Microprocessor Techniques Report

2303ENG

Digital Storage Oscilloscope

By Joseph Simeon – s2966176

Terms of Reference

Digital Storage Oscilloscope using a Tiva C Series LaunchPad.

Joseph Simeon

2966176

18th September 2017

A report submitted in fulfilment of the requirements for Course 2303EG, Griffith School of Engineering, Griffith University.

Summary

This report will cover the creation, programming and implementation of a Digital Storage Oscilloscope (DSO) using a Tiva C Series TM4C123G LaunchPad Evaluation Board. Features that will be used on the board such as ADC, TIMER, and UART will be explained as well as hardware requirements and how those hardware requirements will be connected via a circuit diagram, also showing is how the system communicates with each other in the form of a block diagram. The project will contain complex algorithms that will be broken down in to easily digestible flow charts that can be followed by even the non-programming savvy of readers, the project was completed after facing hurdles which can be viewed in the discussion while only one problem exists which is the ADC halving the input values of the analogue signal that is being tested on the Ain0 pin, with excess time outside of the desired time-frame the problem could be solved. The project had been completed with the simplified project design however with an excess of another week the code could be revaluated to the complex design and a few more days added to create an enclosure.

Table of Contents

Introduction

Tiva C Series LaunchPad (TM4C123G) is an in-expensive single-board microcontroller evaluation platform for the ARM Cortex-M4F created by the company Texas Instruments. This project will be using Texas Instruments Tiva C Series microcontroller in creating a Digital Storage Oscilloscope (DSO); DSO is an electronic test instrument that allows to observe signal voltages, it is analysing the input signals digitally instead of analogue displaying the results in graphic form utilising an x-y co-ordinate system approach to data distribution.

The aim of this project is to create a digital oscilloscope using the Tiva C Series LaunchPad programmed specifically with embedded C-Language within a time-frame of 25th August 2017 to 21st September 2017. The digital oscilloscope will be displayed in graphical form in the terminal using special ANSI escape characters, as well as the parameters will be controlled via the terminal using certain keyboard buttons corresponding with actions such as:

Key	Action
A	Increase the vertical scale by 0.5 V
Z	Decrease the vertical scale by 0.5 V
<	Decrease the sample rate by 0.5
>	Increase the sample rate by 0.5
S	Specify the sample rate
D	Take a single voltage measurement and write the value to the terminal. (DC mode)
V	Specify the voltage range to be displayed (V-min & V-max)
#	Take a single measurement
*	Take a repeating measurement for 5 seconds

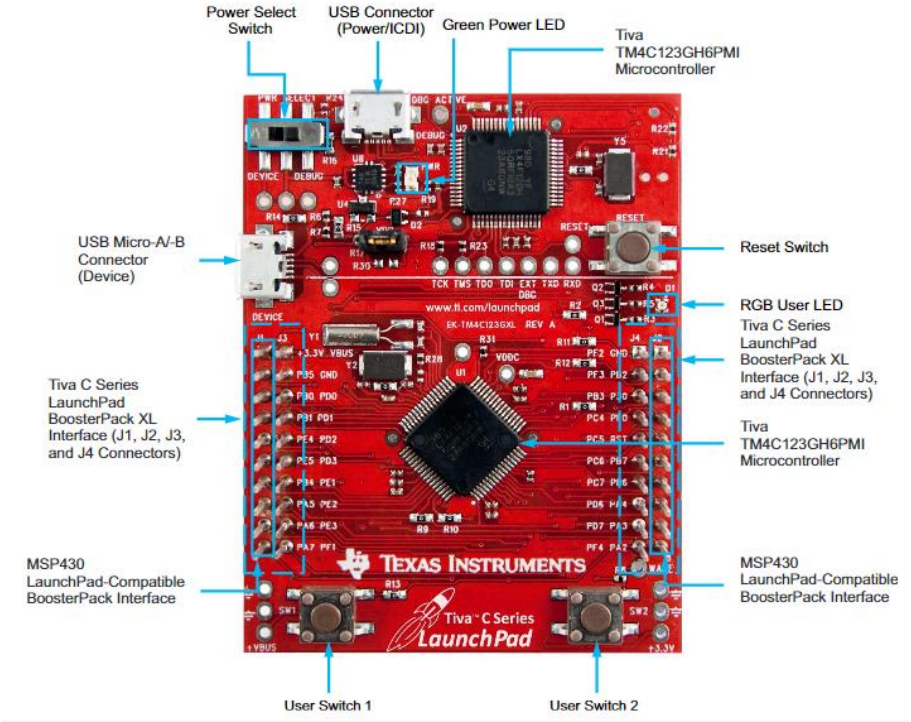
A graphical representation will be displayed on the computer using the terminal which will be where these actions can be implemented. The short-term goal is a minimum graphical design of the graph that will be displayed, where-as a long-term goal would be a deeper design of the graph and friendlier visual in terms of analysing the data presented as well as incorporation of a physical design such as a user-friendly enclosure.

The project's greatest challenge is the setup of the ADC to receive data from the pin and relating the readings over a period into graphical representation using ANSI within the terminal, the more user friendly and better design requires more in-depth knowledge to code a better interface to interact with the user, the physical designing of the connection that will attach voltage signal to wires that will connect the to the observing pin will be minimal first and later if time allows will create a better design to incorporate user friendly interaction.

Methods

Features

Board specific features



Tiva C Series LaunchPad (TM4C123G) Evaluation Board

Image reference "*Tiva C Series TM4C123G LaunchPad Evaluation Board – User's Guide* by Texas Instrument, 2013".

- Tiva TM4C123GH6PMI microcontroller – 32-bit ARM Cortex M4-based microcontroller
- Motion control PWM
- 256 kB Flash memory, 32 kB SRAM, and 80MHz operation
- 16 MHz crystal – microcontroller main internal clock
    - External 32.768 kHz crystal – for hibernation module
    - Internal PLL – for frequencies up to 80 MHz
- USB 2.0 device interface with micro-USB connector
- RGB LED
- 2 programmable user switches/buttons
- I/O available via headers
- On-board Tiva In-Circuit Debug Interface (ICDI)
- Switch-selectable power sources (USB, ICDI)
- Reset switch
- Use of ARM Thumb-compatible Thumb-2 instruction set

(David Rowlands, 2017)

Utilised features

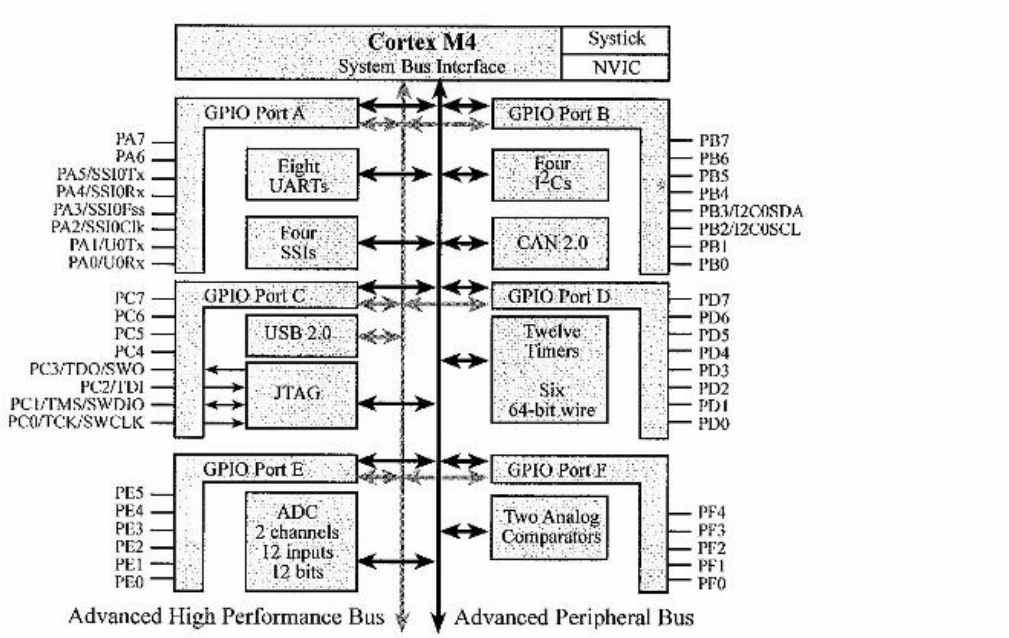


Figure 4.6. I/O port pins for the LM4F120H5QR microcontroller.

Valvano, 2012, Vol.1

Features of the board represented by the different subsystems that exist.

Image reference: "*Embedded systems: Introduction to the ARM Cortex™-M microcontrollers, Volume 1* by Jonathan Valvano, 2012".

The subsystems that will be used in the project of creating a DSO will be:

- UART0 activated on pins A [0, 1]
- ADC0 activated on pin E [3]
- TIMER0 activated via ADC

UART stands for Universal Asynchronous Receiver-Transmitter, the subsystem will be used for serial communication between the board and the computer through a terminal, transmitting and receiving data – this project will be using UART0 which uses General Purpose Input-Output (GPIO) Port A, Pin 0 & Pin 1.

ADC stands for Analogue-Digital Converter, the subsystem that uses an electronic process to receive varying signal in voltage (analogue) and converts the signal to a digital level. The digital levels are determined by how many bits the analogue signal will be digested into, such as if the signal is being digested by 12-bit than the digital level will be between the limits of 0 to 4095, where 4095 would reflect 3.3 Volts (V) because the limits of the Tiva is between its lowest voltage of Ground (GND)\0 V and Supply voltage (Vcc) \ 3.3 V analogue signals that move below or above the limited voltages will be truncated to fit within the boards limitation. The project will be using ADC0 which uses GPIO Port E, Pin 3.

General Purpose Timers (TIMER) is a subsystem of programmable timers that are connected to GPIO pins, it will count down and trigger the ADC once zero is reached, this will relate to a frequency as which the ADC is being triggered. The project will be using TIMER0 which will be activated on the ADC Pin and will connect to the ADC triggering.

Hardware

Two set ups were used as the items used in the laboratory could not be replicated exactly outside the lab when the project was worked on during non-laboratory allocated time. The list of hardware during these times are:

General Equipment

The general equipment will be used in laboratory and non-laboratory testing.

- Tiva C Series TM4C123G LaunchPad
- Breadboard
- Female to Male or Male to Male pins

Laboratory Specific

The laboratory equipment will be used only in the laboratory – an older model Oscilloscope (DSO/CRO) which does not have a wave generator will need a function generator where-as later models will have this feature.

- Function generator
- Digital Storage Oscilloscope

Non-laboratory Specific

The non-laboratory equipment will be used to test the project outside of laboratory allocated time.

- Potentiometer (Varying resistor)

Design

Complex algorithms

The most complex parts of this experiment are the setting up of the UART and ADC with TIMER along with each subsystem's subtasks and manipulating the data received from the ADC's Analogue Input Pin to feed into specific functions such as creating the DSO graph as well as signal information that can be accessed before the calling of the graph output. Main algorithms:

- Subsystems with off-task
- Manipulating data
    - Signal information via user interface
    - DSO output graph

Main blocks & sub-tasks & off-tasks

Subsystems that are not being manipulated by the prepared functions by TivaWare will needed to be accessed via direct register set-up using embedded C, even though Assembly (ASM) direct register set-up takes more lines of code, both can yield errors when via destructive masks or not turning the right bits within the register, within the main block of setting up the registers, off-tasks that utilise the set-up will be included though they are not subtasks but are needed tasks that rely on a perfect set-up.

The manipulating of ADC data received will be given to the DSO output graph after it has been manipulated to fit within the graph size, the data received from the ADC will also be manipulated and fed to the user interface that will allow the user to see information before moving onto the output graph.

Subsystem

- UART initialisation
    - GPIO initialisation for UART
    - Receiving characters
    - Transmitting characters
    - Printing strings via UART transmitting of characters
- ADC initialisation
    - GPIO initialisation for ADC
    - TIMER initialisation for ADC
        - Frequency changes to TIMER via preload change
    - Transmitting of data via ADC

Main-code

- User interface
    - Pre-defined set-up for graph using key and actions
    - Manipulating data to export values
- Graphing of signal
    - Manipulating data to represent values

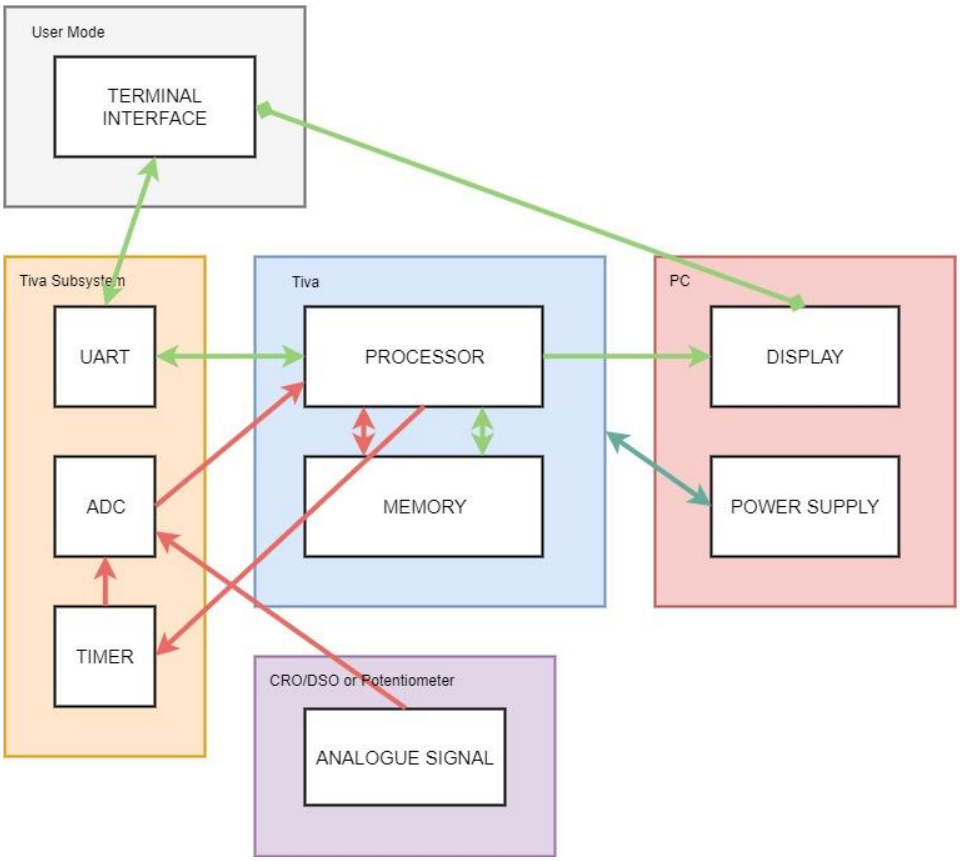
Simplified & complex project design

The simplified design in the project is to create a user interface that allows the person to implement predefined actions before the screen refreshes to two different graphing outputs, one being a single measurement and the other being a repeating measurement for 5 seconds. The physical aspect of the project will be within the style of just pins (female to male or male to male) connected to the Tiva board either directly to the CRO/DSO/Wave generator or via a variable resistor using a breadboard.

The complex design in the project is to create a user interface that sits off-side to the graph, the main feature of the project is an already set graph and at the bottom people can control the features along with continuous fresh readings of the instantaneous voltage and minimum and maximum voltages, with more ability for higher sampling rates, the graph will run at a single measurement or a repeating measurement however adjustments can be made within the same window meaning that at no point will the window refresh but only the values themselves in real time. The physical aspect of the project will be for a more complex design is an enclosure that the Tiva will sit in an enclosure with either a breadboard or pins sticking out of the enclosure to be attached onto it, with enough time also a short instructional pictogram to show how to effectively use it.

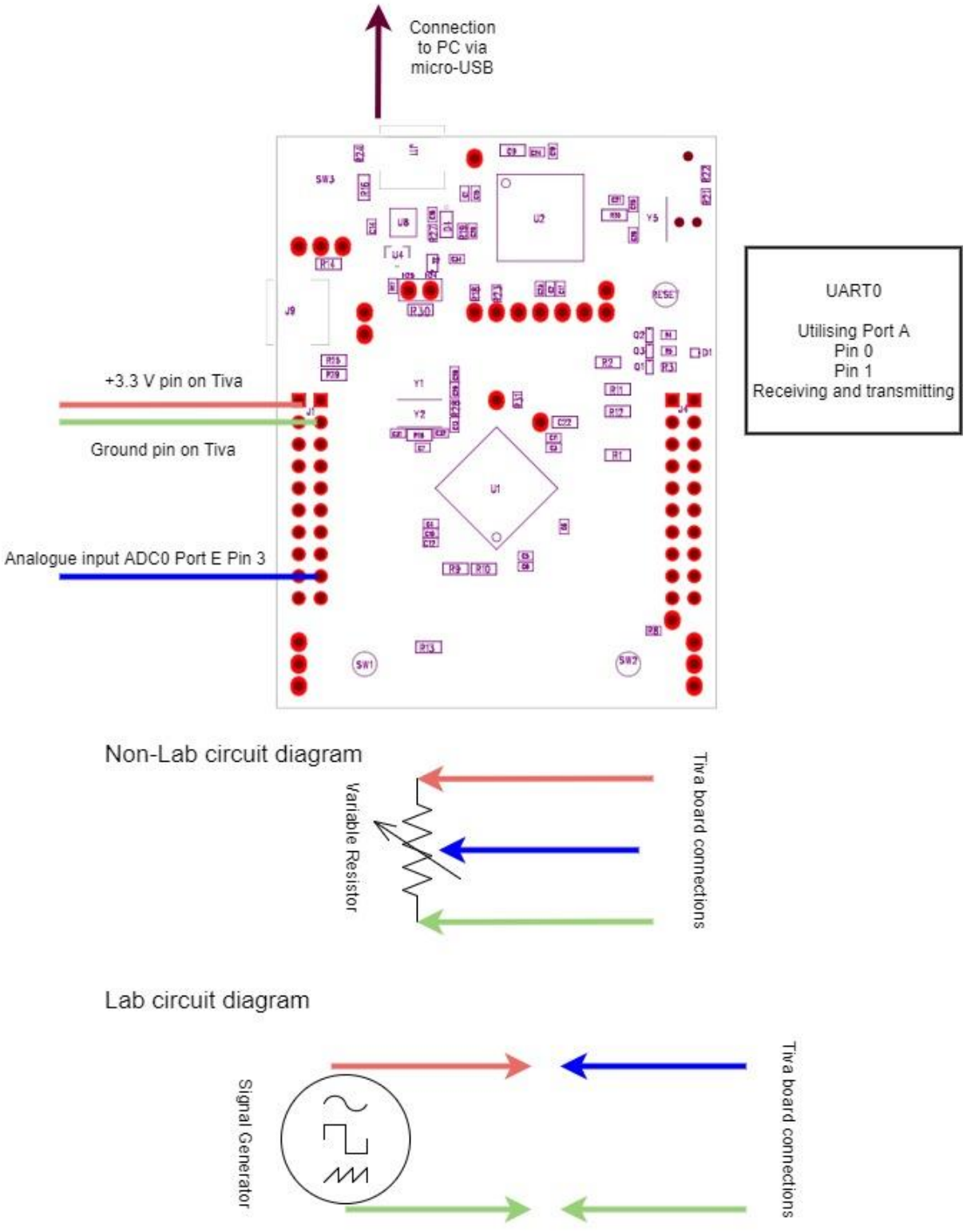
Solution

Block diagram of system structure



Block diagram representing the system and its connections between the ‘moving’ parts.

Circuit diagram



Circuit diagram showing Tiva Board's connection via ADC Ain0 with testing connections to a variable resistor and signal generator.

Subsystem set-up

This section will give the details on setting up the subsystems used in creating a DSO with the Tiva C Seires LaunchPad:

ADC set-up of Port E; Pin 3 for Ain0 -- ADC0

Clock set-up

Register name	Register address	Operation	Value
SYSCTL_RCGC0_R	0x400FE100	ORR	0x1000
SYSCTL_RCGCGPIO_R	0x400FE608	ORR	0x10
SYSCTL_RCGC1_R	0x400FE104	ORR	0x1000

* SYSCTL_RCGC#_R are known as legacy regisers for 0 -- ADC and 1 -- TIMER

The system clock is enabled for ADC0, Port E and TIMER0 – legacy registers were used for ADC and TIMER, delays were used afterwards to force the board to wait and stabilised.

GPIO set-up

GPIO Port E Base: 0x40024000

Register name	Register address	Operation	Value
DIR_R	0x400	BIC	0x8
DEN_R	0x51C	BIC	0x8
AFSEL_R	0x420	ORR	0x8
AMSEL_R	0x528	ORR	0x8

GPIO set-up for Port E clearing all digital settings and input settings to enable alternative select and analogue mode.

ADC preparation before TIMER

Register name	Register address	Operation	Value
ADC0_ACTSS_R	0x40038000	BIC	0x8
ADC0_SSPRI_R	0x40038020	BIC	0x3000
ADC0_SSPRI_R	0x40038020	ORR	0x3

ADC0 – Sequencer 3 (SS3) is disabled and priority of SS3 is made highest while priority of SS0 is made low.

TIMER set-up

Register name	Register address	Operation	Value
TIMER0_CTL_R	0x4003000C	BIC	0x1
TIMER0_CFG_R	0x40030000	ORR	0x4
TIMER0_TAMR_R	0x40030004	ORR	0x2
TIMER0_CTL_R	0x4003000C	ORR	0x20
TIMER0_TAPR_R	0x40030038	MOV	255
TIMER0_TAILR_R	0x40030028	MOV	255
TIMER0_TAMATCH_R	0x40030030	MOV	0

The TIMER0A is disabled to be configured for a periodic triggering to the ADC that has a prescaler of 255 with an arbtriary value of 255 within the preload which will be changed when the program starts, match value is zero for no use of duty cycle.

ADC set-up & enabling of ADC & TIMER

Register name	Register address	Operation	Value
ADC0_EMUX_R	0x40038014	ORR	0x5000
ADC0_SSMUX3_R	0x400380A0	ORR	0x1
ADC0_SSCTL3_R	0x400380A4	ORR	0x2
TIMER0_CTL_R	0x4003000C	ORR	0x1
ADC0_ACTSS_R	0x40038000	ORR	0x8

SS3 is set to timer trigger as well as sampling control bits through source Ain0 then both TIMER0A and SS3 are enabled to be used.

UART set-up of Port A; Pin 0 Pin 1 for U0Rx & U0Tx -- UART0

Clock set-up

Register name	Register address	Operation	Value
SYSCTL_RCGCUART_R	0x400FE618	ORR	0x1
SYSCTL_RCGCGPIO_R	0x400FE608	ORR	0x1

UART set-up

Register name	Register address	Operation	Value
UART0_CTL_R	0x4000C030	BIC	0x1
UART0_CTL_R	0x4000C030	ORR	0x300
UART0_IBRD_R	0x4000C024	MOV	1
UART0_FBRD_R	0x4000C028	MOV	5
UART0_LCRH_R	0x4000C02C	ORR	0x6A

UART0 is disabled so transmitting and receiving data can be set and specified parameters for baud rate, word length, parity, stop bit are set as well.

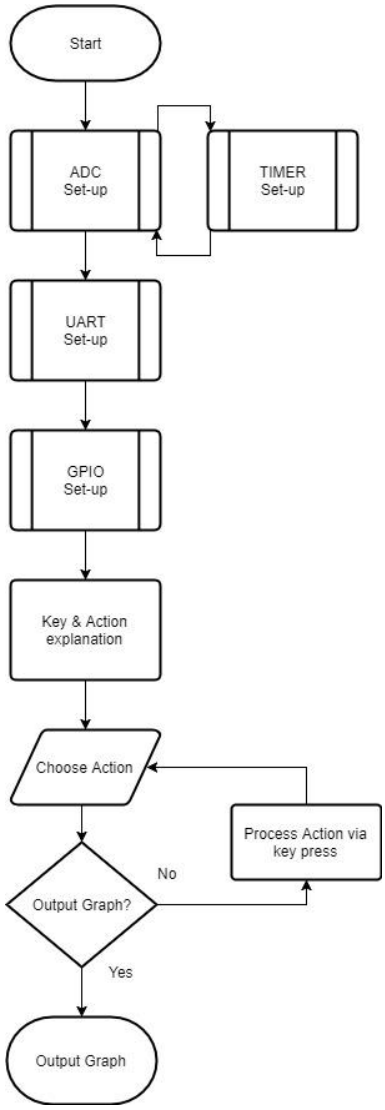
GPIO set-up

GPIO Port A Base: 0x40004000

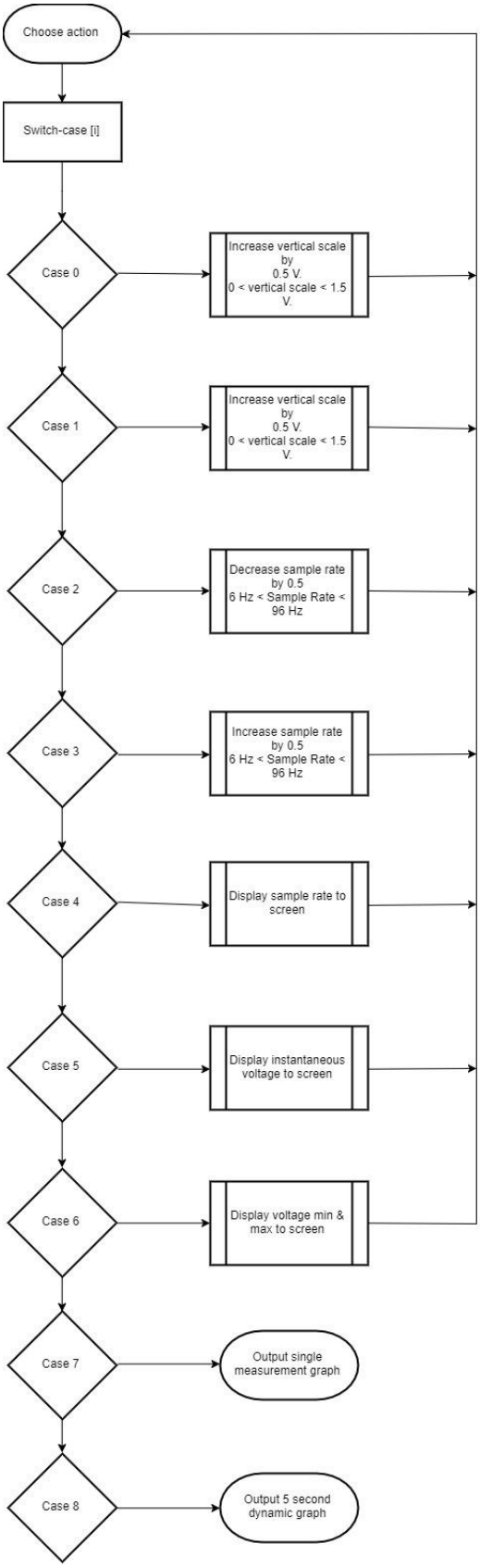
Register name	Register address	Operation	Value
AFSEL_R	0x420	ORR	0x3
DEN_R	0x51C	ORR	0x3
PCTL_R	0x52C	ORR	0x11
UART0_CTL_R	0x4000C030	ORR	0x1

GPIO for UART0 is set with alternative select enabled, digital enabled and the port control numbers to specific the alternative function (U0Rx, U0Tx – UART0) with enabling UART.

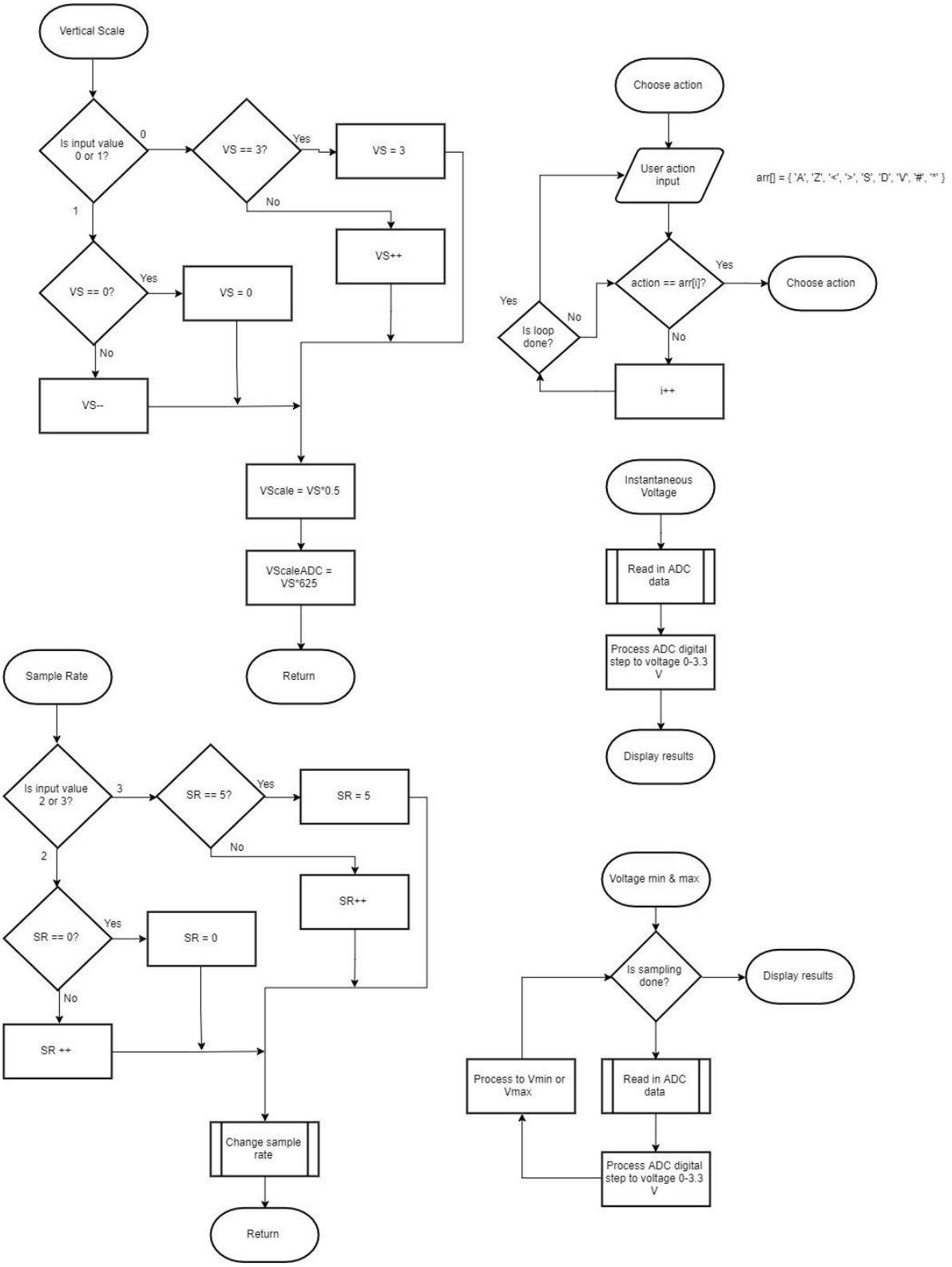
Algorithmic flowcharts



Algorithm of the main code processes the actions via user input, the key and actions are explained in the introduction of the report.



Action algorithm that results in the user choosing specified actions before outputting the actions to the construction of the graph.



Smaller algorithms from the action algorithm revealing how some of the processing works.

Testing methodology

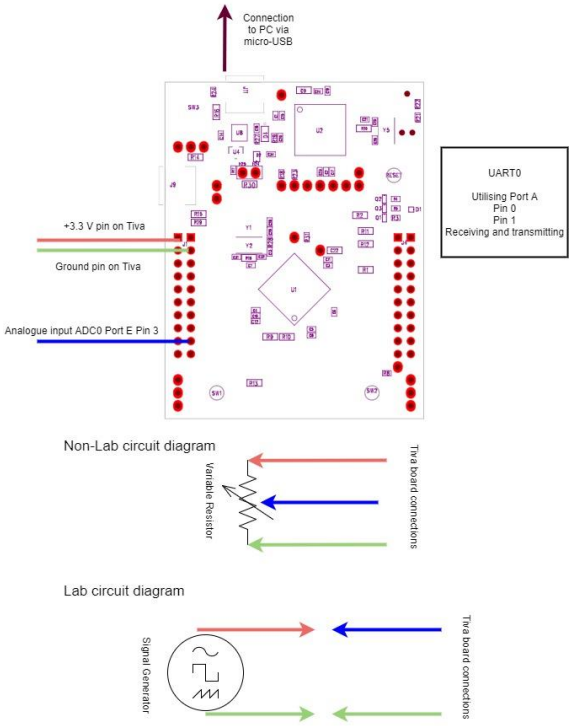
The testing of correct operation of subsystems were a step-by-step activation via the debugger built within Keil and ultimately via the UART testing that the terminal linking to the UART's specified baud

rate, word length, parity, stop bit and flow control (921600, 8-bit, odd parity, 2-bit stop, no flow control) and receiving and transmitting characters via an echo loop.

The ADC set-up was activated slowly by first through activating just the ADC0 and then testing the debug, then GPIO and then finally TIMER, once the debug showed that the set-up was not being caught in a loop, data retrieval was testing using a potentiometer via Port E Pin 3, ADC data was printed straight to screen unprocessed showing indeed that the step values was from 0 to 4095.

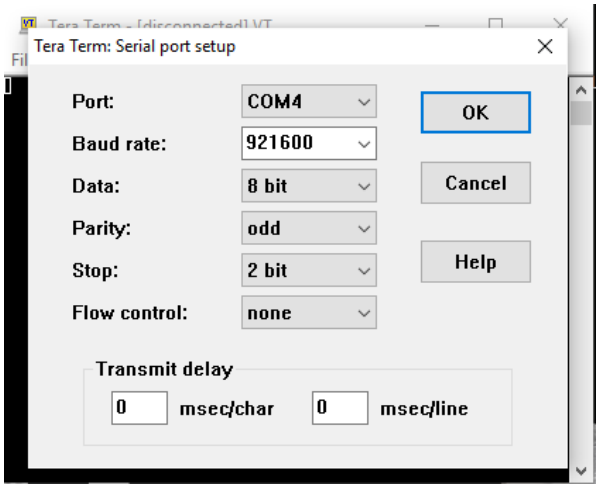
Each function within the program was tested via known variables and hardcoding static values that would be used in the program, the results needed were displayed straight to screen.

User Manual



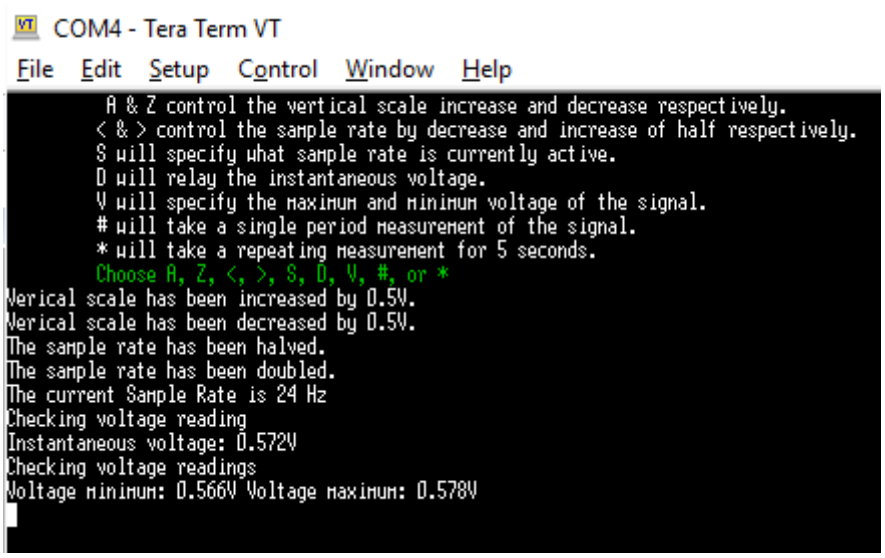
Circuit diagram.

Upon successfully connecting your Tiva C Series “DSO” LaunchPad to the computer and ensured that component that will be tested is attached to the appropriate pins via the circuit diagram, open Tera Term and begin to set-up the serial port to the specifications below:

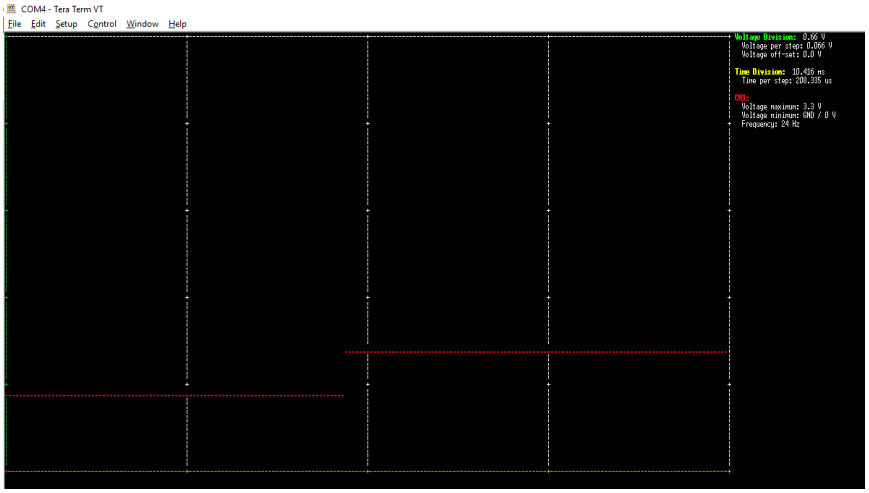


Tera Term specifications.

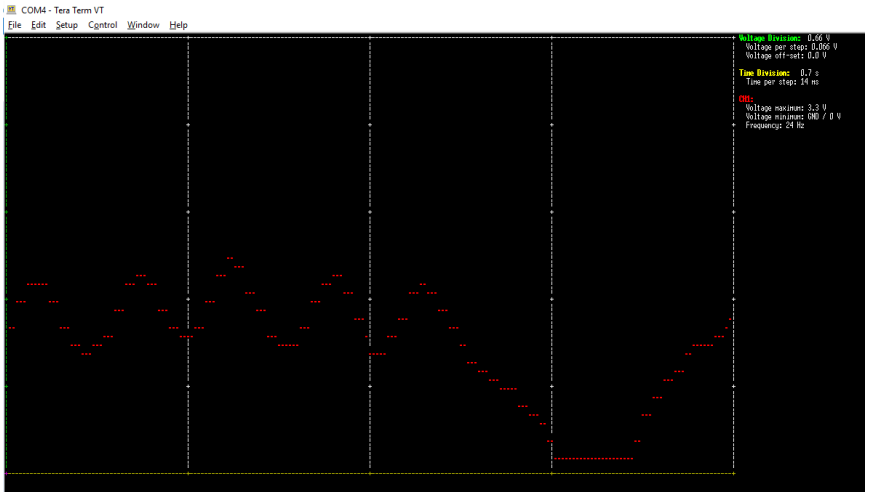
When the specifications is correct you will be given an input list of keyboard buttons and the actions that will occur; below is the list of buttons to press as well as the outputs you will be expected to see, leading up to the predefine values for the graph output and the two different types of graph output that the user can choose from – single measurement or measurement over 5 seconds.



Listed keys and actions.



Single measurement graph output.



5 second measurement graph output.

Evaluation

The project in which a Tiva C series TM4C123G LaunchPad Evaluation Board is used to create a Digital Storage Oscilloscope (DSO) is successful with resulting in a plotting of the signal that is being set once the user has set parameters within the user interface that is separated from both graph outputs while using a simple physical project design of using pins directly connecting to the Tiva board to the CRO/DSO via allotted laboratory time while outside the laboratory the simple connection was via a breadboard and a variable resistors.

Due to the project being within a time-frame, not enough time could have been utilised to create an enclosure for the Tiva board, the code for the complex project was not managed to be implemented however the user interface of the graph was recreated to provide better visual representation than the simplified white.

Problems have occurred during the creation of the project which will be explained in the discussion section of this report however although most problems have been solved or altered; a major on-going problem which due to the time-sensitive nature of this project, the ADC happens to halve all input values through the Ain0 pin which will also be discussed.

Discussion

The problems that have occurred during the project is within the initialisation of the ADC, during the debug testing it would move through the ADC without a problem however the input pin's ADC data value was pulling zero values to the screen, upon inspection of the registers and content it was revealed that the preload register was set to a value that was outside the parameter causing the register to wrap around in value to zero thus ADC not triggering at all.

Embedded C has functions that don't work or are left as a skeleton that needs to be filled in, one such example is the 'printf()' function that exists within the 'stdio.h' header that is left as a skeleton ready to be filled in with self-created functions that are known as 'fgets()', 'fgetc()' and 'ferror()'. These functions that need to be created in order for 'printf()' to work however even with the self-created functions the function still did not work thus using a 'out_char' function that outputs a

character via UART data register is used in a for loop until length of the string is reached, the starting memory address of the string is given to the print string function using a pointer to the start of the string wanted to be printed.

Due to the time-frame and the set back with the direct register set-up taking longer than expected the complex project milestones could not be implemented such as the single page graph with user commands and dynamic values that would be constantly updated upon user selection or while the graph was in use with either single measurement or set time.

A major on-going issue is that the ADC is halving the input numbers being read by the pin, the registers are been looked through and the generator signal was tested by the Tiva board and by the CRO in which was generated by the CRO so evaluated possible problems, the registers although have been looked through intensely via the datasheet available for the TM4C123G – the only solutions thus far is debugging and ultimately re-writing the functions that create the graph as well as re-evaluating the direct register set-up or re-building the project from the ground up using the previous code as a frame-network possibly re-creating more efficient ways of handling the task.

Conclusion

Although the project is completed and complex project model has not been implemented, with the exception of the ADC halving values being sent to the pin and not allowing the solution of hard-coding the project does indeed project a input signal at different frequencies via the user input as predefined values before analysing the signal through a display of the graph output, room for improvement of the project overall as how it could work and how it could look could have been implemented given more time of excess of a week however the dates were already set with no flexibility.

References

Texas Instruments, 2013, *Tiva C Series TM4C123G Evaluation Board – User's Guide*, Texas Instruments Incorporated, Dallas, Texas 75265. Literature Number: SPMU296

David Rowlands, 2017, *1B) Introduction to uCs - 2303ENG Lecture Notes*, Griffith School of Engineering, Griffith University.

David Rowlands, 2017, *6B) General Purpose Timers - 2303ENG Lecture Notes*, Griffith School of Engineering, Griffith University.

David Rowlands, 2017, *Analog Signals - 2303ENG Lecture Notes*, Griffith School of Engineering, Griffith University.

Appendix: Code

main.c

uart.c

```
// uart.c

/*
        UART setup
*/

#include <stdio.h>

#include <string.h>

#include "tm4c123gh6pm.h"

#include "uart.h"

#include "functions.h"


// UART Setup -- 921600, 8 bit, Odd parity, 2 bit stop, No flow control

/*
        ==================================
                UART initialisation
        ==================================
*/

void Setup_UART(void){


        SYSCTL_RCGCUART_R |= 0x1;
        // UART0 Clock Enable

        Delay(2);

        // Forced delay

        SYSCTL_RCGCGPIO_R |= 0x1;
        // GPIO Clock Port A Enable

        Delay(2);

        UART0_CTL_R = UART0_CTL_R & ~0x1;
        // UART0 Disable

        UART0_CTL_R |= 0x300;

        // Rx, Tx Enable, Prescaler is 16

        UART0_IBRD_R = 1;

        // Baud Rate Integer

        UART0_FBRD_R = 5;

        // Baud Rate Fractional

        UART0_LCRH_R |= 0x6A;
        //

        GPIO_PORTA_AFSEL_R |= 0x3;
        // Setting alternative PA[0,1]

        GPIO_PORTA_DEN_R |= 0x3;
        // Setting digital PA[0,1]

        GPIO_PORTA_PCTL_R |= 0x11;
        // Setting Port Control

        UART0_CTL_R |= 0x1;
        // Enable UART0

}

/*
        ==================================
                UART character read in
        ==================================
*/

char ReadChar(void){


        while( (UART0_FR_R & 0x10) != 0 );
        // Polling for input flag

        return( (char)(UART0_DR_R & 0xFF) );
        // Removal of all but data register being returned

}

/*
        ==================================
                UART character read out
        ==================================
*/

void WriteChar(char W){


        while( (UART0_FR_R & 0x20) != 0 );
        // Polling for output flag

        UART0_DR_R = W;
        // Placing in data to the register to be transmittered

}

/*
        ==================================
                String Output
```

```
	==================================
*/
void PrintS(char *string){
	int i;

	for( i=0; i < strlen(string); i++){
		WriteChar(string[i]);
	}
}
```

adc.c

```
// adc.c
/*

*/
#include <stdio.h>
#include "tm4c123gh6pm.h"
#include "adc.h"
#include "functions.h"

/*
	==================================
		ADC Defines
	==================================
*/
#define	Reload0		10417
#define	Reload1		5208
#define	Reload2		2604
#define	Reload3		1302
#define	Reload4		651
/*
	==================================
		ADC variables
	==================================
*/
int RLD[5] = { Reload0, Reload1, Reload2, Reload3, Reload4 };

/*
	==================================
		ADC initialisation
	==================================
*/
void Setup_ADC(void){
	// 00 01 02 03 | 04 05 06 07 | 08 09 10 11 | 12 13 14 15 | 16
	// 00 00 00 00 | 00 00 00 00 | 00 00 00 00 | 00 00 00 00 | 01
	// Clock setup for ADC & GPIO
	SYSCTL_RCGC0_R |= 0x10000;
	// Enabling ADC0
	SYSCTL_RCGC0_R |= 0x00000000;
	// Default sample rate (125k samples/sec)
	SYSCTL_RCGCGPIO_R |= 0x10;
	// GPIO Clock Port E enabled
	Delay(2);
	// Forced clock delay
	Delay(2);
	// GPIO Port E Setup
	GPIO_PORTE_DIR_R &= ~0x8;
	// Input set
	GPIO_PORTE_DEN_R &= ~0x8;
	// Digital disabled
	GPIO_PORTE_AFSEL_R |= 0x8;
	// Alternative set
	GPIO_PORTE_AMSEL_R |= 0x8;
	// Analogue in set
	// ADC disbale and prioirty setting
	ADC0_ACTSS_R &= ~0x00000008;
	// SS3 disable
	ADC0_SSPRI_R &= ~0x3000;
	// SS3 set to high priority
	ADC0_SSPRI_R |= 0x3;
	// SS0 set to lowest prioirty
	// TIMER0 Clock and setting
	SYSCTL_RCGC1_R |= 0x10000;
	// ADC TIMER0 Enabled -- Legacy
	Delay(2);
	// Forced clock delay
	Delay(2);
	// Forced clock delay
	TIMER0_CTL_R &= ~0x1;
	// TIMER0 disabled
	TIMER0_CFG_R |= 0x4;
	//
```

functions.c

```
	TIMER0_TAMR_R |= 0x2;
	// Periodic enabled

	TIMER0_CTL_R |= 0x20;
	// ADC trigger set

	TIMER0_TAPR_R = 255;
	// 256 prescaler

	TIMER0_TAILR_R = 255;
	// Period set

	TIMER0_TAMATCHR_R = 0;
	// No match value

	// ADC setting

	ADC0_EMUX_R |= 0x5000;
	SS3 set to timer trigger

	ADC0_SSMUX3_R |= 0x1;
	// Configure source Ain0

	ADC0_SSCTL3_R |= 0x2;
	// Configure sampling control bits

	TIMER0_CTL_R |= 0x1;
	// TIMER0 enabled

	ADC0_ACTSS_R |= 0x8;
	// ADC enabled


}

/*

	==================================

		Changing sample rate

	==================================

*/

void ChangeSampleRate(int SR){

	TIMER0_TAILR_R = RLD[SR];
	// Changing sampling period

}

/*

	==================================

		Reading ADC data

	==================================

*/

int ADC_Read(void){


	return(ADC0_SSFIFO3_R & ADC_SSFIFO3_DATA_M);
	// Accessing ADC data

}
```

graph.c