Appendix

Codewarrior code – main.c

```
//*********************************************************************
//*                          FINAL PROJECT
//*                       McMaster University
//*                      2DP4 Microcontrollers
//*                  Brian Le leb7 400124577
//*********************************************************************
//*********************************************************************

#include <stdio.h>
#include <hidef.h>      /* common defines and macros */
#include "derivative.h"      /* derivative-specific definitions */
#include "SCI.h"
#include "math.h"

/*Prototypes*/
void setClk(void);
void delay1ms(unsigned int multiple);
void OutCRLF(void);

unsigned int n;
unsigned short val;
int mode;
int theta;

void OutCRLF(void){
  SCI_OutChar(CR);
  SCI_OutChar(LF);
  PTJ ^= 0x20;
}

void main(void) {
  /* put your own code here */

  setClk(); //set bus clock to 8 MHz

  //AD Conversion Presets
  ATDCTL0 = 0x01;
  ATDCTL1 = 0x4F;
  ATDCTL3 = 0x88;
  ATDCTL4 = 0x02;
  ATDCTL5 = 0x25;
  PER1AD = 0x00;

  TSCR1 = 0x90;    //Timer System Control Register 1
                   // TSCR1[7] = TEN:  Timer Enable (0-disable, 1-enable)
                   // TSCR1[6] = TSWAI:  Timer runs during WAI (0-enable, 1-
disable)
                   // TSCR1[5] = TSFRZ:  Timer runs during WAI (0-enable, 1-
disable)
                   // TSCR1[4] = TFFCA:  Timer Fast Flag Clear All (0-normal
1-read/write clears interrupt flags)
```

```
                    // TSCR1[3] = PRT:  Precision Timer (0-legacy, 1-
precision)
                    // TSCR1[2:0] not used

  TSCR2 = 0x01;    //Timer System Control Register 2
                    // TSCR2[7] = TOI: Timer Overflow Interrupt Enable (0-
inhibited, 1-hardware irq when TOF=1)
                    // TSCR2[6:3] not used
                    // TSCR2[2:0] = Timer Prescaler Select: See Table22-12 of
MC9S12G Family Reference Manual r1.25 (set for bus/1)


  TIOS = 0xFE;     //Timer Input Capture or Output capture
                    //set TIC[0] and input (similar to DDR)
  PERT = 0x01;     //Enable Pull-Up resistor on TIC[0]

  TCTL3 = 0x00;    //TCTL3 & TCTL4 configure which edge(s) to capture
  TCTL4 = 0x02;    //Configured for falling edge on TIC[0]

/*
 * The next one assignment statement configures the Timer Interrupt Enable
 */

  TIE = 0x01;      //Timer Interrupt Enable

/*
 * The next one assignment statement configures the ESDX to catch Interrupt
Requests
 */

    EnableInterrupts; //CodeWarrior's method of enabling interrupts
  //Set Environment

  n = 0;
  mode = 0;

  SCI_Init(19200);

  //Set Ports
  DDRJ = 0xFF;      //set all port J as output
  DDR0AD = 0x0F;
  DDR1AD = 0x1F;




  //Main Algorithm begins

  for(;;) {

    val = ATDDR0;
    delay1ms(100);
    if (val <= 1325){
      val = 1325;
    }
    if (val > 1605){
```

```
      val = 1605;
    }

    theta = 9*val/28 - 425;
    SCI_OutUDec(theta);
    OutCRLF();

    if (mode == 0){
      delay1ms(100);

    }else{
      if (mode == 1 && (theta >=0 && theta < 93)){
        PT0AD = 0x00 + (theta%10);
        PT1AD = 0x00 + (theta/10);
      } else if (mode == 2 && (theta >=0 && theta < 93)){
        if (theta >=0 && theta <= 40){
          PT0AD = 0x0F >> 4-(theta/10);
          PT1AD = 0x00;
        }else{
          PT0AD = 0x0F;
          PT1AD = 0x1F >> 9-(theta/10);
        }
      }
    }


  } /* loop forever */
  /* please make sure that you never leave main */


}



void setClk(void){
  CPMUPROT = 0x26;      //Protection of clock configuration is disabled, maybe
CPMUPROT=0.

  CPMUCLKS = 0x80;      //PLLSEL=1. Select Bus Clock Source is PLL clock
  CPMUOSC = 0x80;       //OSCE=1. Select Clock Reference for PLLclk as fOSC (8
MHz).

  CPMUREFDIV = 0x41;    //fREF= fOSC/(REFDIV+1) -> fREF= fOSC/(2) -> fREF= 4
MHz.

  CPMUSYNR=0x01;        //VCOCLK = fVCO= 2 * fREF* (SYNDIV+1) -> fVCO= 2 * 4
MHz * (1+1) fVCO = 16 MHz.

  CPMUPOSTDIV=0x00;     //PLLCLK = VCOCLK/(POSTDIV+1) -> PLLCLK = 16 MHz/(0+1)
-> PLLCLK = 16 MHz.
                        // fBUS=fPLL/2=16 MHz/2 = 8 MHz

  while (CPMUFLG_LOCK == 0) {}  //Wait for PLL to achieve desired tolerance
of target frequency. NOTE: For use when the source clock is PLL. comment out
when using external oscillator as source clock
```

```
  CPMUPROT = 1;                  //Protection for clock configuration is
reenabled

}

void delay1ms(unsigned int multiple){

  unsigned int i;
  unsigned int j;

  for(j = 0; j<multiple; j++){
    for(i = 0; i<100; i++){
      // Delay
      PTJ = PTJ;
      PTJ = PTJ;
      PTJ = PTJ;
      PTJ = PTJ;
      PTJ = PTJ;
    }
  }

}

interrupt  VectorNumber_Vtimch0 void ISR_Vtimch0(void)
{
  unsigned int temp;
  PT0AD = PT0AD;
  PT1AD = PT1AD;
  if(mode==0)
  {
    mode = 1;
    SCI_OutChar(CR);
    //SCI_OutString("ESX not ready for input"); SCI_OutChar(CR);
    PTJ ^= 0x01;      //Toggles pin/LED state
  } else if(mode==1)
  {
    mode = 2;
  } else if (mode == 2)
  {
    mode = 0;
  }
  delay1ms(250);
  temp = TC0;       //Refer back to TFFCA, we enabled FastFlagClear, thus by
reading the Timer Capture input we automatically clear the flag, allowing
another TIC interrupt
}
```

Matlab Code

```
% Brian Le - 400124577 Leb7

time = now;
theta = 0;

old = instrfind;
if(~isempty(old))
```

```
    fclose(old);
    delete(old);
end
clear;
s = serial('COM3','BaudRate',19200,'Terminator','CR');
fopen(s);
display = figure('Name', 'Time v. Theta');
axesHandle = axes('Parent',display,'Color',[1 1 1]);
hold on;
ylim([0 90]);
title('Time (s) vs Theta (deg)','FontWeight','bold','Color','k');
xlabel('Time (s)','Color','k');
ylabel('Theta (deg)','Color','k');

count = 1;
while true
    time(count) = now;
    theta(count) = fread(s,1,'uchar');
    count = count + 1;
    plot(axesHandle,time,theta,'k');
    datetick('x','SS');
    pause(0.05);
end
```