**Date Submitted: 10/23/19**

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

**Task 01:**

Youtube Link: <https://www.youtube.com/watch?v=ap84pSm2nBE&list=PLLbVEP8QAFUFxdqWNafGkmwVemtXHsc3k&index=1>

**Modified Schematic (if applicable):**

**N/A**

**Modified Code:**

//libraries :)

#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"

#ifndef M\_PI

#define M\_PI 3.14159265358979323846 //defines PI's float variable

#endif

#define SERIES\_LENGTH 100 //for our 100 data point long array

float gSeriesData[SERIES\_LENGTH]; //contains our 100 data point values

int32\_t i32DataCount = 0; //counter variable as a limiter

int main(void)

{

float fRadians; //float radians variable

ROM\_FPULazyStackingEnable(); //Enables the lazy stacking of floating-point registers. If a floating-point

//instruction is executed from within the interrupt context, the floating-point context

//is first saved into the space reserved on the stack.

//On completion of the interrupt handler, the floating-point context is only restored

//if it was saved (as the result of executing a floating-point instruction).

ROM\_FPUEnable(); //Enables the floating-point unit.

//This function enables the floating-point unit, allowing the floating-point

//instructions to be executed.

ROM\_SysCtlClockSet(SYSCTL\_SYSDIV\_4 | SYSCTL\_USE\_PLL | SYSCTL\_XTAL\_16MHZ | SYSCTL\_OSC\_MAIN); //Sets system clock at 50MHz

fRadians = ((2 \* M\_PI) / SERIES\_LENGTH); //Calculates for the variable to hold the value 2pi/depth of data buffer

while(i32DataCount < SERIES\_LENGTH) //Calculates the sine wave values for each of the 100 values

{

//Calculates the sine wave values

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

//Continues to count till it reaches the 100th value

i32DataCount++;

}

while(1)

{

//main loop

}

}

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

**Task 02:**

Youtube Link: <https://www.youtube.com/watch?v=uCAcMqG-vcY&list=PLLbVEP8QAFUFxdqWNafGkmwVemtXHsc3k&index=2>

**Modified Schematic (if applicable):**

**Modified Code:**

#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 "utils/uartstdio.h"

#include "driverlib/uart.h"

#include "driverlib/pin\_map.h"

#include "inc/hw\_gpio.h"

#include "driverlib/gpio.h"

#ifndef M\_PI

#define M\_PI 3.14159265358979323846 //defines PI's float variable

#endif

#define SERIES\_LENGTH 1000 //for our 1000 data point long array

float gSeriesData[SERIES\_LENGTH]; //contains our 1000 data point values

int32\_t i32DataCount = 0; //counter variable as a limiter

void InitUART(void)

{

SysCtlPeripheralEnable(SYSCTL\_PERIPH\_UART0); //enables UART module 0

SysCtlPeripheralEnable(SYSCTL\_PERIPH\_GPIOA); //enables GPIO port a

GPIOPinConfigure(GPIO\_PA0\_U0RX); //PA0 as RX pin

GPIOPinConfigure(GPIO\_PA1\_U0TX); //PA1 as TX pin

GPIOPinTypeUART(GPIO\_PORTA\_BASE, GPIO\_PIN\_0 | GPIO\_PIN\_1); //sets UART pin type

UARTClockSourceSet(UART0\_BASE, UART\_CLOCK\_PIOSC); //sets the clock source

UARTStdioConfig(0, 115200, 16000000); //disables uartstdio, sets baud rate to 115200, uses clock

}

int main(void)

{

float fRadians\_sin, fRadians\_cos; //float radians variable

ROM\_FPULazyStackingEnable(); //Enables the lazy stacking of floating-point registers. If a floating-point

//instruction is executed from within the interrupt context, the floating-point context

//is first saved into the space reserved on the stack.

//On completion of the interrupt handler, the floating-point context is only restored

//if it was saved (as the result of executing a floating-point instruction).

ROM\_FPUEnable(); //Enables the floating-point unit.

//This function enables the floating-point unit, allowing the floating-point

//instructions to be executed.

ROM\_SysCtlClockSet(SYSCTL\_SYSDIV\_4 | SYSCTL\_USE\_PLL | SYSCTL\_XTAL\_16MHZ | SYSCTL\_OSC\_MAIN); //Sets system clock at 50MHz

fRadians\_sin = ((2 \* M\_PI \* 50) / SERIES\_LENGTH); //Calculates for the variable to hold the value 2pi/depth of data buffer

fRadians\_cos = ((2 \* M\_PI \* 200) / SERIES\_LENGTH); //Calculates for the variable to hold the value 2pi/depth of data buffer

InitUART();

while(i32DataCount < SERIES\_LENGTH) //Calculates the sine wave values for each of the 1000 values

{

//Calculates the sine wave values

//sin(2pi\*50t) + 0.5cos(2pi\*200t)

gSeriesData[i32DataCount] = (1.0 \* sinf(fRadians\_sin \* i32DataCount)) + (0.5 \* cosf(fRadians\_cos \* i32DataCount)); //given equation

//print equation to terminal

UARTprintf("gSeriesData[%d] = sin(2PI \* 50 \* %d) + 0.5cos(2PI \* 200 \* %d)\n",

i32DataCount, i32DataCount, i32DataCount);

//Continues to count till it reaches the 100th value

i32DataCount++;

SysCtlDelay(10000000);

}

while(1)

{

//main loop

}

}

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