**Date Submitted: 10/30/2018**

**Assignment Youtube Playlist:** **https://www.youtube.com/playlist?list=PL4oTyvRrubXegms9\_06I4Rup2xGBCZlng**

# Task 01: Submit a comprehensive commented file of the original code.

For Task 01, I have commented on the existing code, note that the predefined symbol of TARGET\_IS\_TM4C123\_RB1 (or valid alternative) needs to be added to allow the rom code to be recognized.

**Youtube Link:** **https://youtu.be/YKZBrKOnSGU**

**Original Code (added comments):**

**#include** <stdint.h>

**#include** <stdbool.h>

**#include** <math.h>

**#include** "inc/hw\_memmap.h"

**#include** "inc/hw\_types.h"

**#include** "driverlib/fpu.h"

**#include** "driverlib/sysctl.h"

**#include** "driverlib/rom.h"

//Include all referenced files.

//If M\_PI is not defined, define it as the below value.

**#ifndef** M\_PI

**#define** M\_PI 3.14159265358979323846

**#endif**

//Define SERIES\_LENGTH, substituting the given value where it occurs.

**#define** SERIES\_LENGTH 100

//Declare gSeriesData as a float array of size SERIES\_LENGTH, defined above.

**float** gSeriesData[SERIES\_LENGTH];

//Create variable to count up from 0 to SERIES\_LENGTH

int32\_t i32DataCount = 0;

**int** **main**(**void**)

{

//Float variable to hold how large each step in the calculation is, in radians.

**float** fRadians;

//Enable lazy stacking and the FPU using the ROM functions.

ROM\_FPULazyStackingEnable();

ROM\_FPUEnable();

//Use ROM function to set system clock, with these settings clock is at 50MHz.

ROM\_SysCtlClockSet(SYSCTL\_SYSDIV\_4 | SYSCTL\_USE\_PLL | SYSCTL\_XTAL\_16MHZ | SYSCTL\_OSC\_MAIN);

//Calculate Fradians as 2pi/length, to calculate each point for one full cycle.

fRadians = ((2 \* M\_PI) / SERIES\_LENGTH);

//Calculate each point, one full cycle divided into SERIES\_LENGTH number of segments.

**while**(i32DataCount < SERIES\_LENGTH)

{

//Calculate SIN function from 0 to 2pi.

gSeriesData[i32DataCount] = **sinf**(fRadians \* i32DataCount);

//Increment step

i32DataCount++;

}

//Once calculation is complete, hold in an endless loop.

**while**(1)

{

}

}

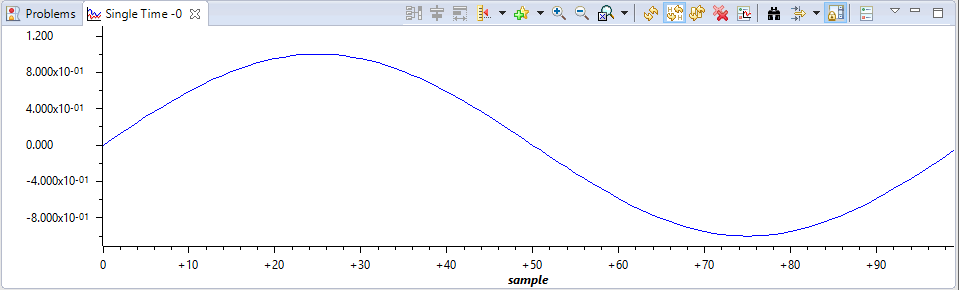


Figure - Our calculated sine wave

**-----------------------------------------------------------------------------------**

# Task 02: Modify the code to implement the below equation with 1000+ sample points to generate a frequency of 50 Hz.

# 1.5 + 1.0\*sin(2**π**50t) + 0.5\*cos(2π200t)

**Youtube Link:** **https://youtu.be/-IPdRC5Aymg**

Note: The class lecture slides and the Lab 9 slides define two different equations.

For Task 02 I have modified the code to calculate the above equation. These data points could be fed to a DAC to closely reproduce the refenced function, depending on the accuracy of the DAC.

**#include** <stdint.h>

**#include** <stdbool.h>

**#include** <math.h>

**#include** "inc/hw\_memmap.h"

**#include** "inc/hw\_types.h"

**#include** "driverlib/fpu.h"

**#include** "driverlib/sysctl.h"

**#include** "driverlib/rom.h"

//Include all referenced files.

//If M\_PI is not defined, define it as the below value.

**#ifndef** M\_PI

**#define** M\_PI 3.14159265358979323846

**#endif**

//Define SERIES\_LENGTH, substituting the given value where it occurs.

**#define** SERIES\_LENGTH 5000

//Declare gSeriesData as a float array of size SERIES\_LENGTH, defined above.

**float** gSeriesData[SERIES\_LENGTH];

//Create variable to count up from 0 to SERIES\_LENGTH

int32\_t i32DataCount = 0;

**int** **main**(**void**)

{

//Float variable to hold how large each step in the calculation is, in radians.

**float** fRadians;

//Enable lazy stacking and the FPU using the ROM functions.

ROM\_FPULazyStackingEnable();

ROM\_FPUEnable();

//Use ROM function to set system clock, with these settings clock is at 50MHz.

ROM\_SysCtlClockSet(SYSCTL\_SYSDIV\_4 | SYSCTL\_USE\_PLL | SYSCTL\_XTAL\_16MHZ | SYSCTL\_OSC\_MAIN);

//Calculate Fradians as 2pi/length, to calculate each point for one full cycle.

fRadians = ((2 \* M\_PI) / SERIES\_LENGTH);

//Calculate each point, one full cycle divided into SERIES\_LENGTH number of segments.

**while**(i32DataCount < SERIES\_LENGTH)

{

//Calculate the defined function for each segment.

gSeriesData[i32DataCount] = 1.5 + **sinf**(fRadians \* i32DataCount \* 50) + 0.5 \* **cosf**(fRadians \* i32DataCount \* 200);

//Increment step

i32DataCount++;

}

//Once calculation is complete, hold in an endless loop.

**while**(1)

{

}

}

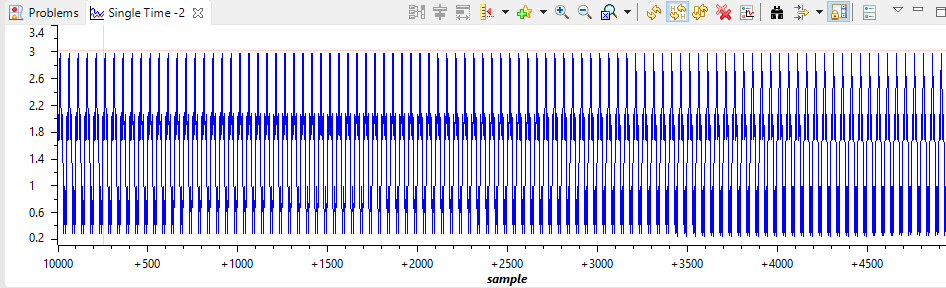


Figure - 5000 samples of 50Hz signal (200Hz signal modulated)

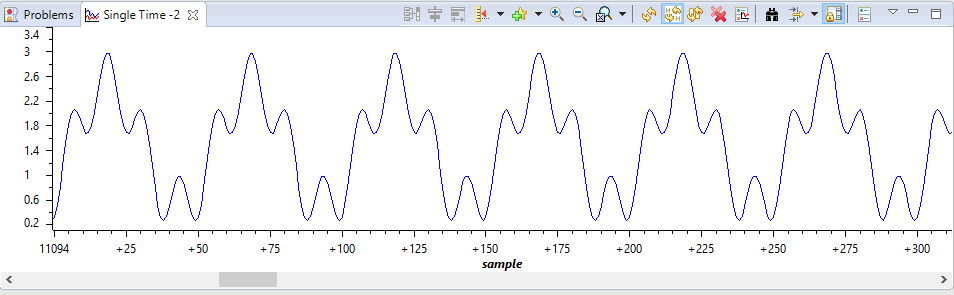


Figure - Zoomed portion of 50Hz/200Hz signals

**-----------------------------------------------------------------------------------**