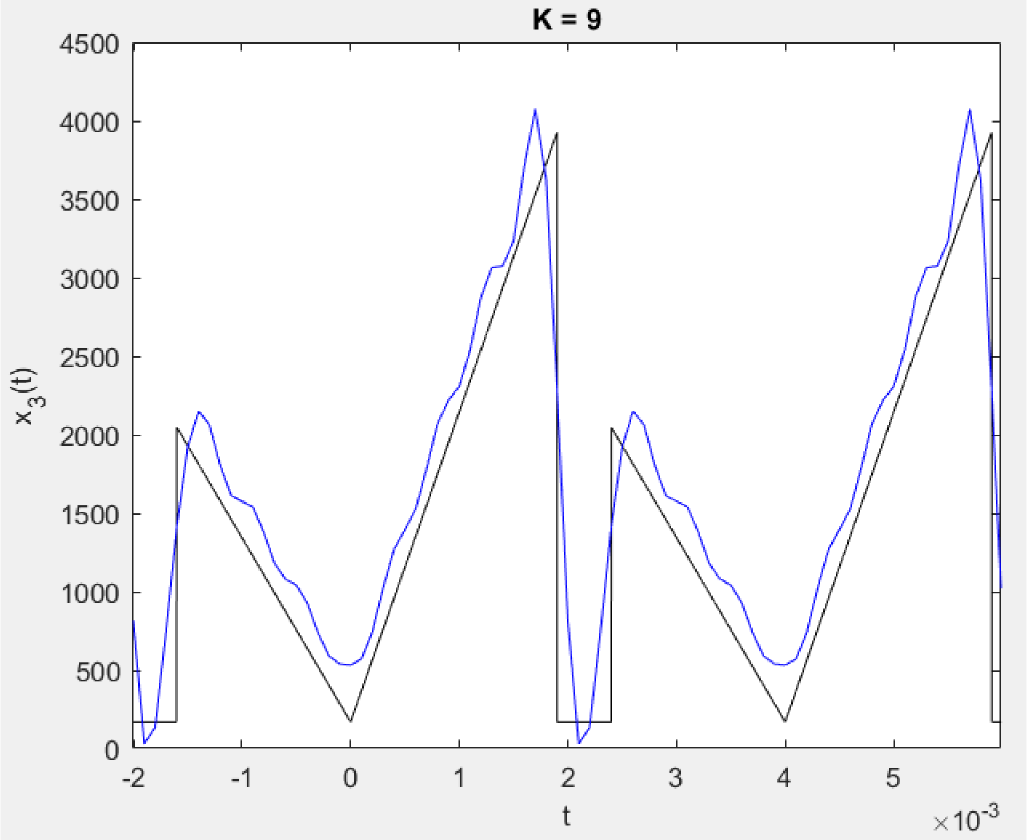

K = 4
4000
3500
3000
2500
2000
1500
1000
500
0
$x_3(t)$
-2
-1
0
1
2
3
4
5
6
t
$\times 10^{-3}$

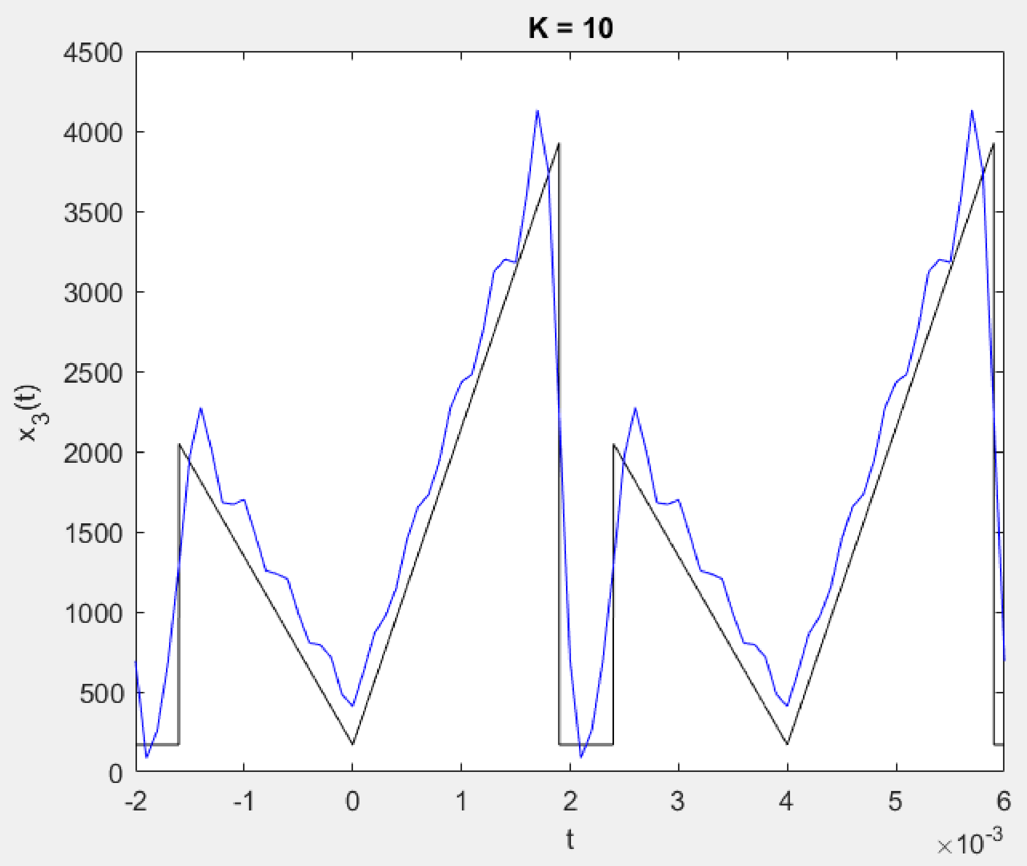

K = 5
x_3(t)
0
500
1000
1500
2000
2500
3000
3500
-1
0
1
2
3
4
5
6
t
×10^-3


K=6
x_3(t)
0
500
1000
1500
2000
2500
3000
3500
4000
-2
-1
0
1
2
3
4
5
6
t
×10^-3

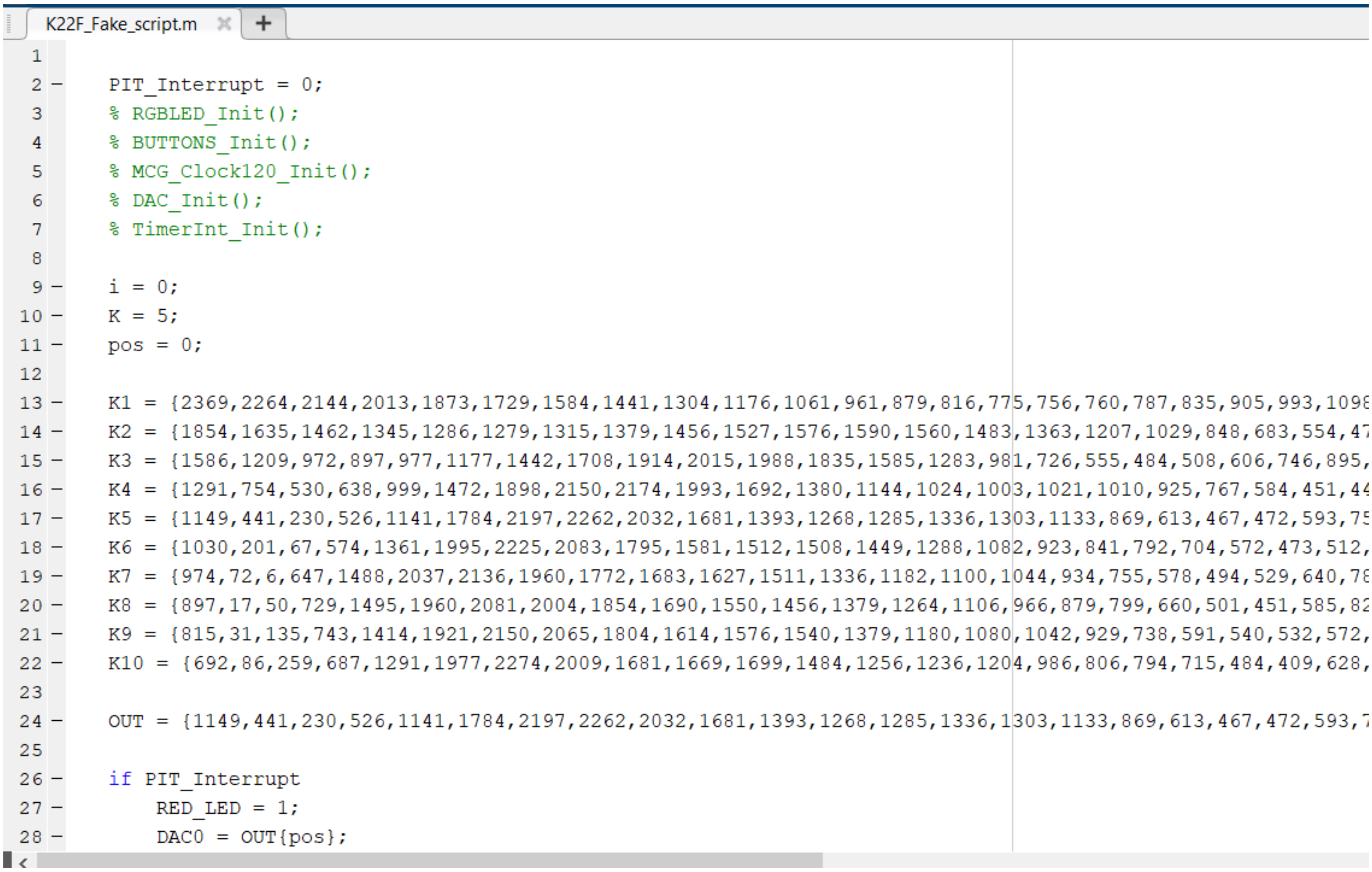

K = 7
4000
3500
3000
2500
2000
1500
1000
500
$x_3(t)$
-2
-1
0
1
2
3
4
5
6
t
$\times 10^{-3}$


K = 8
4500
4000
3500
3000
2500
2000
1500
1000
500
0
$x_3(t)$
-2
-1
0
1
2
3
4
5
6
t
$\times 10^{-3}$

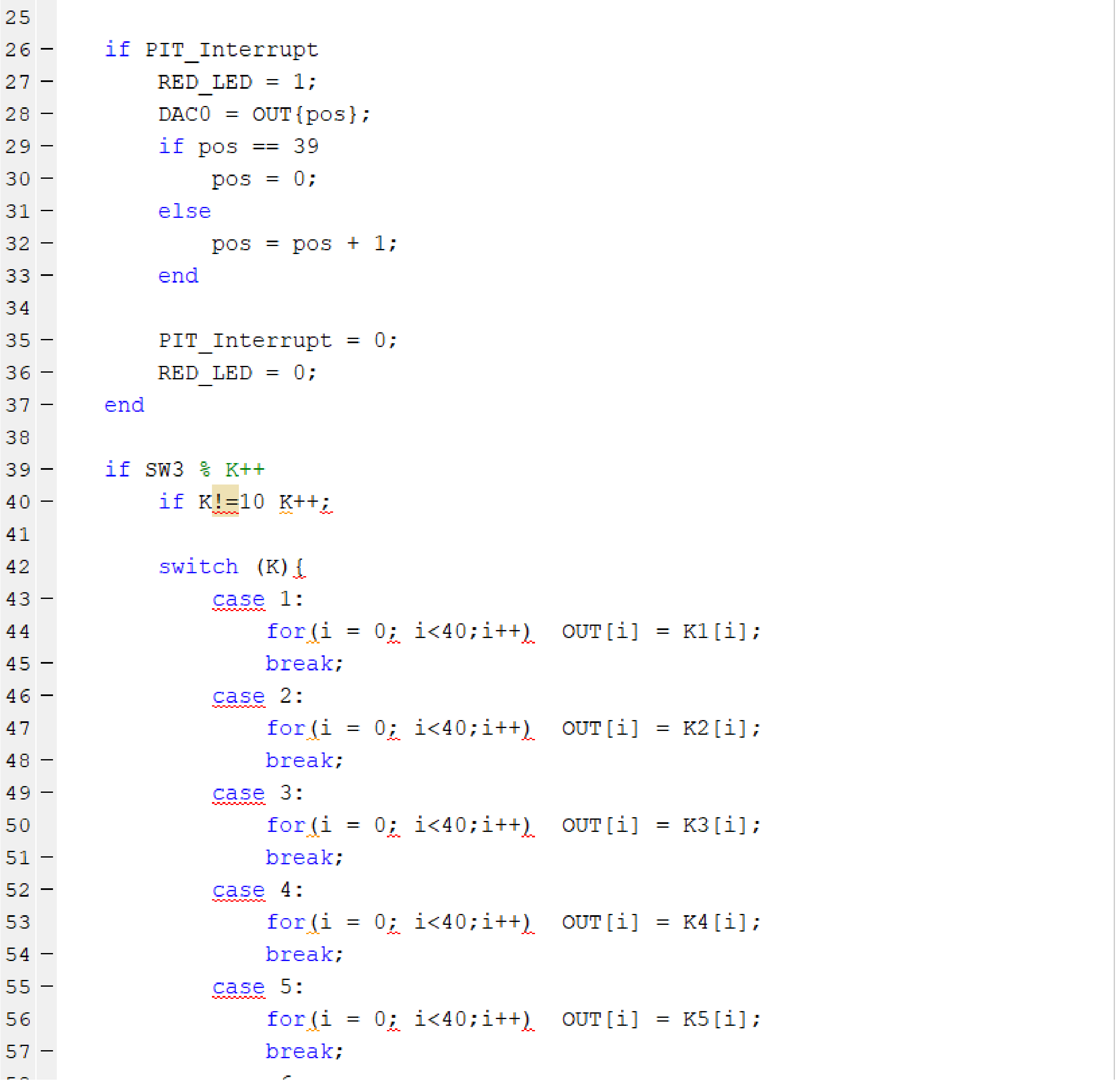

K = 9
4500
4000
3500
3000
2500
2000
1500
1000
500
0
$x_3(t)$
-2
-1
0
1
2
3
4
5
t
$\times 10^{-3}$


K = 10
4500
4000
3500
3000
2500
2000
1500
1000
500
0
$x_3(t)$
-2
-1
0
1
2
3
4
5
6
t
$\times 10^{-3}$

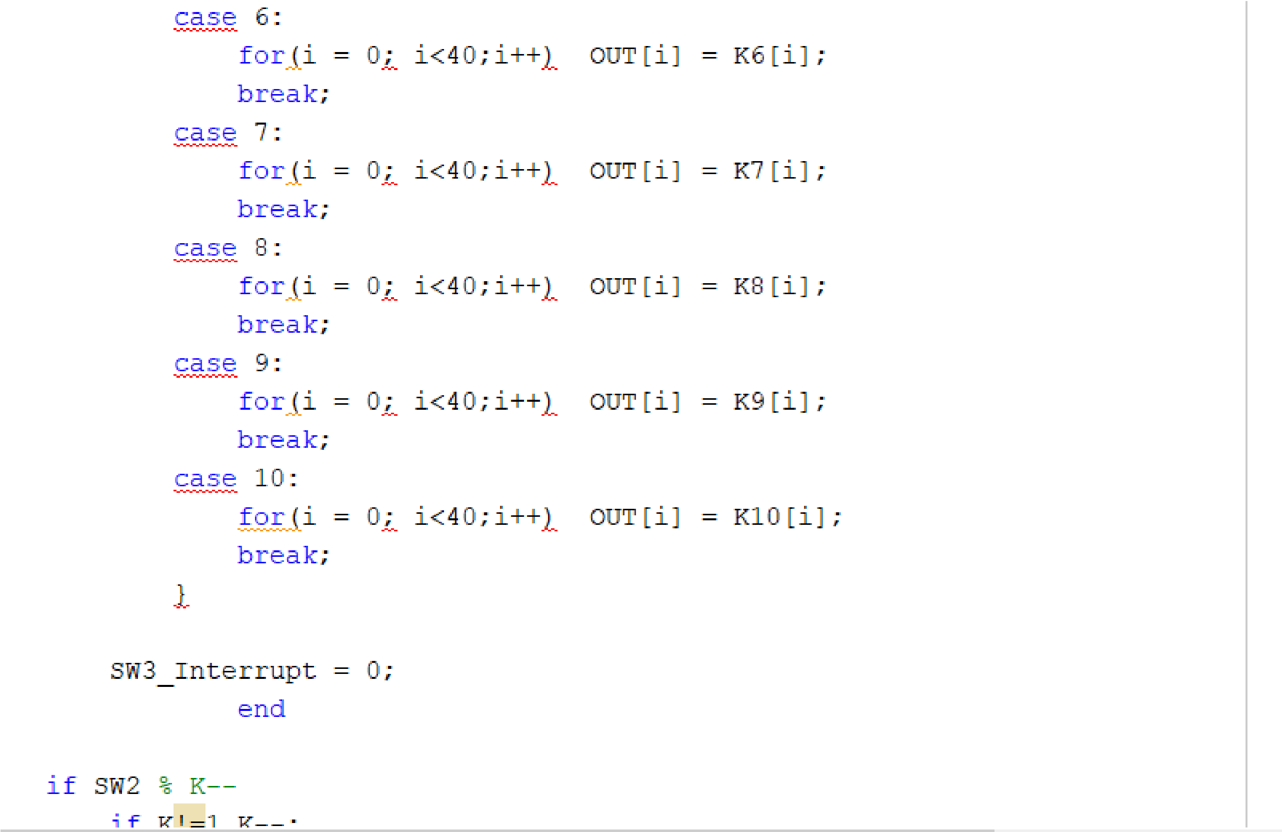
e) From part c, I said I wanted to output four values at the highest frequency component of this signal (when K=10). This means there will be 40 outputs/period. I can easily store 400 (40 outputs / period * 10 different Yk possibilities) + 40 (for output array) = 440, 16-bit unsigned integers. Here’s pseudocode in MATLAB



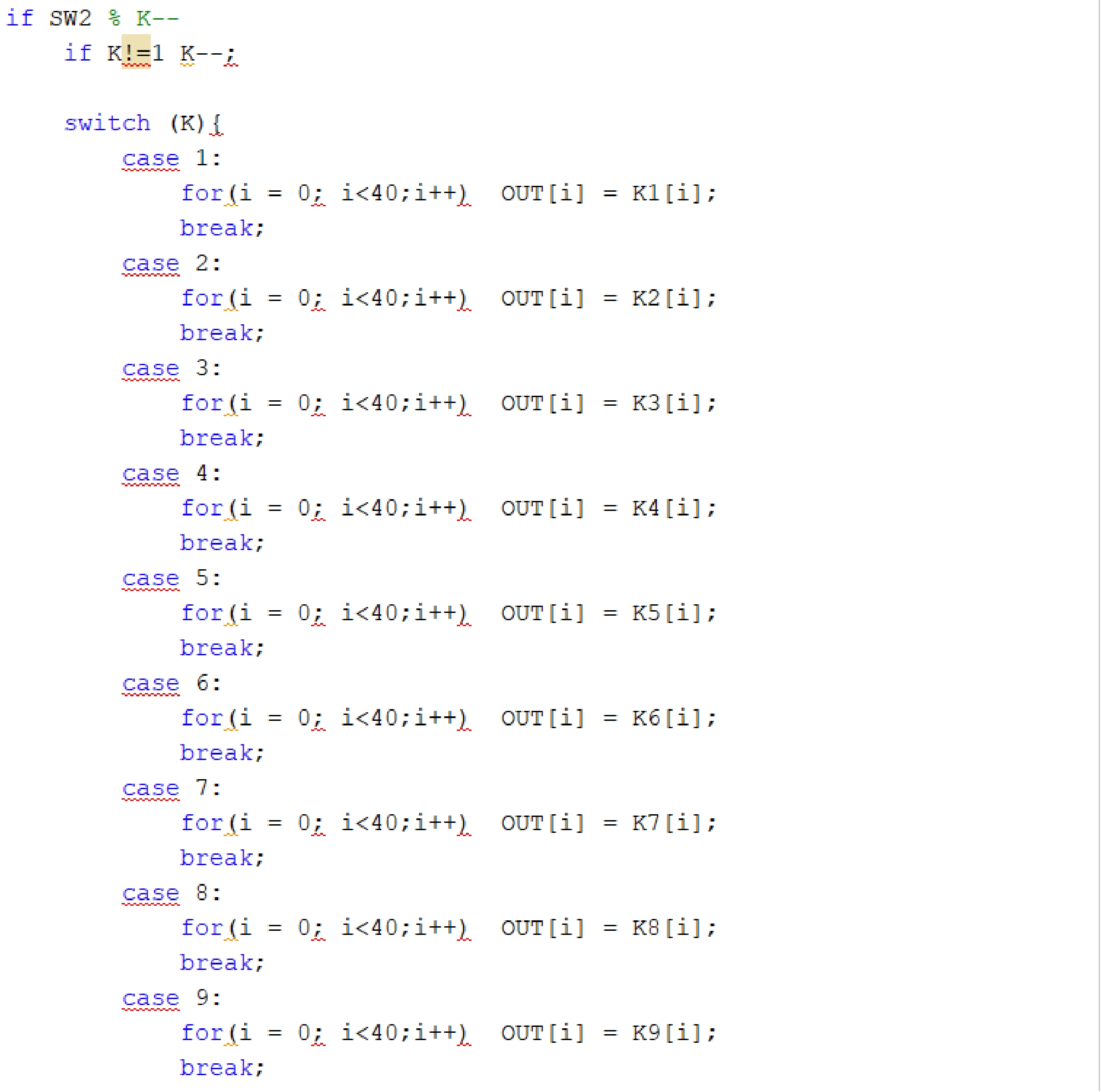
```

PIT_Interrupt = 0;
% RGBLED_Init();
% BUTTONS_Init();
% MCG_Clock120_Init();
% DAC_Init();
% TimerInt_Init();

i = 0;
K = 5;
pos = 0;

K1 = {2369,2264,2144,2013,1873,1729,1584,1441,1304,1176,1061,961,879,816,775,756,760,787,835,905,993,1098
K2 = {1854,1635,1462,1345,1286,1279,1315,1379,1456,1527,1576,1590,1560,1483,1363,1207,1029,848,683,554,47
K3 = {1586,1209,972,897,977,1177,1442,1708,1914,2015,1988,1835,1585,1283,981,726,555,484,508,606,746,895,
K4 = {1291,754,530,638,999,1472,1898,2150,2174,1993,1692,1380,1144,1024,1003,1021,1010,925,767,584,451,44
K5 = {1149,441,230,526,1141,1784,2197,2262,2032,1681,1393,1268,1285,1336,1303,1133,869,613,467,472,593,75
K6 = {1030,201,67,574,1361,1995,2225,2083,1795,1581,1512,1508,1449,1288,1082,923,841,792,704,572,473,512,
K7 = {974,72,6,647,1488,2037,2136,1960,1772,1683,1627,1511,1336,1182,1100,1044,934,755,578,494,529,640,78
K8 = {897,17,50,729,1495,1960,2081,2004,1854,1690,1550,1456,1379,1264,1106,966,879,799,660,501,451,585,82
K9 = {815,31,135,743,1414,1921,2150,2065,1804,1614,1576,1540,1379,1180,1080,1042,929,738,591,540,532,572,
K10 = {692,86,259,687,1291,1977,2274,2009,1681,1669,1699,1484,1256,1236,1204,986,806,794,715,484,409,628,

OUT = {1149,441,230,526,1141,1784,2197,2262,2032,1681,1393,1268,1285,1336,1303,1133,869,613,467,472,593,7

if PIT_Interrupt
    RED_LED = 1;
    DAC0 = OUT{pos};
```



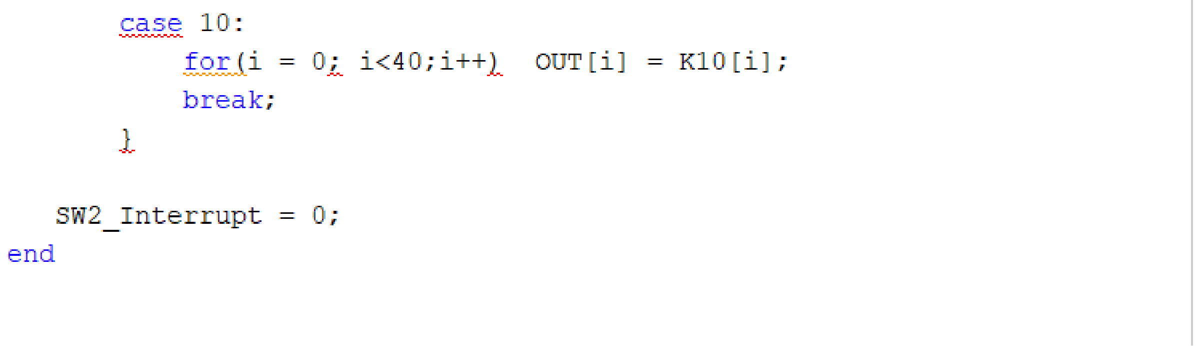
```
if PIT_Interrupt
    RED_LED = 1;
    DAC0 = OUT{pos};
    if pos == 39
        pos = 0;
    else
        pos = pos + 1;
    end

    PIT_Interrupt = 0;
    RED_LED = 0;
end

if SW3 % K++
    if K!=10 K++;

    switch (K){
        case 1:
            for(i = 0; i<40;i++)  OUT[i] = K1[i];
            break;
        case 2:
            for(i = 0; i<40;i++)  OUT[i] = K2[i];
            break;
        case 3:
            for(i = 0; i<40;i++)  OUT[i] = K3[i];
            break;
        case 4:
            for(i = 0; i<40;i++)  OUT[i] = K4[i];
            break;
        case 5:
            for(i = 0; i<40;i++)  OUT[i] = K5[i];
            break;
```



```
        case 6:
            for (i = 0; i<40;i++)  OUT[i] = K6[i];
            break;
        case 7:
            for (i = 0; i<40;i++)  OUT[i] = K7[i];
            break;
        case 8:
            for (i = 0; i<40;i++)  OUT[i] = K8[i];
            break;
        case 9:
            for (i = 0; i<40;i++)  OUT[i] = K9[i];
            break;
        case 10:
            for (i = 0; i<40;i++)  OUT[i] = K10[i];
            break;
        }

    SW3_Interrupt = 0;
            end

if SW2 % K--
    if K!=1 K--;
```

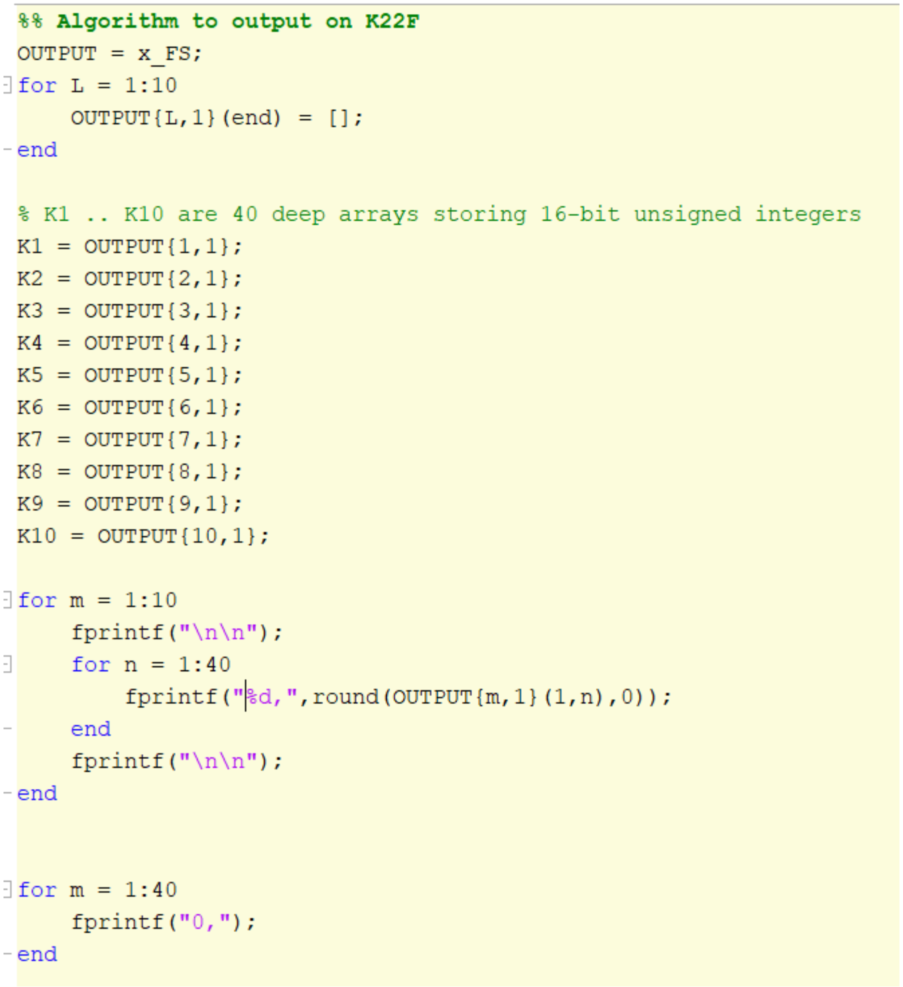


```
if SW2 % K--
    if K!=1 K--;

    switch (K){
        case 1:
            for(i = 0; i<40;i++)  OUT[i] = K1[i];
            break;
        case 2:
            for(i = 0; i<40;i++)  OUT[i] = K2[i];
            break;
        case 3:
            for(i = 0; i<40;i++)  OUT[i] = K3[i];
            break;
        case 4:
            for(i = 0; i<40;i++)  OUT[i] = K4[i];
            break;
        case 5:
            for(i = 0; i<40;i++)  OUT[i] = K5[i];
            break;
        case 6:
            for(i = 0; i<40;i++)  OUT[i] = K6[i];
            break;
        case 7:
            for(i = 0; i<40;i++)  OUT[i] = K7[i];
            break;
        case 8:
            for(i = 0; i<40;i++)  OUT[i] = K8[i];
            break;
        case 9:
            for(i = 0; i<40;i++)  OUT[i] = K9[i];
            break;
```

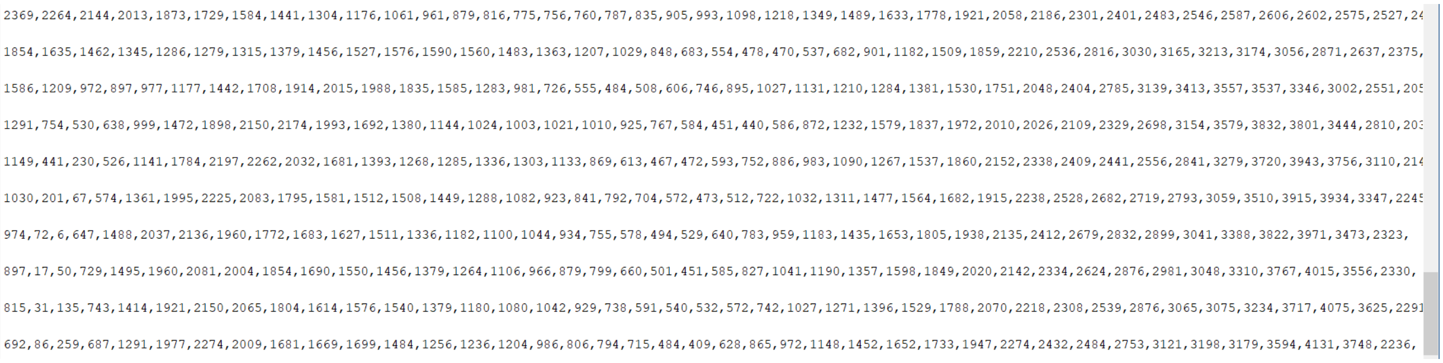


```
        case 10:
            for(i = 0; i<40;i++)  OUT[i] = K10[i];
            break;
        }

    SW2_Interrupt = 0;
end
```

MATLAB for all the numbers I got for K1, K2, K3 … K10 40 length arrays:



```
%% Algorithm to output on K22F
OUTPUT = x_FS;
for L = 1:10
    OUTPUT{L,1}(end) = [];
end

% K1 .. K10 are 40 deep arrays storing 16-bit unsigned integers
K1 = OUTPUT{1,1};
K2 = OUTPUT{2,1};
K3 = OUTPUT{3,1};
K4 = OUTPUT{4,1};
K5 = OUTPUT{5,1};
K6 = OUTPUT{6,1};
K7 = OUTPUT{7,1};
K8 = OUTPUT{8,1};
K9 = OUTPUT{9,1};
K10 = OUTPUT{10,1};

for m = 1:10
    fprintf("\n\n");
    for n = 1:40
        fprintf("%d,",round(OUTPUT{m,1}(1,n),0));
    end
    fprintf("\n\n");
end


for m = 1:40
    fprintf("0,");
end
```



```
2369,2264,2144,2013,1873,1729,1584,1441,1304,1176,1061,961,879,816,775,756,760,787,835,905,993,1098,1218,1349,1489,1633,1778,1921,2058,2186,2301,2401,2483,2546,2587,2606,2602,2575,2527,24

1854,1635,1462,1345,1286,1279,1315,1379,1456,1527,1576,1590,1560,1483,1363,1207,1029,848,683,554,478,470,537,682,901,1182,1509,1859,2210,2536,2816,3030,3165,3213,3174,3056,2871,2637,2375,

1586,1209,972,897,977,1177,1442,1708,1914,2015,1988,1835,1585,1283,981,726,555,484,508,606,746,895,1027,1131,1210,1284,1381,1530,1751,2048,2404,2785,3139,3413,3557,3537,3346,3002,2551,205

1291,754,530,638,999,1472,1898,2150,2174,1993,1692,1380,1144,1024,1003,1021,1010,925,767,584,451,440,586,872,1232,1579,1837,1972,2010,2026,2109,2329,2698,3154,3579,3832,3801,3444,2810,203

1149,441,230,526,1141,1784,2197,2262,2032,1681,1393,1268,1285,1336,1303,1133,869,613,467,472,593,752,886,983,1090,1267,1537,1860,2152,2338,2409,2441,2556,2841,3279,3720,3943,3756,3110,214

1030,201,67,574,1361,1995,2225,2083,1795,1581,1512,1508,1449,1288,1082,923,841,792,704,572,473,512,722,1032,1311,1477,1564,1682,1915,2238,2528,2682,2719,2793,3059,3510,3915,3934,3347,2245

974,72,6,647,1488,2037,2136,1960,1772,1683,1627,1511,1336,1182,1100,1044,934,755,578,494,529,640,783,959,1183,1435,1653,1805,1938,2135,2412,2679,2832,2899,3041,3388,3822,3971,3473,2323,

897,17,50,729,1495,1960,2081,2004,1854,1690,1550,1456,1379,1264,1106,966,879,799,660,501,451,585,827,1041,1190,1357,1598,1849,2020,2142,2334,2624,2876,2981,3048,3310,3767,4015,3556,2330,

815,31,135,743,1414,1921,2150,2065,1804,1614,1576,1540,1379,1180,1080,1042,929,738,591,540,532,572,742,1027,1271,1396,1529,1788,2070,2218,2308,2539,2876,3065,3075,3234,3717,4075,3625,2291

692,86,259,687,1291,1977,2274,2009,1681,1669,1699,1484,1256,1236,1204,986,806,794,715,484,409,628,865,972,1148,1452,1652,1733,1947,2274,2432,2484,2753,3121,3198,3179,3594,4131,3748,2236,
```

f) Keil

main.c)

```
#include "MK22F51212.h"                                    //Device header
#include "MCG.h"                                           //Clock header
#include "TimerInt.h"                                      //Timer Interrupt Header
#include "DAC.h"                                           //DAC Header
#include "BUTTONS.h"                                       //BUTTONS Header
#include "RGBLED.h"                                        //RGB LED Header

uint8_t i = 0;
uint8_t K = 5;
uint8_t pos = 0;
uint8_t Kinc = 0;
uint8_t Kdec = 0;

uint16_t K1[] =
{2369,2264,2144,2013,1873,1729,1584,1441,1304,1176,1061,961,879,816,775,756,760,787,835,905,993,1098,1218,13
49,1489,1633,1778,1921,2058,2186,2301,2401,2483,2546,2587,2606,2602,2575,2527,2457};
uint16_t K2[] =
{1854,1635,1462,1345,1286,1279,1315,1379,1456,1527,1576,1590,1560,1483,1363,1207,1029,848,683,554,478,470,53
7,682,901,1182,1509,1859,2210,2536,2816,3030,3165,3213,3174,3056,2871,2637,2375,2107};
uint16_t K3[] =
{1586,1209,972,897,977,1177,1442,1708,1914,2015,1988,1835,1585,1283,981,726,555,484,508,606,746,895,1027,113
1,1210,1284,1381,1530,1751,2048,2404,2785,3139,3413,3557,3537,3346,3002,2551,2055};
uint16_t K4[] =
{1291,754,530,638,999,1472,1898,2150,2174,1993,1692,1380,1144,1024,1003,1021,1010,925,767,584,451,440,586,87
2,1232,1579,1837,1972,2010,2026,2109,2329,2698,3154,3579,3832,3801,3444,2810,2033};
uint16_t K5[] =
{1149,441,230,526,1141,1784,2197,2262,2032,1681,1393,1268,1285,1336,1303,1133,869,613,467,472,593,752,886,98
3,1090,1267,1537,1860,2152,2338,2409,2441,2556,2841,3279,3720,3943,3756,3110,2145};
uint16_t K6[] =
{1030,201,67,574,1361,1995,2225,2083,1795,1581,1512,1508,1449,1288,1082,923,841,792,704,572,473,512,722,1032,
1311,1477,1564,1682,1915,2238,2528,2682,2719,2793,3059,3510,3915,3934,3347,2245};
uint16_t K7[] =
{974,72,6,647,1488,2037,2136,1960,1772,1683,1627,1511,1336,1182,1100,1044,934,755,578,494,529,640,783,959,118
3,1435,1653,1805,1938,2135,2412,2679,2832,2899,3041,3388,3822,3971,3473,2323};
uint16_t K8[] =
{897,17,50,729,1495,1960,2081,2004,1854,1690,1550,1456,1379,1264,1106,966,879,799,660,501,451,585,827,1041,11
90,1357,1598,1849,2020,2142,2334,2624,2876,2981,3048,3310,3767,4015,3556,2330};
uint16_t K9[] =
{815,31,135,743,1414,1921,2150,2065,1804,1614,1576,1540,1379,1180,1080,1042,929,738,591,540,532,572,742,1027,
1271,1396,1529,1788,2070,2218,2308,2539,2876,3065,3075,3234,3717,4075,3625,2291};
uint16_t K10[] =
{692,86,259,687,1291,1977,2274,2009,1681,1669,1699,1484,1256,1236,1204,986,806,794,715,484,409,628,865,972,11
```

```
48,1452,1652,1733,1947,2274,2432,2484,2753,3121,3198,3179,3594,4095,3748,2236}; // 4095 value in this line was actually 4131...

uint16_t OUT[] = {1149,441,230,526,1141,1784,2197,2262,2032,1681,1393,1268,1285,1336,1303,1133,869,613,467,472,593,752,886,983,1090,1267,1537,1860,2152,2338,2409,2441,2556,2841,3279,3720,3943,3756,3110,2145};

void PIT0_IRQHandler(void){      //This function is called when the timer interrupt expires
        //Place Interrupt Service Routine Here
        GPIOA->PSOR          |= GPIO_PSOR_PTSO(0x1u << 1); // R = 1

        DAC0->DAT[0].DATL = DAC_DATL_DATA0(OUT[pos] & 0xFF);              //Set Lower 8 bits of Output
        DAC0->DAT[0].DATH = DAC_DATH_DATA1(OUT[pos] >> 0x8);              //Set Higher 8 bits of Output

        if(pos == 39) pos = 0;
        else pos++;

        NVIC_ClearPendingIRQ(PIT0_IRQn);                         //Clears interrupt flag in NVIC Register
        PIT->CHANNEL[0].TFLG  = PIT_TFLG_TIF_MASK;               //Clears interrupt flag in PIT Register

        GPIOA->PCOR          |= GPIO_PCOR_PTCO(0x1u << 1); // R = 0
}

// K++ BUTTON
void PORTB_IRQHandler(void){ //This function might be called when the SW3 is pushed
        if(K!=10) K++;

        Kinc++;
        switch (K){
                case 1:
                        for(i = 0; i<40;i++)  OUT[i] = K1[i];
                        break;
                case 2:
                        for(i = 0; i<40;i++)  OUT[i] = K2[i];
                        break;
                case 3:
                        for(i = 0; i<40;i++)  OUT[i] = K3[i];
                        break;
                case 4:
                        for(i = 0; i<40;i++)  OUT[i] = K4[i];
                        break;
                case 5:
                        for(i = 0; i<40;i++)  OUT[i] = K5[i];
                        break;
                case 6:
                        for(i = 0; i<40;i++)  OUT[i] = K6[i];
                        break;
                case 7:
```

```
            for(i = 0; i<40;i++)  OUT[i] = K7[i];
            break;
        case 8:
            for(i = 0; i<40;i++)  OUT[i] = K8[i];
            break;
        case 9:
            for(i = 0; i<40;i++)  OUT[i] = K9[i];
            break;
        case 10:
            for(i = 0; i<40;i++)  OUT[i] = K10[i];
            break;
        }

    NVIC_ClearPendingIRQ(PORTB_IRQn);         //CMSIS Function to clear pending interrupts on PORTB
    PORTB->ISFR                                                = (0x1u << 17);

}

// K-- BUTTON
void PORTC_IRQHandler(void){ //This function might be called when the SW2 is pushed

    if(K!=1) K--;
    Kdec++;
    switch (K){
        case 1:
            for(i = 0; i<40;i++)  OUT[i] = K1[i];
            break;
        case 2:
            for(i = 0; i<40;i++)  OUT[i] = K2[i];
            break;
        case 3:
            for(i = 0; i<40;i++)  OUT[i] = K3[i];
            break;
        case 4:
            for(i = 0; i<40;i++)  OUT[i] = K4[i];
            break;
        case 5:
            for(i = 0; i<40;i++)  OUT[i] = K5[i];
            break;
        case 6:
            for(i = 0; i<40;i++)  OUT[i] = K6[i];
            break;
        case 7:
            for(i = 0; i<40;i++)  OUT[i] = K7[i];
            break;
        case 8:
            for(i = 0; i<40;i++)  OUT[i] = K8[i];
            break;
```

```
                case 9:
                        for(i = 0; i<40;i++)  OUT[i] = K9[i];
                        break;
                case 10:
                        for(i = 0; i<40;i++)  OUT[i] = K10[i];
                        break;
                }

        NVIC_ClearPendingIRQ(PORTC_IRQn);               //CMSIS Function to clear pending interrupts on PORTC
        PORTC->ISFR                                                     = (0x1u << 1);
}

int main(void){

        RGBLED_Init();
        BUTTONS_Init();
        MCG_Clock120_Init();
        DAC_Init();
        TimerInt_Init();
        while(1){

        }
}
```

RGBLED.c)

```
#include "MK22F51212.h"                                 //Device header

#include "RGBLED.h"                             // RGBLED header


void RGBLED_Init(void){

                SIM->SCGC5          |= SIM_SCGC5_PORTA_MASK;            //Enables clock to PORTA

                SIM->SCGC5          |= SIM_SCGC5_PORTD_MASK; //Enables clock to PORTD

                PORTA->PCR[1]        = PORT_PCR_MUX(0x1u);   // Set Signal Multiplexing to ALT1 for PTA1

                PORTA->PCR[2]        = PORT_PCR_MUX(0x1u);  // Set Signal Multiplexing to ALT1 for PTA2

                PORTD->PCR[5]        = PORT_PCR_MUX(0x1u); // Set Signal Multiplexing to ALT1 for PTD5


                GPIOA->PDDR         |= GPIO_PDDR_PDD(~(0x0u << 1)); //Sets PTA1 to Output GPIO
```

```
                 GPIOA->PDDR        |= GPIO_PDDR_PDD(~(0x0u << 2)); //Sets PTA2 to Output GPIO

                GPIOD->PDDR        |= GPIO_PDDR_PDD(~(0x0u << 5)); //Sets PTD5 to Output GPIO


                GPIOA->PDDR        |= GPIO_PDOR_PDO(0x1u << 1); // R = 0

                GPIOA->PDOR                                     |= GPIO_PDOR_PDO(0x1u << 2); // G = 0

                GPIOD->PDOR                                     |= GPIO_PDOR_PDO(0x1u << 5); // B = 0

}//End RGBLED_Init
```

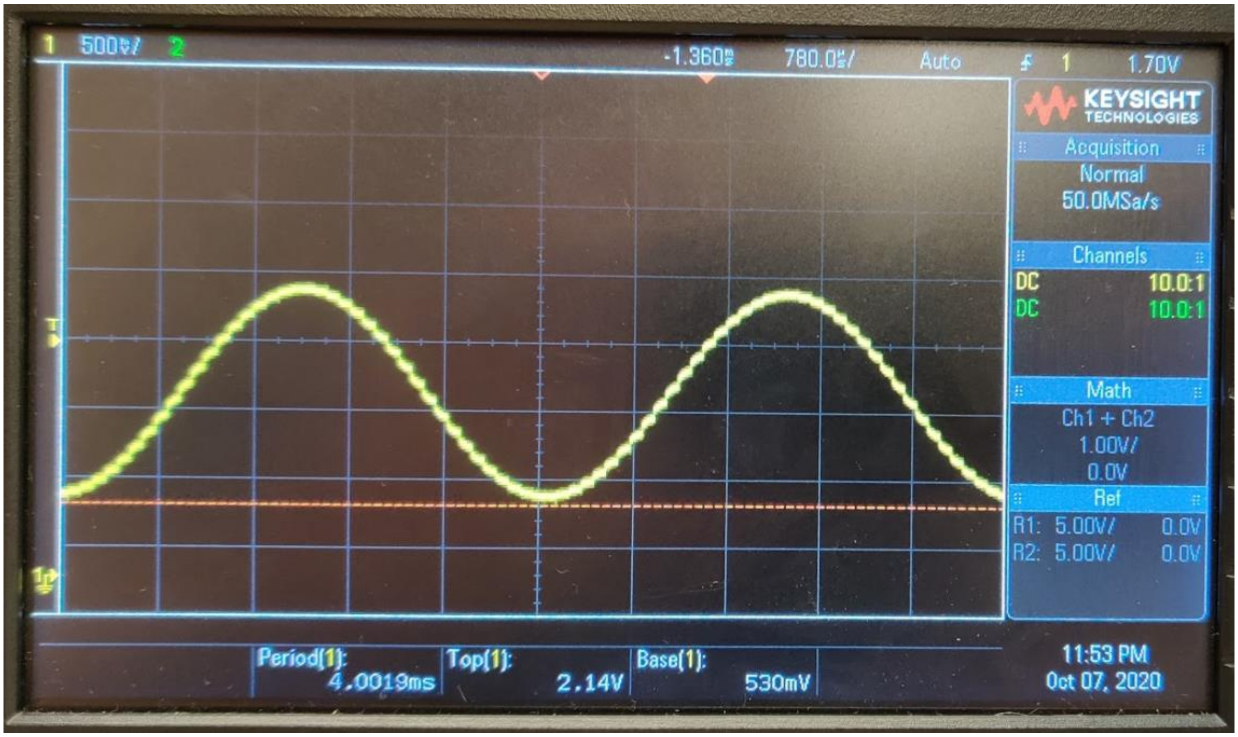
BUTTONS.c)

```
#include "MK22F51212.h"                                //Device header

#include "BUTTONS.h"                                   // BUTTONS header

void BUTTONS_Init(void){

   SIM->SCGC5          |= SIM_SCGC5_PORTB_MASK;                               //Enables Clock to PORTB

   SIM->SCGC5          |= SIM_SCGC5_PORTC_MASK;                               //Enables Clock to PORTC

   PORTB->PCR[17]       = PORT_PCR_MUX(0x1u);                                 //Set Signal Multiplexing to

   PORTC->PCR[1]        = PORT_PCR_MUX(0x1u);                                 //Set Signal Multiplexing to


   GPIOB->PDDR         |= GPIO_PDDR_PDD(~(0x1u << 17)); //Sets PTB17 to Input GPIO

   GPIOC->PDDR         |= GPIO_PDDR_PDD(~(0x1u << 1));  //Sets PTC1 to Input GPIO


   PORTB->PCR[17]      |= PORT_PCR_IRQC(0xA);    //This configures the interrupt flag to be set on a falling edge

   PORTC->PCR[1]       |= PORT_PCR_IRQC(0xA);    //This configures the interrupt flag to be set on a falling edge


   NVIC_ClearPendingIRQ(PORTB_IRQn);            //CMSIS Function to clear pending interrupts on PORTB

   NVIC_ClearPendingIRQ(PORTC_IRQn);            //CMSIS Function to clear pending interrupts on PORTC

   NVIC_EnableIRQ(PORTB_IRQn);                  //CMSIS Function to enable interrupt via PORTB

   NVIC_EnableIRQ(PORTC_IRQn);                  //CMSIS Function to enable interrupt via PORTC

}
```

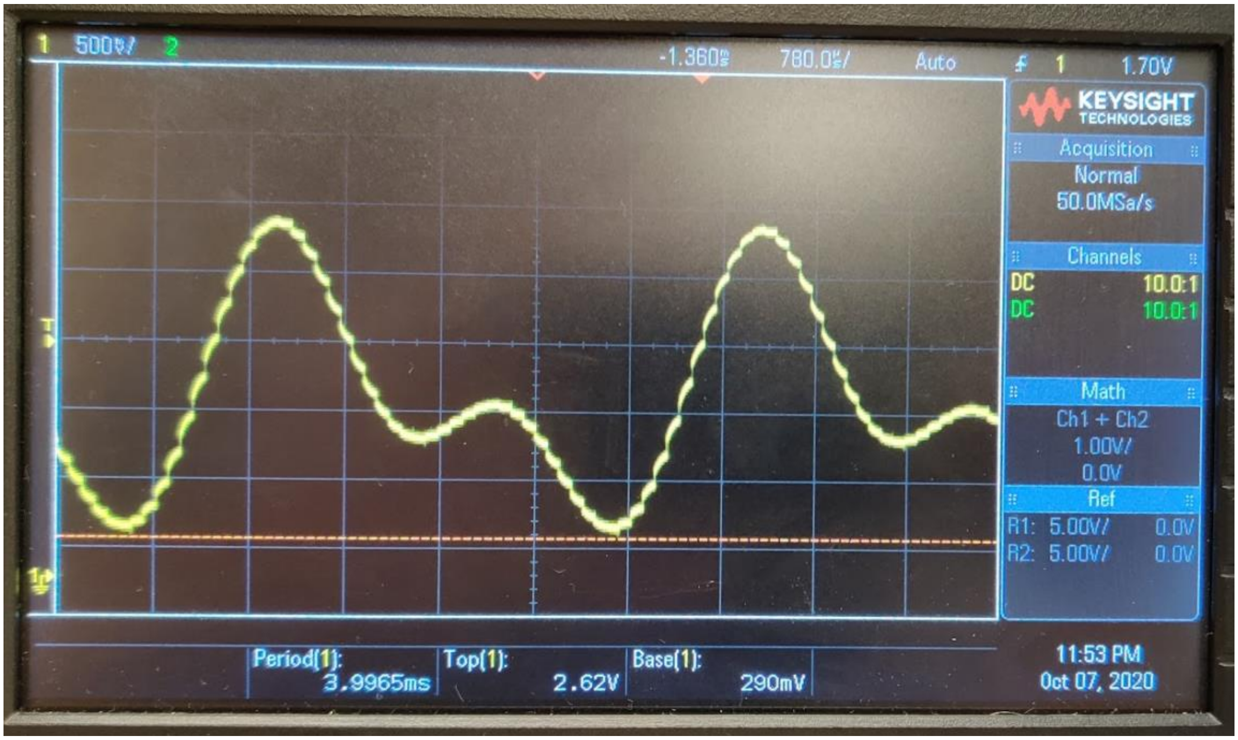
G) Verifications (pictures 😊)

DAC0 toggling K++ and K—buttons (SW2 and SW3)

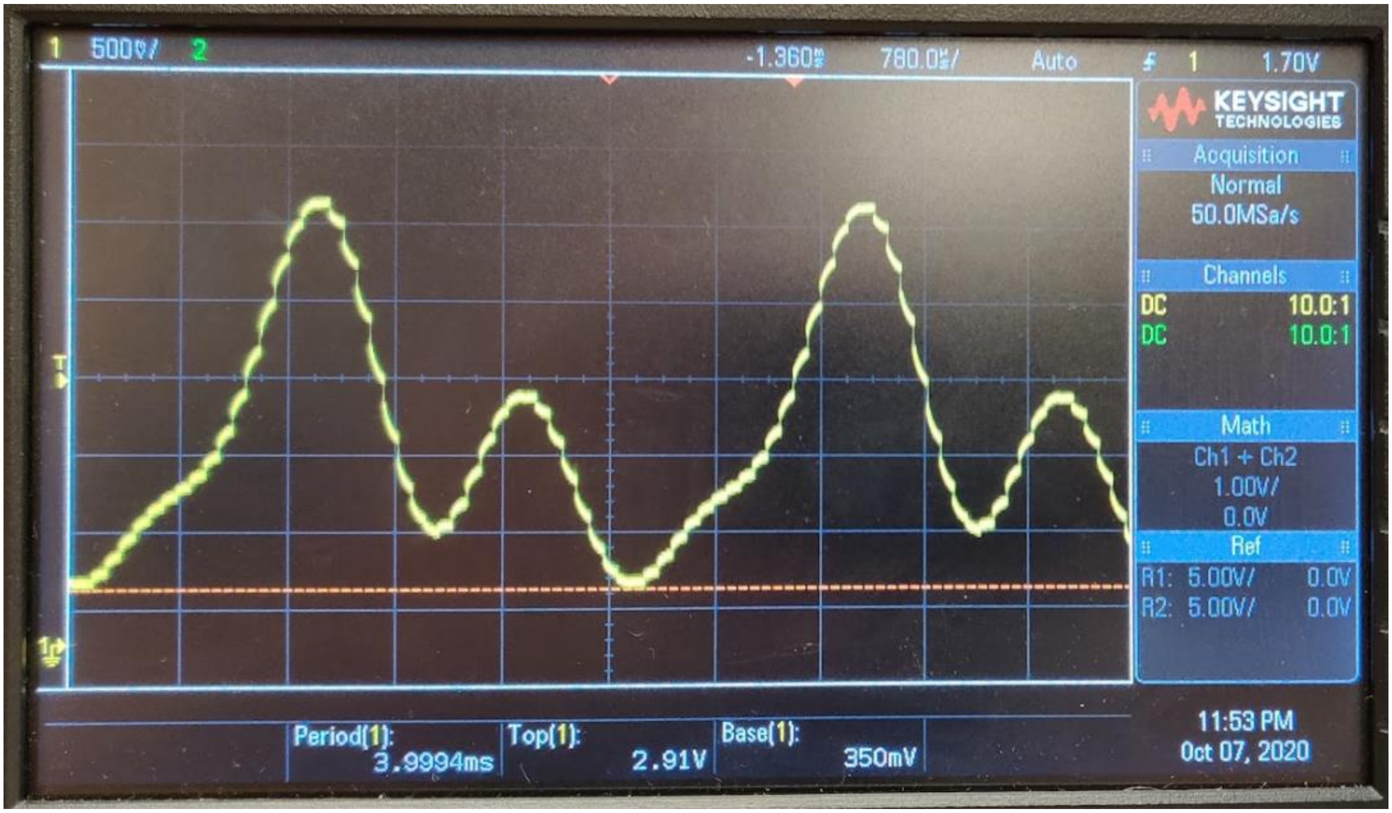
K = 0



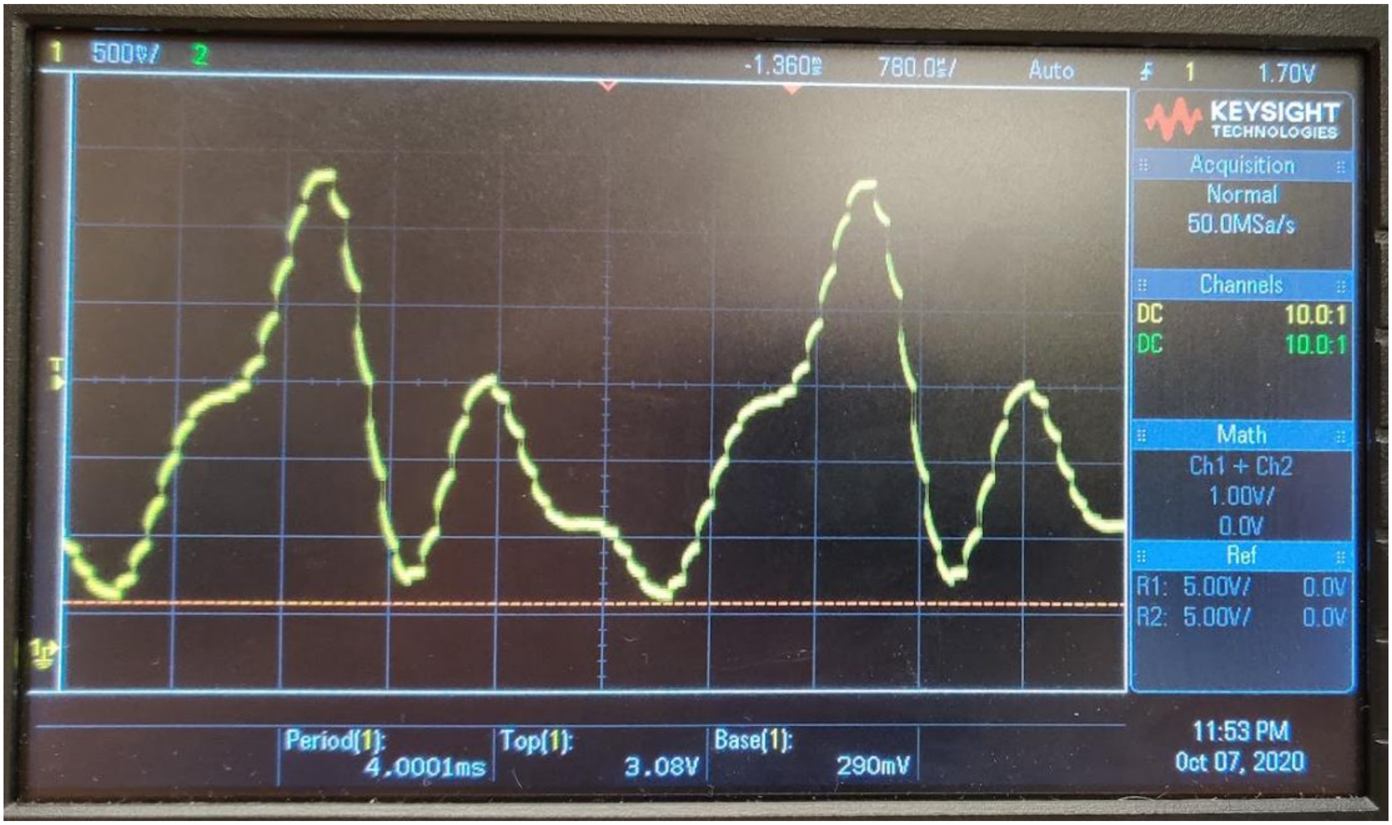
K = 1



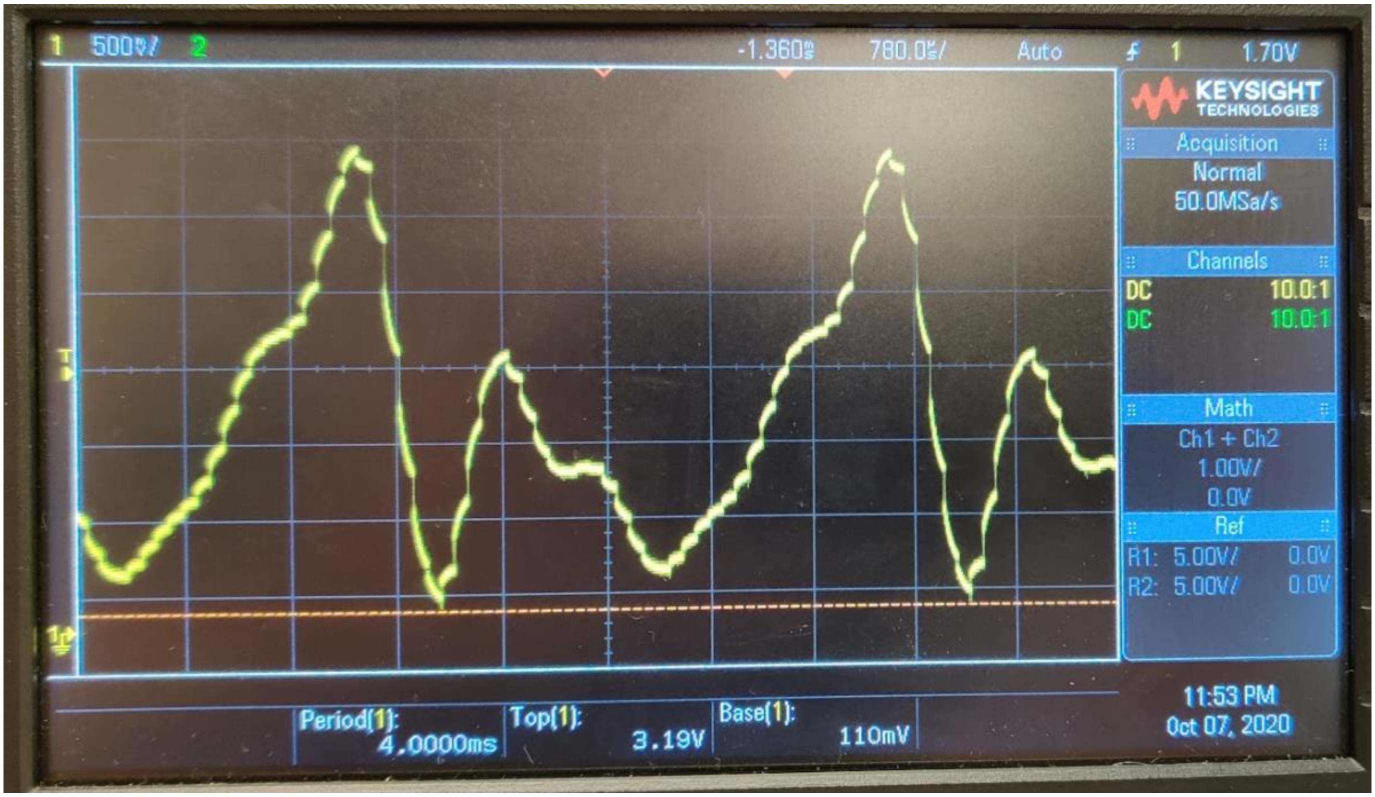
K = 3


1 500mV/ 2
-1.360ms 780.0us/ Auto
1 1.70V
KEYSIGHT TECHNOLOGIES
Acquisition
Normal
50.0MSa/s
Channels
DC 10.0:1
DC 10.0:1
Math
Ch1 + Ch2
1.00V/
0.0V
Ref
R1: 5.00V/ 0.0V
R2: 5.00V/ 0.0V
Period(1): 3.9994ms
Top(1): 2.91V
Base(1): 350mV
11:53 PM
Oct 07, 2020

K = 4


1 500mV/ 2
-1.360ms 780.0us/ Auto
1 1.70V
KEYSIGHT TECHNOLOGIES
Acquisition
Normal
50.0MSa/s
Channels
DC 10.0:1
DC 10.0:1
Math
Ch1 + Ch2
1.00V/
0.0V
Ref
R1: 5.00V/ 0.0V
R2: 5.00V/ 0.0V
Period(1): 4.0001ms
Top(1): 3.08V
Base(1): 290mV
11:53 PM
Oct 07, 2020

K = 5



1 500mV/ 2 -1.360ms 780.0µs/ Auto 1 1.70V

Acquisition
Normal
50.0MSa/s

Channels
DC 10.0:1
DC 10.0:1

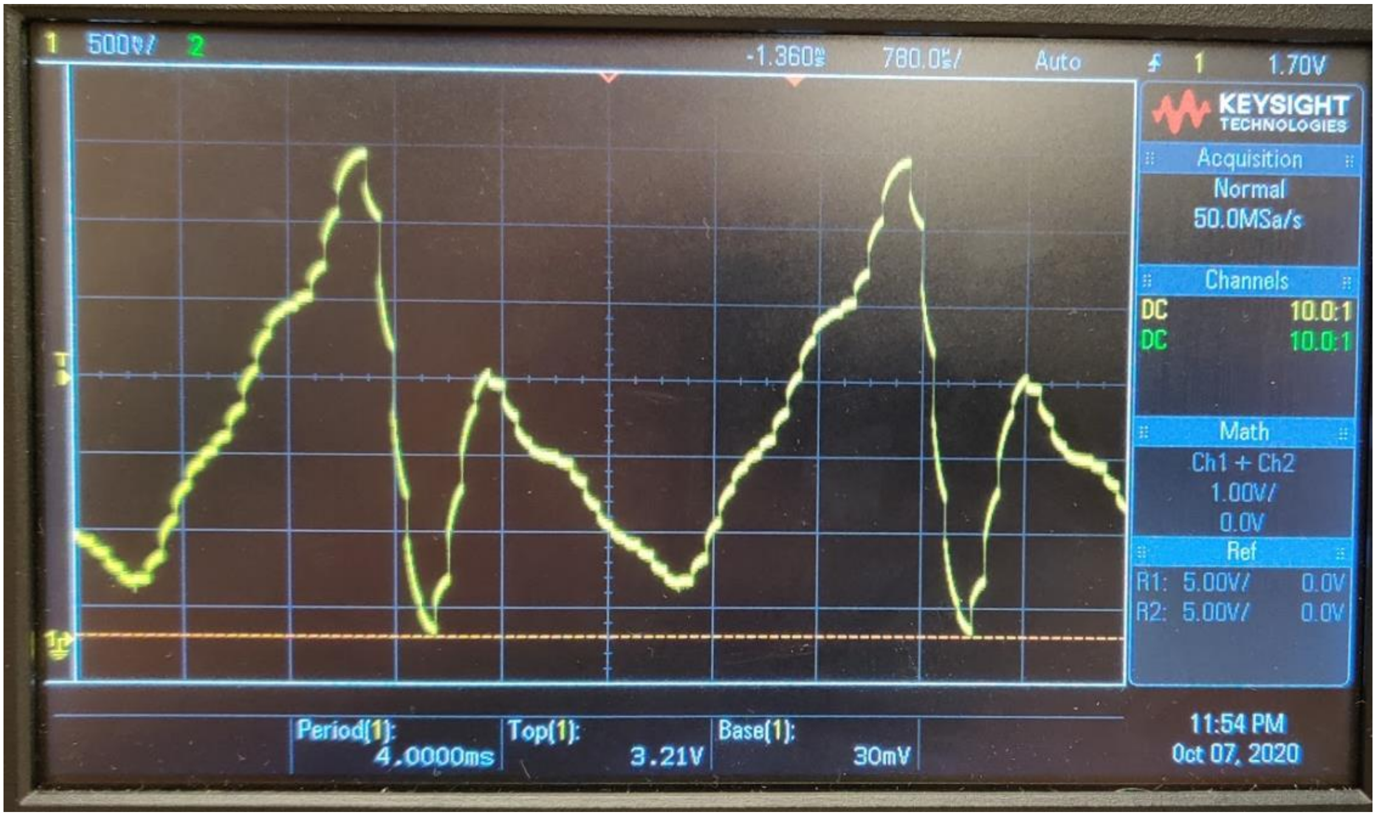
Math
Ch1 + Ch2
1.00V/
0.0V

Ref
R1: 5.00V/ 0.0V
R2: 5.00V/ 0.0V

Period(1): 4.0000ms Top(1): 3.19V Base(1): 110mV

11:53 PM
Oct 07, 2020

K = 6



1 500mV/ 2 -1.360ms 780.0µs/ Auto 1 1.70V

Acquisition
Normal
50.0MSa/s

Channels
DC 10.0:1
DC 10.0:1

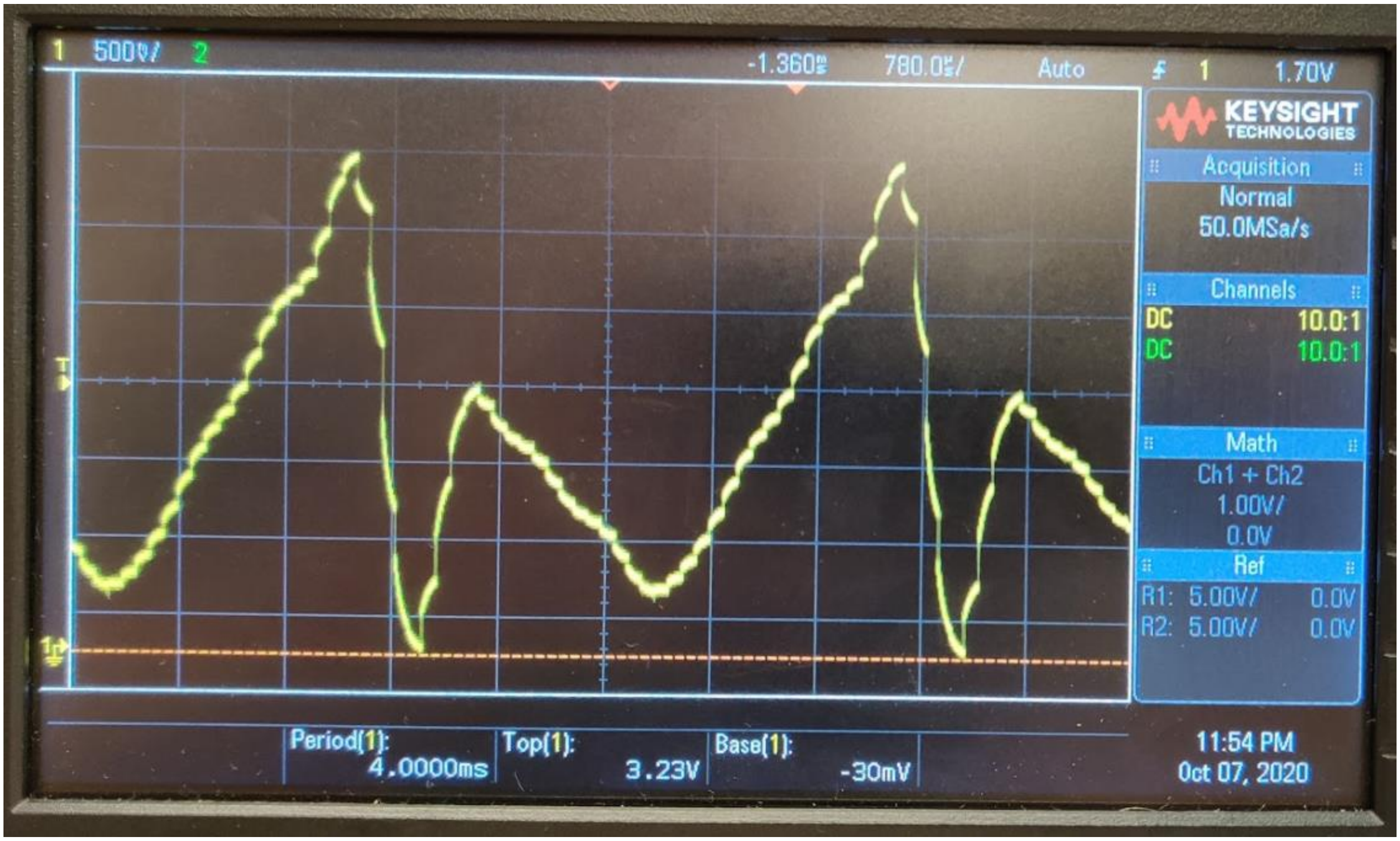
Math
Ch1 + Ch2
1.00V/
0.0V

Ref
R1: 5.00V/ 0.0V
R2: 5.00V/ 0.0V

Period(1): 4.0000ms Top(1): 3.21V Base(1): 30mV

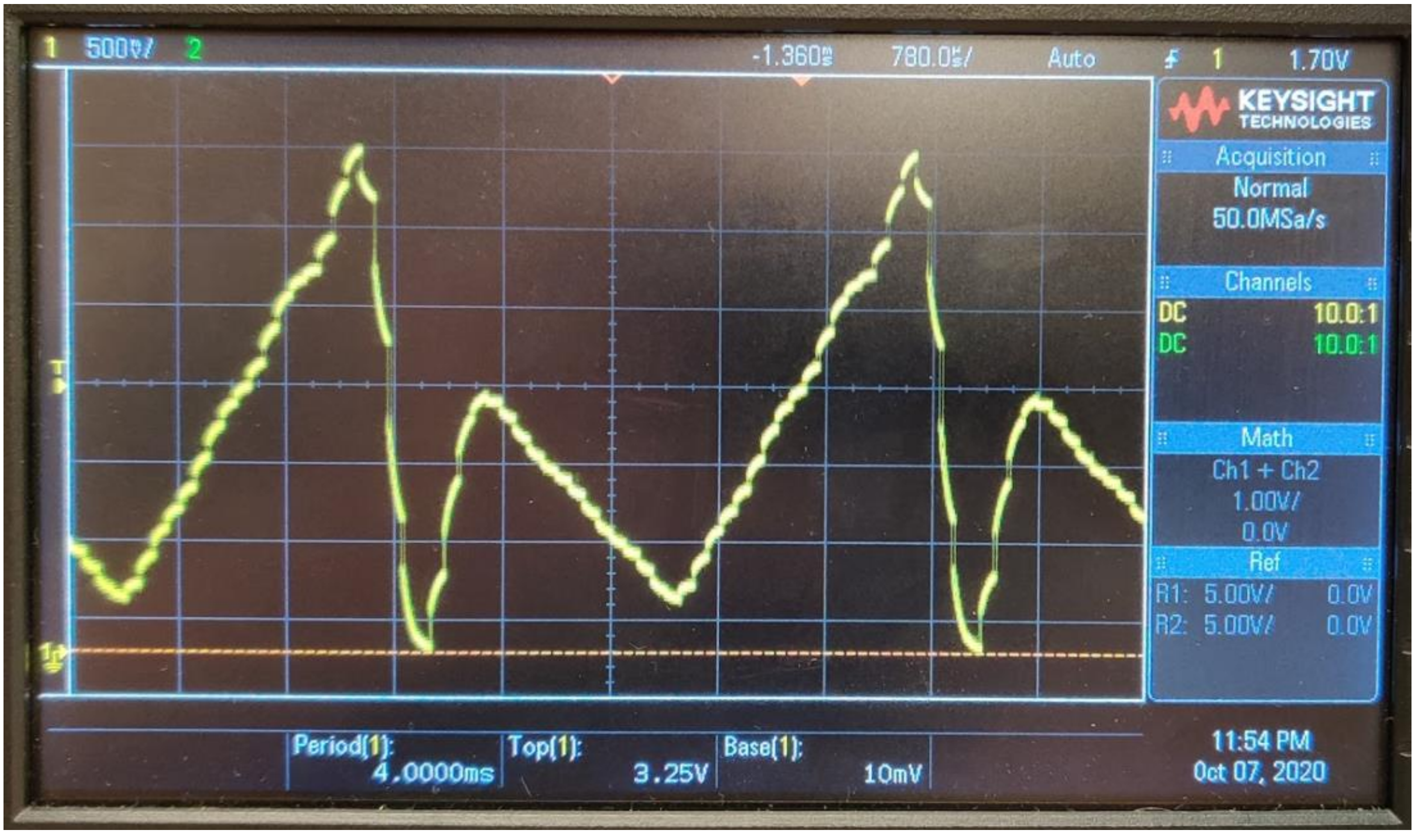
11:54 PM
Oct 07, 2020

K = 7



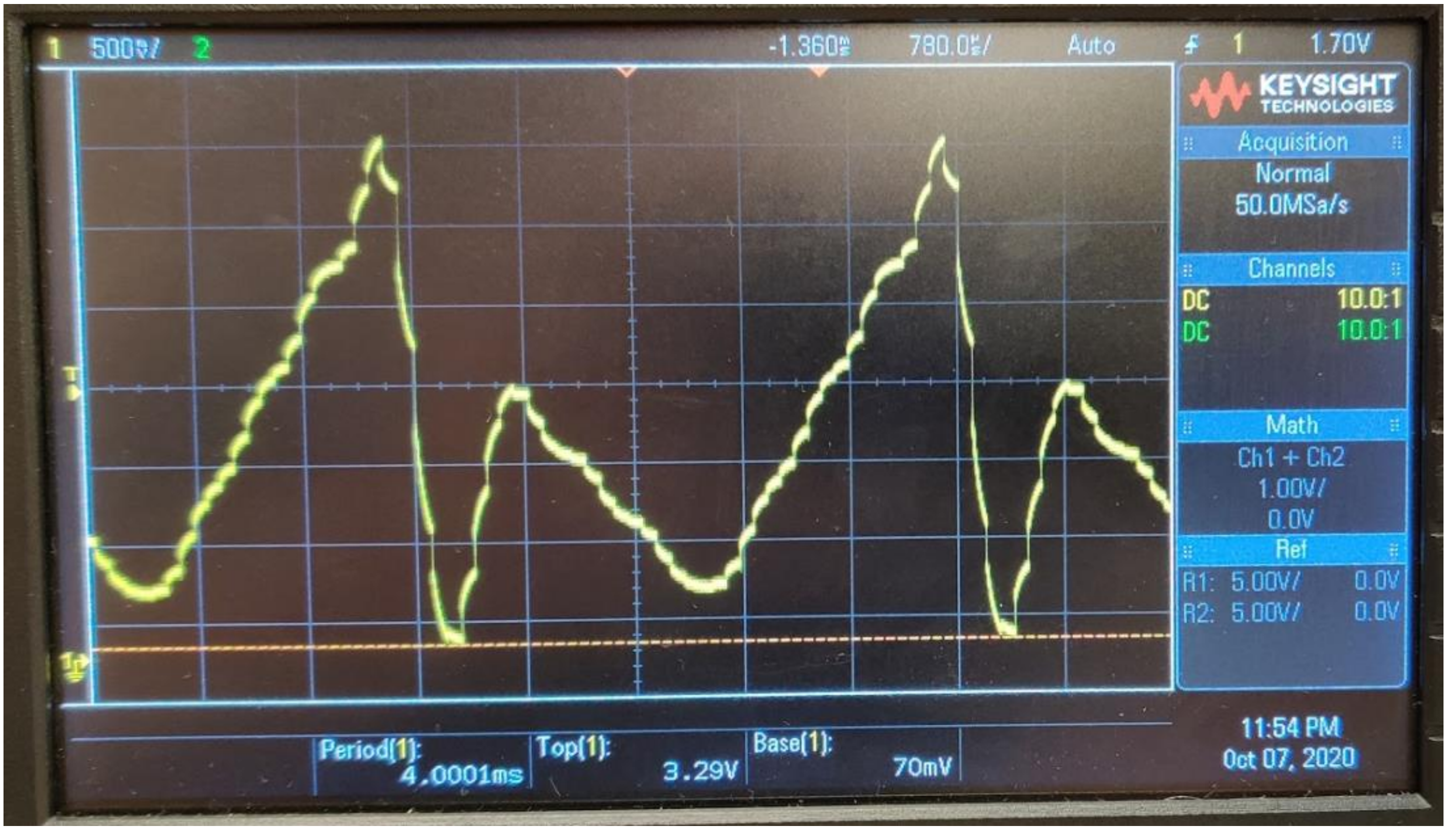
1 500mV/ 2
-1.360ms 780.0us/ Auto
1 1.70V
KEYSIGHT TECHNOLOGIES
Acquisition
Normal
50.0MSa/s
Channels
DC 10.0:1
DC 10.0:1
Math
Ch1 + Ch2
1.00V/
0.0V
Ref
R1: 5.00V/ 0.0V
R2: 5.00V/ 0.0V
Period(1): 4.0000ms
Top(1): 3.23V
Base(1): -30mV
11:54 PM
Oct 07, 2020

K = 8

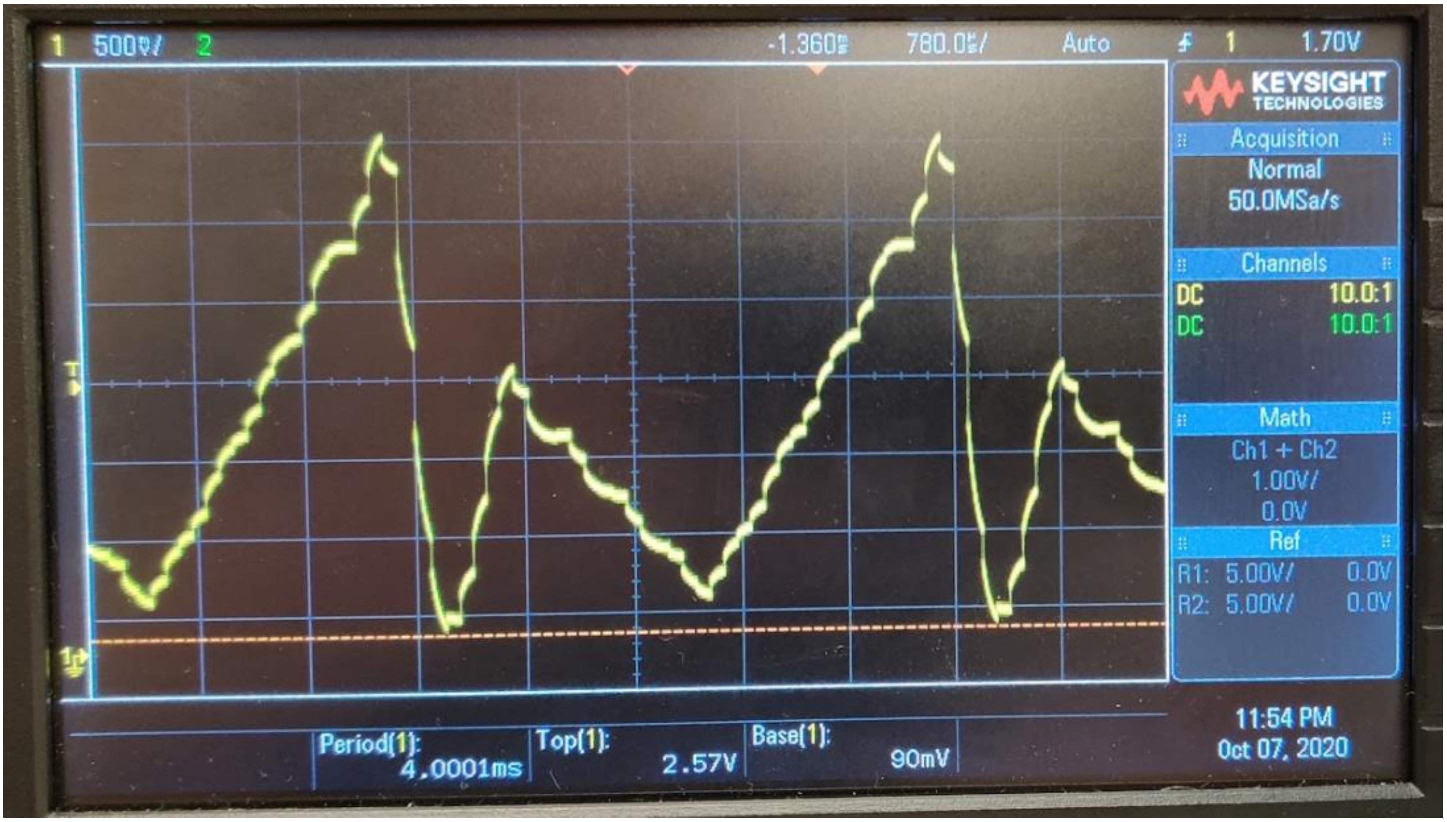


1 500mV/ 2
-1.360ms 780.0us/ Auto
1 1.70V
KEYSIGHT TECHNOLOGIES
Acquisition
Normal
50.0MSa/s
Channels
DC 10.0:1
DC 10.0:1
Math
Ch1 + Ch2
1.00V/
0.0V
Ref
R1: 5.00V/ 0.0V
R2: 5.00V/ 0.0V
Period(1): 4.0000ms
Top(1): 3.25V
Base(1): 10mV
11:54 PM
Oct 07, 2020

K = 9

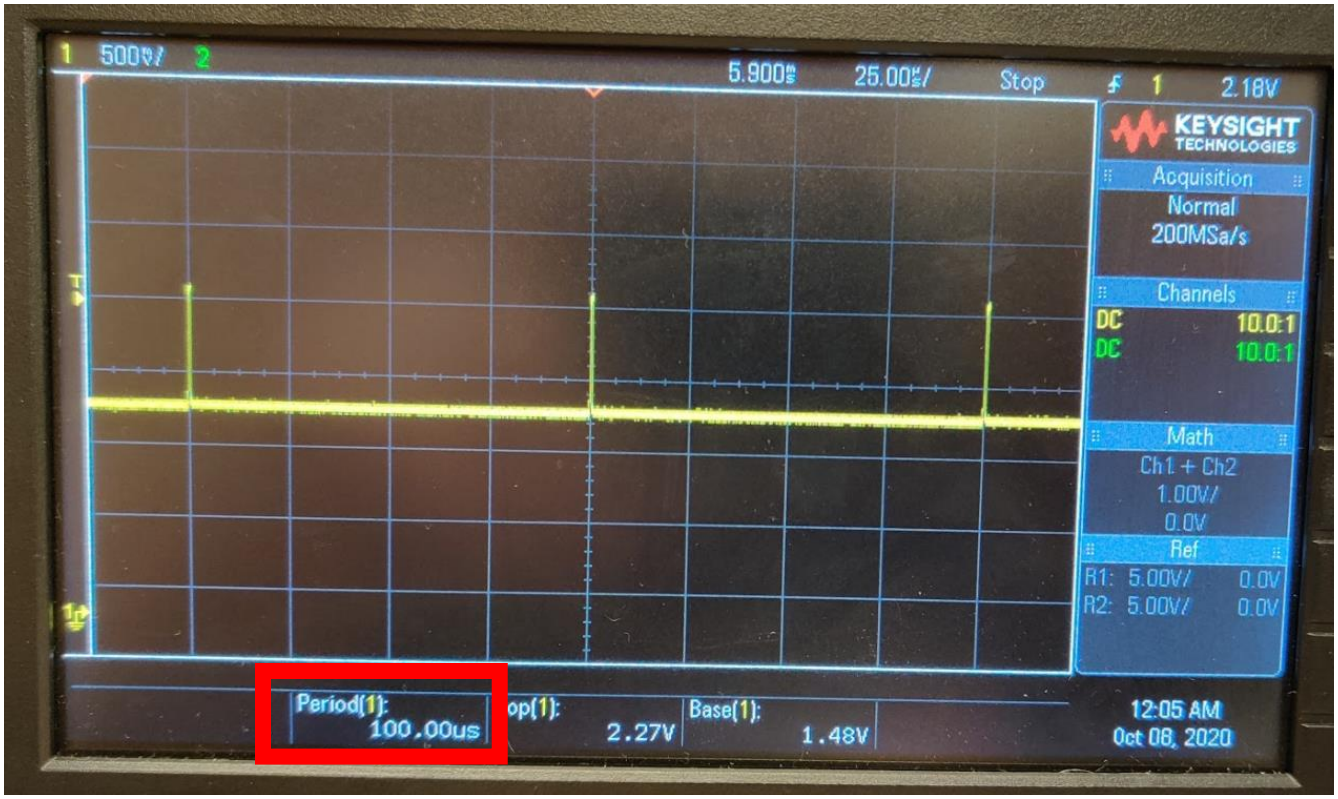

1 500mV/ 2
-1.360ms 780.0us/ Auto
1 1.70V
KEYSIGHT TECHNOLOGIES
Acquisition
Normal
50.0MSa/s
Channels
DC 10.0:1
DC 10.0:1
Math
Ch1 + Ch2
1.00V/
0.0V
Ref
R1: 5.00V/ 0.0V
R2: 5.00V/ 0.0V
Period(1): 4.0001ms
Top(1): 3.29V
Base(1): 70mV
11:54 PM
Oct 07, 2020

K = 10


1 500mV/ 2
-1.360ms 780.0us/ Auto
1 1.70V
KEYSIGHT TECHNOLOGIES
Acquisition
Normal
50.0MSa/s
Channels
DC 10.0:1
DC 10.0:1
Math
Ch1 + Ch2
1.00V/
0.0V
Ref
R1: 5.00V/ 0.0V
R2: 5.00V/ 0.0V
Period(1): 4.0001ms
Top(1): 2.57V
Base(1): 90mV
11:54 PM
Oct 07, 2020

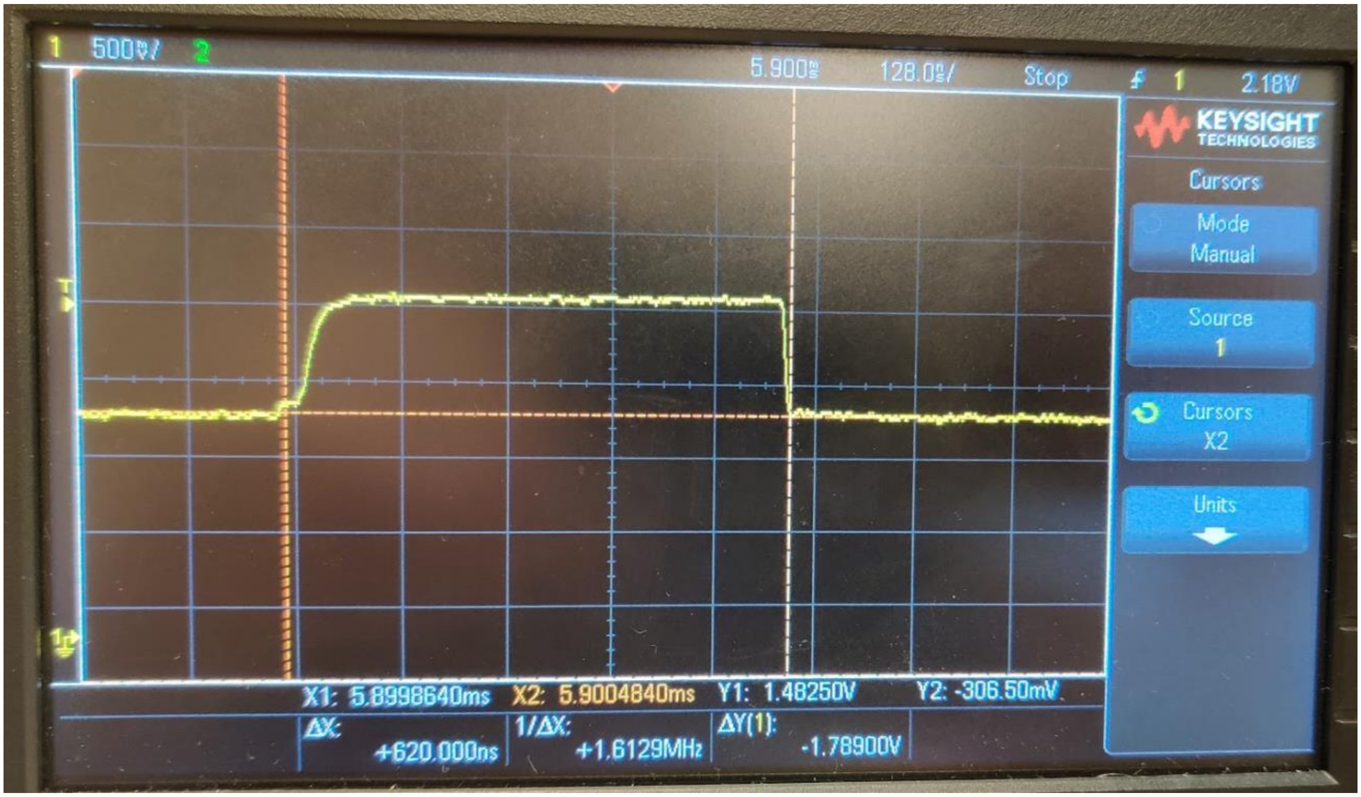
Thomas Smallarz ECE444 Homework 3

RGB LED Output waveform:



We can see the period is correct (interrupt set to 0.1ms → 100us. The LED goes high for just a split second, as that is the time it takes to go through our PIT Interrupt routine.

If we zoom into one of these pulses from the LED, we can see that the main ISR routine takes ~620ns


1 500mV/ 2
5.900ms 128.0ns/ Stop
1 2.18V
KEYSIGHT TECHNOLOGIES
Cursors
Mode
Manual
Source
Cursors
X2
Units
X1: 5.8998640ms X2: 5.9004840ms Y1: 1.48250V Y2: -306.50mV
ΔX: +620.000ns
1/ΔX: +1.6129MHz
ΔY(1): -1.78900V