Appendix

```
//*****************************************************************
//*                   Final Project (without bonus)
//*                       McMaster University
//*                      2DP4 Microcontrollers
//*                   Zishu Wu   wuz78    400089778
//*****************************************************************
//*****************************************************************
//*                            Description
//*The code receives the data from angle sensor and send it to the MATLAB.
//*The value of the angle (0 ~ 90 degree) is displaced by LED with 2 MODE
//*Mode 0: BCD; Mode 1: 10-degree bar (rounded).
//*****************************************************************
//*****************************************************************
//*                             References
//* 2DP4Lecture_DoyleFazliani_W8 Slide22,28,52
//* 2DP4Lecture_DoyleFazliani_W9 Slide5,7,15
//* HCS12 instruction set reference
//* The HCS19 9S12 An Introduction to Software and Hardware Interface
//*****************************************************************
//*****************************************************************
  #include <hidef.h>      /* common defines and macros */
  #include "derivative.h" /* derivative-specific definitions */
  #include "SCI.h"
/*Prototypes*/
  void setClk(void);
  void delay1ms(unsigned int multiple);
  void OutCRLF(void);


         /* DECLARE GLOBAL VARIABLES BELOW*/
 ////////////////////////////////////////////////////
unsigned int num;
unsigned int x;
unsigned int deg;
char char_1;
char char_2;
unsigned int ten;
unsigned int one;
////////////////////////////////////////////////////

void main(void) {
  ////////Configure ACD conversion///////////
  ATDCTL1 = 0x4F;       // set for 12-bit resolution
    ATDCTL3 = 0x88;     // right justified, one sample per sequence
    ATDCTL4 = 0x02;     // prescaler = 2; ATD clock = 6MHz / (2 * (2 + 1)) ==
1MHz
    ATDCTL5 = 0x25;     // continuous conversion on channel 0

  // /////Initialize uC environment///////
  setClk();         //set bus speed 6MHz
  SCI_Init(14400);

  ///////Configure timer module/////////////////
  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 = 0x00;     //Timer System Control Register 2
  TIOS = 0x20;      //Timer Input Capture or Output capture
                    //set TIC[0] as input
                    //set TIC[5] as output
  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] (channel 0)
  TIE = 0x03;       //Timer Interrupt Enable

  ///////CONFIGURE INPUT/OUTPUT port//////
  DDRS = 0b11111011;//S2 as input --> switch.
  PERS = 0b00000100;//enable pull-up resisters for input pins
  DDRT = 0b00011110;//DDRT: data direction register [1,2,3,4]
  DDRP = 0b00111110;//port port 1,2,3,4,5 as output
  DDRJ = 0xFF;       //set all port J as output

  num = 0;
    EnableInterrupts;

  for(;;) {
   if (num % 2 == 1){/////START COMMUNICATION///////
     x= ATDDR0;
   char_1 = x & 0x00FF;
   char_2 = x >> 8;
   SCI_OutChar(char_1);
   SCI_OutChar(char_2);
   delay1ms(200);
   deg = SCI_InChar();
     ////////MODE 0/////////
     if (PTIS == 7){ //angle = XY
         ten = deg/10;  // --> X
         one = deg%10;  // --> Y
         PTJ = 0x01;
         ///////tens digit(X) --> PORT p[4:1]///////
         if(ten==0){
          PTT = 0b00000000;
         }else if (ten == 3||ten == 2){
          PTT = 0b00000010 ;
         }else if (ten == 4||ten == 5){
          PTT = 0b00000100;
         }else if (ten == 6||ten == 7){
          PTT = 0b00000110;
         }else if (ten == 8||ten == 9){
          PTT = 0b00001000;
         }
         ///////single digit (Y) --> port P [5:1]////
         if (one == 1){
```

```
      PTP = 0b00000000;
      } else if (one == 2){
      PTP = 0b00000010;
      } else if (one == 3){
      PTP = 0b00000100;
      } else if (one == 4){
      PTP = 0b00001000;
      } else if (one == 5){
      PTP = 0b00001010;
      } else if (one == 6){
      PTP = 0b00001100;
      } else if (one == 7){
      PTP = 0b00001110;
      } else if (one == 8){
      PTP = 0b00010010;
      } else if (one == 9){
      PTP = 0b00010010;
      }

      if (ten%2==1){
       PTP = PTP | 0b00100000;
      }//even number --> PP5 = 0, otherwise PP5 = 1
 }
 else{
 //////////////////////
 ////////MODE 1/////////
  if(deg>=85){
      PTT=0b10011111;
      PTP=0b10111110;
  }else if(deg<85 && deg>=75){
      PTT=0b10011111;
      PTP=0b10111100;
  }else if (deg<75 && deg>=65){
      PTT=0b10011111;
      PTP=0b10111000;
  }else if (deg<65 && deg>=55){
      PTT=0b10011111;
      PTP=0b10110000;
  }else if (deg<55 && deg>=45){
      PTT=0b10011111;
      PTP=0b10100000;
  }else if (deg<45 && deg>=35){
      PTT=0b10011110;
      PTP=0b10000000;
  }else if (deg<35 && deg>=25){
      PTT=0b10011100;
      PTP=0b10000000;
  }else if (deg<25 && deg>=15){
      PTT=0b10011000;
      PTP=0b10000000;
  }else if (deg<15){
      PTT=0b10010000;
      PTP=0b10000000;
  } //mode 1
 }

delay1ms(100);
```

```
   } //communication
//////STOP COMMUNICATION///////



        }  /*loop end*/
  } /* main end */



////////////////INTERRUPT BUTTON////////////////////
interrupt  VectorNumber_Vtimch0 void ISR_Vtimch0(void)
{
  unsigned int temp;

    num +=1;

    temp = TC0;
}

void delay1ms(unsigned int multiple){
  unsigned int i; //loop control variable
  TSCR1 = 0x90;   //enable timer and fast timer flag clear
  TSCR2 = 0x00;   //Disable timer interrupt, set prescaler=1
  //TIOS |= 0x01;   //Enable OC0 (not necessary)
  TIOS = 0x20;  //0010 0000 --> IOS[5]as output for timer,IOS[0]as input for
button
  TC5 = TCNT + 6000;
  for(i=0;i<multiple;i++) {
    TFLG2 = 0x80; //clear the TOF flag
    while (!(TFLG1_C5F));
    TC5 += 6000;
  }
  TIOS &= -0x20; //Disable OC0 (not necessary)
}
////////////////Set Clock////////////////////
#define VCOFRQ 0x00        //VCOFRQ[1:0]  32MHz <= VCOCLK <= 48MHz
#define SYNDIV 0x05        //SYNDIV[5:0]
#define REFFRQ 0x00        //REFFRQ[1:0]  2MHz < fREF <= 6MHz
#define REFDIV 0x80        //REFDIV[3:0]  Ref Divide is 1
void setClk(void){
  CPMUPROT = 0x26;              //Protection of clock configuration is
disabled
                                // NOTE: On some Esduinos you may need to
use CPMUPROT=0. Try both and see which value allows you to change the
registers below in your debugger!

  CPMUCLKS = 0x80;              //PLLSEL=1. Select Bus Clock Source:  PLL
clock or Oscillator.
  CPMUOSC = 0x00;               //OSCE=1. Select Clock Reference for PLLclk
as:fOSC (8 MHz).

  CPMUREFDIV = REFFRQ+REFDIV;    //Set fREF divider and selects reference
clock frequency Range. fREF= 4 MHz.
```

```
  CPMUSYNR=VCOFRQ + SYNDIV;      //Set Syn divide and selects VCO frequency
range. fVCO = 48 MHz.

  CPMUPOSTDIV=0x00;              //Set Post Divider (0x00= 0000 0000).

  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
}


////////////////OutCRLF////////////////////
void OutCRLF(void){
  SCI_OutChar(CR);
  SCI_OutChar(LF);
}
```

BONUS 2 - XY-Axis

```
//*****************************************************************
//*                         Final Project
//*                       BONUS 2 (X-Y axis)
//*                       McMaster University
//*                      2DP4 Microcontrollers
//*                  Zishu Wu   wuz78    400089778
//*****************************************************************
//*****************************************************************
//*                            Description
//* Plot X and Y axis on the same graph.
//* Use Matlab "xy axis to performance this function"
//*****************************************************************
//*****************************************************************
//*                             References
//* 2DP4Lecture_DoyleFazliani_W8 Slide22,28,52
//* 2DP4Lecture_DoyleFazliani_W9 Slide5,7,15
//* HCS12 instruction set reference
//* The HCS19 9S12 An Introduction to Software and Hardware Interface
//*****************************************************************
//*****************************************************************
/*Include*/
  #include <hidef.h>      /* common defines and macros */
  #include "derivative.h" /* derivative-specific definitions */
  #include "SCI.h"
/*Prototypes*/
  void setClk(void);
  void delay1ms(unsigned int multiple);
  void OutCRLF(void);

         /* DECLARE GLOBAL VARIABLES BELOW*/
 ////////////////////////////////////////////////////
```

```
unsigned int x;
unsigned int deg;
char char_1;
char char_2;
unsigned int y;
 ////////////////////////////////////////////////////

void main(void) {

  // Initialize uC environment//
  setClk();          //set bus speed 6MHz
  SCI_Init(14400);

  ////////////////////ADC configuration//////////////////////////
  ATDCTL1 = 0x4F;       // set for 12-bit resolution
    ATDCTL3 = 0x90;     // right justified, one sample per sequence
    ATDCTL4 = 0x02;     // prescaler = 2; ATD clock = 6.25MHz / (2 * (2 + 1))
== 1.04MHz
    ATDCTL5 = 0x35;     // continuous conversion on channel 0

  for(;;) {
   x = ATDDR0;
   y = ATDDR1;

   char_1 = x & 0x00FF;
   char_2 = x >> 8;
   SCI_OutChar(char_1);
   SCI_OutChar(char_2);

   //SCI_OutString("X is");SCI_OutUDec(x);
   //SCI_OutString("   Y is");SCI_OutUDec(y); OutCRLF();

   char_1 = y & 0x00FF;
   char_2 = y >> 8;
   SCI_OutChar(char_1);
   SCI_OutChar(char_2);
   delay1ms(50);

        }  /*loop end*/
  } /* main end */


//////////////////////FUNCTION ////////////////////////
//////////////////Delay Function/////////////////////////
void delay1ms(unsigned int multiple){
  unsigned int i; //loop control variable
  TSCR1 = 0x90;   //enable timer and fast timer flag clear
  TSCR2 = 0x00;   //Disable timer interrupt, set prescaler=1
  TIOS |= 0x01;   //Enable OC0 (not necessary)
  TC0 = TCNT + 6000;
  for(i=0;i<multiple;i++) {
    TFLG2 = 0x80; //clear the TOF flag
    while (!(TFLG1_C0F));
    TC0 += 6000;
  }
  TIOS &= -0x01; //Disable OC0 (not necessary)
}
```

```
////////////////Set Clock//////////////////
#define VCOFRQ 0x00        //VCOFRQ[1:0]  32MHz <= VCOCLK <= 48MHz
#define SYNDIV 0x05        //SYNDIV[5:0]
#define REFFRQ 0x00        //REFFRQ[1:0]  2MHz < fREF <= 6MHz
#define REFDIV 0x80        //REFDIV[3:0]  Ref Divide is 1
void setClk(void){
  CPMUPROT = 0x26;              //Protection of clock configuration is
disabled
                                // NOTE: On some Esduinos you may need to
use CPMUPROT=0. Try both and see which value allows you to change the
registers below in your debugger!

  CPMUCLKS = 0x80;              //PLLSEL=1. Select Bus Clock Source:  PLL
clock or Oscillator.
  CPMUOSC = 0x00;               //OSCE=1. Select Clock Reference for PLLclk
as:fOSC (8 MHz).

  CPMUREFDIV = REFFRQ+REFDIV;    //Set fREF divider and selects reference
clock frequency Range. fREF= 4 MHz.

  CPMUSYNR=VCOFRQ + SYNDIV;      //Set Syn divide and selects VCO frequency
range. fVCO = 48 MHz.

  CPMUPOSTDIV=0x00;              //Set Post Divider (0x00= 0000 0000).

  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
}

////////////////OutCRLF//////////////////
void OutCRLF(void){
  SCI_OutChar(CR);
  SCI_OutChar(LF);
}
```

BONUS 3 - 360 degrees measurement

```
//*********************************************************************
//*                          Final Project
//*                      BONUS 3 - 360 degrees
//*                       McMaster University
//*                      2DP4 Microcontrollers
//*                  Zishu Wu   wuz78    400089778
//*********************************************************************
//*********************************************************************
//*                            Description
//*Measure 1 dimension angle for full 360 degrees
//*Please use MATLAB file "full_degree" to implement this function!
//*********************************************************************
//*********************************************************************
//*                             References
```

```
//* 2DP4Lecture_DoyleFazliani_W8 Slide22,28,52
//* 2DP4Lecture_DoyleFazliani_W9 Slide5,7,15
//* HCS12 instruction set reference
//* The HCS19 9S12 An Introduction to Software and Hardware Interface
//*********************************************************************
//*********************************************************************
/*Include*/
  #include <hidef.h>      /* common defines and macros */
  #include "derivative.h" /* derivative-specific definitions */
  #include "SCI.h"
/*Prototypes*/
  void setClk(void);
  void delay1ms(unsigned int multiple);
  void OutCRLF(void);


         /* DECLARE GLOBAL VARIABLES BELOW*/
 ////////////////////////////////////////////////////

unsigned int x;
unsigned int deg;
unsigned int ones;
unsigned int tens;
char char_1;
char char_2;
unsigned int z;
 ////////////////////////////////////////////////////

void main(void) {

  // Initialize uC environment//
  setClk();         //set bus speed 6MHz
  SCI_Init(14400);

  ////////////////////////////////////////
  ATDCTL1 = 0x4F;       // set for 12-bit resolution
    ATDCTL3 = 0x90;     // right justified, one sample per sequence
    ATDCTL4 = 0x02;     // prescaler = 2; ATD clock = 6.25MHz / (2 * (2 + 1))
== 1.04MHz
    ATDCTL5 = 0x35;     // continuous conversion on channel 0


  ///////CONFIGURE INPUT/OUTPUT port//////

  for(;;) {
   x = ATDDR0;
   z = ATDDR1;

   char_1 = x & 0x00FF;
   char_2 = x >> 8;
   SCI_OutChar(char_1);
   SCI_OutChar(char_2);

   //SCI_OutString("X is");SCI_OutUDec(x);
   //SCI_OutString("   Z is");SCI_OutUDec(z); OutCRLF();

   char_1 = z & 0x00FF;
```

```
    char_2 = z >> 8;
    SCI_OutChar(char_1);
    SCI_OutChar(char_2);
    delay1ms(50);

         }  /*loop end*/
  } /* main end */



//////////////////////FUNCTION ////////////////////////
//////////////////Delay Function//////////////////////////
void delay1ms(unsigned int multiple){
  unsigned int i; //loop control variable
  TSCR1 = 0x90;   //enable timer and fast timer flag clear
  TSCR2 = 0x00;   //Disable timer interrupt, set prescaler=1
  TIOS |= 0x01;   //Enable OC0 (not necessary)
  TC0 = TCNT + 6000;
  for(i=0;i<multiple;i++) {
    TFLG2 = 0x80; //clear the TOF flag
    while (!(TFLG1_C0F));
    TC0 += 6000;
  }
  TIOS &= -0x01; //Disable OC0 (not necessary)
}


////////////////Set Clock////////////////////
#define VCOFRQ 0x00        //VCOFRQ[1:0]  32MHz <= VCOCLK <= 48MHz
#define SYNDIV 0x05        //SYNDIV[5:0]
#define REFFRQ 0x00        //REFFRQ[1:0]  2MHz < fREF <= 6MHz
#define REFDIV 0x80        //REFDIV[3:0]  Ref Divide is 1
void setClk(void){
  CPMUPROT = 0x26;              //Protection of clock configuration is
disabled
                                // NOTE: On some Esduinos you may need to
use CPMUPROT=0. Try both and see which value allows you to change the
registers below in your debugger!

  CPMUCLKS = 0x80;              //PLLSEL=1. Select Bus Clock Source:  PLL
clock or Oscillator.
  CPMUOSC = 0x00;               //OSCE=1. Select Clock Reference for PLLclk
as:fOSC (8 MHz).

  CPMUREFDIV = REFFRQ+REFDIV;    //Set fREF divider and selects reference
clock frequency Range. fREF= 4 MHz.

  CPMUSYNR=VCOFRQ + SYNDIV;      //Set Syn divide and selects VCO frequency
range. fVCO = 48 MHz.

  CPMUPOSTDIV=0x00;              //Set Post Divider (0x00= 0000 0000).

  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
}


////////////////OutCRLF////////////////////
void OutCRLF(void){
  SCI_OutChar(CR);
  SCI_OutChar(LF);
}
```