Abenezer Namaga
CPE 403 – 1001
TIVAC LAB9

Github root directory: https://github.com/Ayertena/AdvEmbededSys

Date Submitted: 10/16/2018

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

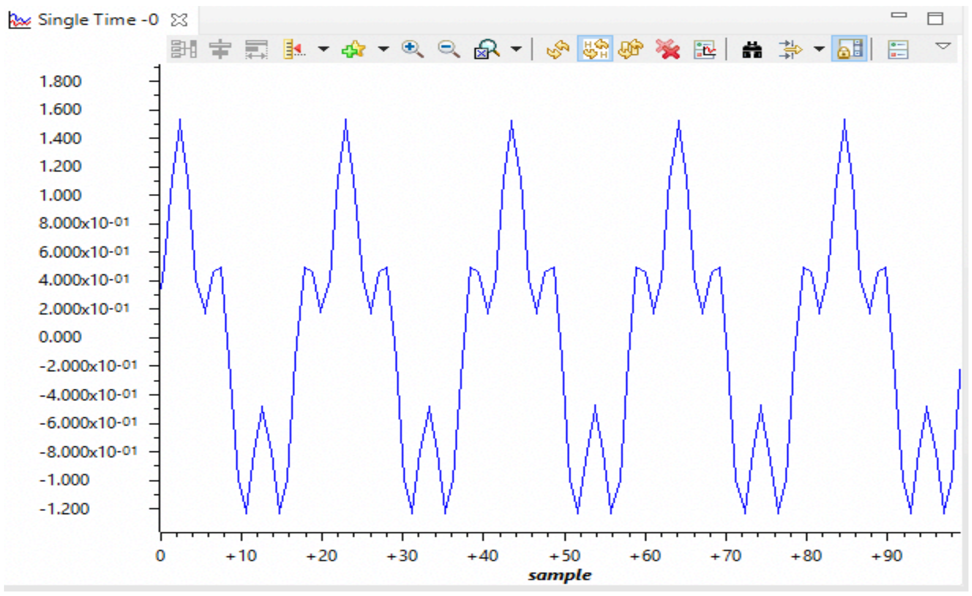
```
#include <stdint.h>  // Variable definitions for the C99 standard
#include <stdbool.h> // Boolean definitions for the C99 standard
#include <math.h>    // Provides access to sinf()
#include "inc/hw_memmap.h" // Macros defining the memory map of the
Tiva C Series device.
// This includes defines such as peripheral base address locations
// such as GPIO_PORTF_BASE.
#include "inc/hw_types.h"    // Defines common types and macros
#include "driverlib/fpu.h"   // Provides access floating point values
#include "driverlib/sysctl.h" // Defines and macros for System Control
API of DriverLib.
#include "driverlib/rom.h" // ROM functions

#ifndef M_PI
#define M_PI 3.14159265358979323846 // Value of pi
#endif
#define SERIES_LENGTH 100 // Amount to calculate
float gSeriesData[SERIES_LENGTH]; // Stores the evaluated value
int32_t i32DataCount = 0; // Keeps count # of calculations made

int main(void)
{
     float fRadians; // Declare variables to hold 2pi / 100
ROM_FPULazyStackingEnable(); // Enable lazy stacking
ROM_FPUEnable(); // Enable FPU

// 50 MHz clock
ROM_SysCtlClockSet(SYSCTL_SYSDIV_4 | SYSCTL_USE_PLL |
SYSCTL_XTAL_16MHZ | SYSCTL_OSC_MAIN);

fRadians = ((2 * M_PI) / SERIES_LENGTH); // 2pi / 100

while(i32DataCount < SERIES_LENGTH) {  // 100 calculations are made
gSeriesData[i32DataCount] = sinf(fRadians * i32DataCount); // Get sine
value
i32DataCount++; } // Increment to obtain next sine value
while(1){} }
```

Task 02: Modify the code to implement the below equation to generate a frequency of 5 Hz. Display the equation for 1 sec.
1.0*sin(2p50t) + 0.5*cos(2p 200t)

Youtube Link:

```
#include <stdint.h>   // Variable definitions for the C99 standard
#include <stdbool.h> // Boolean definitions for the C99 standard
#include <math.h>    // Provides access to sinf()
#include "inc/hw_memmap.h" // Macros defining the memory map of the
Tiva C Series device.
// This includes defines such as peripheral base address locations
// such as GPIO_PORTF_BASE.
#include "inc/hw_types.h"    // Defines common types and macros
#include "driverlib/fpu.h"   // Provides access floating point values
#include "driverlib/sysctl.h" // Defines and macros for System Control
API of DriverLib.
#include "driverlib/rom.h" // ROM functions

#ifndef M_PI
#define M_PI 3.14159265358979323846 // Value of pi
#endif
#define SERIES_LENGTH 100 // Amount to calculate
float gSeriesData[SERIES_LENGTH]; // Stores the evaluated value
int32_t i32DataCount = 0; // Keeps count # of calculations made

int main(void)
{
     float fRadians; // Declare variables to hold 2pi / 100
ROM_FPULazyStackingEnable(); // Enable lazy stacking
ROM_FPUEnable(); // Enable FPU

// 50 MHz clock
ROM_SysCtlClockSet(SYSCTL_SYSDIV_4 | SYSCTL_USE_PLL |
SYSCTL_XTAL_16MHZ | SYSCTL_OSC_MAIN);
fRadians = ((2 * M_PI) / SERIES_LENGTH); // 2pi / 100

while(i32DataCount < SERIES_LENGTH) {  // 100 calculations are made

        // 1.0*sin(2p50t) + 0.5*cos(2p 200t)
         // Get sine value

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

i32DataCount++; // Increment to obtain next sine value }

while(1) // Continue for ever

{
 }

}
```

Waveform


Single Time -0
8.000x10-01
6.000x10-01
4.000x10-01
2.000x10-01
-2.000x10-01
-4.000x10-01
-6.000x10-01
-8.000x10-01
sample