```c
/*
Author: Group 2 (Hang Xu, Wen Wu, Wenjun Ma)
Date complete: 28/2/2018
Filename: EE2A Experiment5 Wire-Following Sensor and Associated
Signal Processing-Improved version
Target device: PIC18F27K40
Fuse settings:NOMCLR, NOWDT,NOPROTECT,NOCPD
Program function: To determine the direction of the vehicle by
implementing the signal processing algorithm.
*/

#include <18F27K40.h>
#device adc=8
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
/***************main frequency setting***************/
#use delay(internal=64Mhz,clock_out)
/***************rs232 setting***************/
#pin_select U1TX=PIN_C0 // transmit data
#pin_select U1RX=PIN_C1 // receive data
#use rs232(uart1, baud=9600, ERRORS)
/***************spi setting***************/
#use spi(MASTER,DO=PIN_A2,MODE=0,CLK=PIN_A3,BITS=8) //set
SPI
/***************pwm setting***************/
#pin_select PWM4=PIN_A0 //select PIN_A0 as output of PWM
/***************structure***************/
struct IO_Port_Definition
   {
   int1 PWM;//PIN_A0(LDAC)
   int1 cs; //PIN_A1
   int1 SDO;//PIN_A2
   int1 SCK; //PIN_A3
   int unusedA:3;//PIN_A4..6
   int1 ADC;//PIN_A7
   int unusedB:8;//PIN_B0..7
   int1 ts;//PIN_C0
   int1 rc;//PIN_C1
   int unusedC:6; //PIN_C2..7
   };
struct IO_Port_Definition Port;
struct IO_Port_Definition PortDirection;
#byte Port = 0xF8D
#byte PortDirection = 0xF88
/***************variables***************/
//RDA//
char CommandString[32];
int    in=0;
//Command Stirng//
char collectdata[]="COLLECT DATA";
//ADC//
signed int8    adctable[128];
long    count=0;
//Main//
int    pause=1;//1 for continue;0 for stop
//Look up table//
unsigned int16 CosTable[32]={1024,1000,930,823,693,556,429,325,
256,226,233,272,331,397,457,497,512,497,457,397,331,272,233,226,
256,325,429,556,693,823,930,1000};//LUT for combined 1 kHz + 2
kHz signal
signed int32 Multi_1coscos;// multiplication of collected data and
local oscillator of 1kHz cosine wave
signed int32 Multi_1cossin;// multiplication of collected data and
local oscillator of 1kHz sine wave
signed int32 Multi_2coscos;// multiplication of collected data and
local oscillator of 2kHz cosine wave
signed int32 Multi_2cossin;// multiplication of collected data and
local oscillator of 2kHz sine wave
float cos_1k_scaled;//cosine part for 1K signal
float sin_1k_scaled;//sine part for 1K signal
float cos_2k_scaled;//cosine part for 2K signal
float sin_2k_scaled;//sine part for 2K signal
int Look_Up_Table_Index=0;
int i;
/***************RDA_interrupt***************/
#INT_RDA
void rda_isr(void)
{
   pause=1;
   CommandString[0]=0;//reset CommandString
   in=0;
   do
   {
      CommandString[in]=getc();
      putc(CommandString[in]);
      if(CommandString[in]==127)//backspace check
      {
         in=in-2;
      }
      in=in+1;
   }
   while((in<31)&&(CommandString[in-1]!=13));
   CommandString[in-1]=0;
   putc(13);//enter
   putc(10);//back to first column
   //ERROR JUDGEMENT//
   if(STRICMP(CommandString,collectdata)!=0) puts("ERROR");
```

```
}
/***************Timer2_interrupt***************/
#int_timer2
void Timer2_Service_Routine(void)
{
    Port.cs = 0b0;//SPI Chip select signal low
    spi_xfer((CosTable[Look_Up_Table_Index])>>8); //High byte
(+4096(2^12) for SHDN=1)
    spi_xfer((CosTable[Look_Up_Table_Index])&0x00FF);// Low byte
    Port.cs = 0b1;//SPI Chip select signal high
    Look_Up_Table_Index=++Look_Up_Table_Index % 32;//if already
count to 32, then reset to 0
}
/***************ADC_interrupt***************/
#INT_AD
void adc_isr(void)
{
    if(count<128)
    {
        adctable[count] = read_adc(ADC_READ_ONLY)-128;
        /**int8 multiply**/
        signed int8 ADCdata,ARG1,ARG2,ARG3,ARG4; // ARG1=
Local_cos_1k, ARG2=Local_sin_1k,
ARG2=Local_cos_2k,ARG4=Local_sin_2k
        int8 RES1H,RES1L,RES2H,RES2L,RES3H,RES3L,RES4H,RES4L;
//RES1H=High 8 bit for result1, RES1L=Low 8 bits for result2
        register _PRODH int8 PRODH;// high 8 bits for product result
        register _PRODL int8 PRODL;// low 8 bits for product result
        if(count==0) {ARG1=64;ARG2=0;ARG3=64;ARG4=0;}//store
LUT for local oscillators into ARGs for multipilcation
        if(count==1) {ARG1=63;ARG2=-12;ARG3=59;ARG4=-24;}
        if(count==2) {ARG1=59;ARG2=-24;ARG3=45;ARG4=-45;}
        if(count==3) {ARG1=53;ARG2=-36;ARG3=24;ARG4=-59;}
        if(count==4) {ARG1=45;ARG2=-45;ARG3=0;ARG4=-64;}
        if(count==5) {ARG1=36;ARG2=-53;ARG3=-24;ARG4=-59;}
        if(count==6) {ARG1=24;ARG2=-59;ARG3=-45;ARG4=-45;}
        if(count==7) {ARG1=12;ARG2=-63;ARG3=-59;ARG4=-24;}
        if(count==8) {ARG1=0;ARG2=-64;ARG3=-64;ARG4=0;}
        if(count==9) {ARG1=-12;ARG2=-63;ARG3=-59;ARG4=24;}
        if(count==10) {ARG1=-24;ARG2=-59;ARG3=-45;ARG4=45;}
        if(count==11) {ARG1=-36;ARG2=-53;ARG3=-24;ARG4=59;}
        if(count==12) {ARG1=-45;ARG2=-45;ARG3=0;ARG4=64;}
        if(count==13) {ARG1=-53;ARG2=-36;ARG3=24;ARG4=59;}
        if(count==14) {ARG1=-59;ARG2=-24;ARG3=45;ARG4=45;}
        if(count==15) {ARG1=-63;ARG2=-12;ARG3=59;ARG4=24;}
        if(count==16) {ARG1=-64;ARG2=0;ARG3=64;ARG4=0;}
        if(count==17) {ARG1=-63;ARG2=12;ARG3=59;ARG4=-24;}
        if(count==18) {ARG1=-59;ARG2=24;ARG3=45;ARG4=-45;}
        if(count==19) {ARG1=-53;ARG2=36;ARG3=24;ARG4=-59;}
if(count==20) {ARG1=-45;ARG2=45;ARG3=0;ARG4=-64;}
if(count==21) {ARG1=-36;ARG2=53;ARG3=-24;ARG4=-59;}
if(count==22) {ARG1=-24;ARG2=59;ARG3=-45;ARG4=-45;}
if(count==23) {ARG1=-12;ARG2=63;ARG3=-59;ARG4=-24;}
if(count==24) {ARG1=0;ARG2=64;ARG3=-64;ARG4=0;}
if(count==25) {ARG1=12;ARG2=63;ARG3=-59;ARG4=24;}
if(count==26) {ARG1=24;ARG2=59;ARG3=-45;ARG4=45;}
if(count==27) {ARG1=36;ARG2=53;ARG3=-24;ARG4=59;}
if(count==28) {ARG1=45;ARG2=45;ARG3=0;ARG4=64;}
if(count==29) {ARG1=53;ARG2=36;ARG3=24;ARG4=59;}
if(count==30) {ARG1=59;ARG2=24;ARG3=45;ARG4=45;}
if(count==31) {ARG1=63;ARG2=12;ARG3=59;ARG4=24;}
if(count==32) {ARG1=64;ARG2=0;ARG3=64;ARG4=0;}
if(count==33) {ARG1=63;ARG2=-12;ARG3=59;ARG4=-24;}
if(count==34) {ARG1=59;ARG2=-24;ARG3=45;ARG4=-45;}
if(count==35) {ARG1=53;ARG2=-36;ARG3=24;ARG4=-59;}
if(count==36) {ARG1=45;ARG2=-45;ARG3=0;ARG4=-64;}
if(count==37) {ARG1=36;ARG2=-53;ARG3=-24;ARG4=-59;}
if(count==38) {ARG1=24;ARG2=-59;ARG3=-45;ARG4=-45;}
if(count==39) {ARG1=12;ARG2=-63;ARG3=-59;ARG4=-24;}
if(count==40) {ARG1=0;ARG2=-64;ARG3=-64;ARG4=0;}
if(count==41) {ARG1=-12;ARG2=-63;ARG3=-59;ARG4=24;}
if(count==42) {ARG1=-24;ARG2=-59;ARG3=-45;ARG4=45;}
if(count==43) {ARG1=-36;ARG2=-53;ARG3=-24;ARG4=59;}
if(count==44) {ARG1=-45;ARG2=-45;ARG3=0;ARG4=64;}
if(count==45) {ARG1=-53;ARG2=-36;ARG3=24;ARG4=59;}
if(count==46) {ARG1=-59;ARG2=-24;ARG3=45;ARG4=45;}
if(count==47) {ARG1=-63;ARG2=-12;ARG3=59;ARG4=24;}
if(count==48) {ARG1=-64;ARG2=0;ARG3=64;ARG4=0;}
if(count==49) {ARG1=-63;ARG2=12;ARG3=59;ARG4=-24;}
if(count==50) {ARG1=-59;ARG2=24;ARG3=45;ARG4=-45;}
if(count==51) {ARG1=-53;ARG2=36;ARG3=24;ARG4=-59;}
if(count==52) {ARG1=-45;ARG2=45;ARG3=0;ARG4=-64;}
if(count==53) {ARG1=-36;ARG2=53;ARG3=-24;ARG4=-59;}
if(count==54) {ARG1=-24;ARG2=59;ARG3=-45;ARG4=-45;}
if(count==55) {ARG1=-12;ARG2=63;ARG3=-59;ARG4=-24;}
if(count==56) {ARG1=0;ARG2=64;ARG3=-64;ARG4=0;}
if(count==57) {ARG1=12;ARG2=63;ARG3=-59;ARG4=24;}
if(count==58) {ARG1=24;ARG2=59;ARG3=-45;ARG4=45;}
if(count==59) {ARG1=36;ARG2=53;ARG3=-24;ARG4=59;}
if(count==60) {ARG1=45;ARG2=45;ARG3=0;ARG4=64;}
if(count==61) {ARG1=53;ARG2=36;ARG3=24;ARG4=59;}
if(count==62) {ARG1=59;ARG2=24;ARG3=45;ARG4=45;}
if(count==63) {ARG1=63;ARG2=12;ARG3=59;ARG4=24;}
if(count==64) {ARG1=64;ARG2=0;ARG3=64;ARG4=0;}
if(count==65) {ARG1=63;ARG2=-12;ARG3=59;ARG4=-24;}
if(count==66) {ARG1=59;ARG2=-24;ARG3=45;ARG4=-45;}
if(count==67) {ARG1=53;ARG2=-36;ARG3=24;ARG4=-59;}
if(count==68) {ARG1=45;ARG2=-45;ARG3=0;ARG4=-64;}
if(count==69) {ARG1=36;ARG2=-53;ARG3=-24;ARG4=-59;}
if(count==70) {ARG1=24;ARG2=-59;ARG3=-45;ARG4=-45;}
if(count==71) {ARG1=12;ARG2=-63;ARG3=-59;ARG4=-24;}
if(count==72) {ARG1=0;ARG2=-64;ARG3=-64;ARG4=0;}
if(count==73) {ARG1=-12;ARG2=-63;ARG3=-59;ARG4=24;}
if(count==74) {ARG1=-24;ARG2=-59;ARG3=-45;ARG4=45;}
if(count==75) {ARG1=-36;ARG2=-53;ARG3=-24;ARG4=59;}
if(count==76) {ARG1=-45;ARG2=-45;ARG3=0;ARG4=64;}
if(count==77) {ARG1=-53;ARG2=-36;ARG3=24;ARG4=59;}
if(count==78) {ARG1=-59;ARG2=-24;ARG3=45;ARG4=45;}
if(count==79) {ARG1=-63;ARG2=-12;ARG3=59;ARG4=24;}
if(count==80) {ARG1=-64;ARG2=0;ARG3=64;ARG4=0;}
if(count==81) {ARG1=-63;ARG2=12;ARG3=59;ARG4=-24;}
if(count==82) {ARG1=-59;ARG2=24;ARG3=45;ARG4=-45;}
if(count==83) {ARG1=-53;ARG2=36;ARG3=24;ARG4=-59;}
if(count==84) {ARG1=-45;ARG2=45;ARG3=0;ARG4=-64;}
if(count==85) {ARG1=-36;ARG2=53;ARG3=-24;ARG4=-59;}
if(count==86) {ARG1=-24;ARG2=59;ARG3=-45;ARG4=-45;}
if(count==87) {ARG1=-12;ARG2=63;ARG3=-59;ARG4=-24;}
if(count==88) {ARG1=0;ARG2=64;ARG3=-64;ARG4=0;}
if(count==89) {ARG1=12;ARG2=63;ARG3=-59;ARG4=24;}
if(count==90) {ARG1=24;ARG2=59;ARG3=-45;ARG4=45;}
if(count==91) {ARG1=36;ARG2=53;ARG3=-24;ARG4=59;}
if(count==92) {ARG1=45;ARG2=45;ARG3=0;ARG4=64;}
if(count==93) {ARG1=53;ARG2=36;ARG3=24;ARG4=59;}
if(count==94) {ARG1=59;ARG2=24;ARG3=45;ARG4=45;}
if(count==95) {ARG1=63;ARG2=12;ARG3=59;ARG4=24;}
if(count==96) {ARG1=64;ARG2=0;ARG3=64;ARG4=0;}
if(count==97) {ARG1=63;ARG2=-12;ARG3=59;ARG4=-24;}
if(count==98) {ARG1=59;ARG2=-24;ARG3=45;ARG4=-45;}
if(count==99) {ARG1=53;ARG2=-36;ARG3=24;ARG4=-59;}
if(count==100) {ARG1=45;ARG2=-45;ARG3=0;ARG4=-64;}
if(count==101) {ARG1=36;ARG2=-53;ARG3=-24;ARG4=-59;}
if(count==102) {ARG1=24;ARG2=-59;ARG3=-45;ARG4=-45;}
if(count==103) {ARG1=12;ARG2=-63;ARG3=-59;ARG4=-24;}
if(count==104) {ARG1=0;ARG2=-64;ARG3=-64;ARG4=0;}
if(count==105) {ARG1=-12;ARG2=-63;ARG3=-59;ARG4=24;}
if(count==106) {ARG1=-24;ARG2=-59;ARG3=-45;ARG4=45;}
if(count==107) {ARG1=-36;ARG2=-53;ARG3=-24;ARG4=59;}
if(count==108) {ARG1=-45;ARG2=-45;ARG3=0;ARG4=64;}
if(count==109) {ARG1=-53;ARG2=-36;ARG3=24;ARG4=59;}
if(count==110) {ARG1=-59;ARG2=-24;ARG3=45;ARG4=45;}
if(count==111) {ARG1=-63;ARG2=-12;ARG3=59;ARG4=24;}
if(count==112) {ARG1=-64;ARG2=0;ARG3=64;ARG4=0;}
if(count==113) {ARG1=-63;ARG2=12;ARG3=59;ARG4=-24;}
if(count==114) {ARG1=-59;ARG2=24;ARG3=45;ARG4=-45;}
if(count==115) {ARG1=-53;ARG2=36;ARG3=24;ARG4=-59;}
if(count==116) {ARG1=-45;ARG2=45;ARG3=0;ARG4=-64;}
if(count==117) {ARG1=-36;ARG2=53;ARG3=-24;ARG4=-59;}
if(count==118) {ARG1=-24;ARG2=59;ARG3=-45;ARG4=-45;}
if(count==119) {ARG1=-12;ARG2=63;ARG3=-59;ARG4=-24;}
if(count==120) {ARG1=0;ARG2=64;ARG3=-64;ARG4=0;}
if(count==121) {ARG1=12;ARG2=63;ARG3=-59;ARG4=24;}
if(count==122) {ARG1=24;ARG2=59;ARG3=-45;ARG4=45;}
if(count==123) {ARG1=36;ARG2=53;ARG3=-24;ARG4=59;}
if(count==124) {ARG1=45;ARG2=45;ARG3=0;ARG4=64;}
if(count==125) {ARG1=53;ARG2=36;ARG3=24;ARG4=59;}
if(count==126) {ARG1=59;ARG2=24;ARG3=45;ARG4=45;}
if(count==127) {ARG1=63;ARG2=12;ARG3=59;ARG4=24;}
ADCdata=adctable[count];
/**int8 multiply**/
#asm
//Local_cos_1k*ADCdata
MOVF ADCdata, W; move ADCdata to W
MULWF ARG1 ; ADCdata * ARG1 -> PRODH:PRODL
BTFSC ARG1, 7 ; Test Sign Bit
SUBWF PRODH, F ; PRODH = PRODH - ADCdata
MOVF ARG1, W
BTFSC ADCdata, 7 ; Test Sign Bit
SUBWF PRODH, F ; PRODH = PRODH - ADCdata
//Save Local_cos_1k*ADCdata in RES1
MOVFF PRODH, RES1H ;
MOVFF PRODL, RES1L ;
;
//Local_sin_1k*ADCdata
MOVF ADCdata, W
MULWF ARG2 ; ADCdata * ARG2 -> PRODH:PRODL
BTFSC ARG2, 7 ; Test Sign Bit
SUBWF PRODH, F ; PRODH = PRODH - ADCdata
MOVF ARG2, W
BTFSC ADCdata, 7 ; Test Sign Bit
SUBWF PRODH, F ; PRODH = PRODH - ADCdata
MOVFF PRODH, RES2H ;
MOVFF PRODL, RES2L ;
;
//Local_cos_2k*ADCdata
MOVF ADCdata, W
MULWF ARG3 ; ADCdata * ARG3 -> PRODH:PRODL
BTFSC ARG3, 7 ; Test Sign Bit
SUBWF PRODH, F ; PRODH = PRODH - ADCdata
MOVF ARG3, W
BTFSC ADCdata, 7 ; Test Sign Bit
SUBWF PRODH, F ; PRODH = PRODH - ADCdata
MOVFF PRODH, RES3H ;
MOVFF PRODL, RES3L ;
;
//Local_sin_2k*ADCdata
MOVF ADCdata, W
```

```
        MULWF ARG4 ; ADCdata * ARG4 -> PRODH:PRODL
        BTFSC ARG4, 7 ; Test Sign Bit
        SUBWF PRODH, F ; PRODH = PRODH - ADCdata
        MOVF ARG4, W
        BTFSC ADCdata, 7 ; Test Sign Bit
        SUBWF PRODH, F ; PRODH = PRODH - ADCdata
        MOVFF PRODH, RES4H ;
        MOVFF PRODL, RES4L ;
        ;
        #endasm
        // combine high bits and low bits together and accumulate
        Multi_1coscos = Multi_1coscos+ (signed int32)((signed
int16)make16(RES1H,RES1L));
        Multi_1cossin = Multi_1cossin+ (signed int32)((signed
int16)make16(RES2H,RES2L));
        Multi_2coscos = Multi_2coscos+ (signed int32)((signed
int16)make16(RES3H,RES3L));
        Multi_2cossin = Multi_2cossin+ (signed int32)((signed
int16)make16(RES4H,RES4L));
        /**int8 multiply**/
        count++;
    }
}
/***************main_function***************/
void main()
{
    //Port Setting//
    int BWPU;//weak pull up PIN_B
    #byte BWPU = 0x0F18;
    BWPU = 0b11111111;
    PortDirection.PWM=0b0;
    PortDirection.ADC=0b1;
    PortDirection.ts=0b0;
    PortDirection.rc=0b1;
    PortDirection.cs=0b0;
    PortDirection.SDO=0b0;
    PortDirection.SCK=0b0;
    //RDA//
    enable_interrupts(INT_RDA);
    //TIMER2//
    setup_timer_2(T2_CLK_INTERNAL|T2_DIV_BY_2,249,1);
    enable_interrupts(INT_TIMER2);// Timer 2 interrupt enable
    //PWM//
    setup_ccp2(CCP_PWM|CCP_USE_TIMER1_AND_TIMER2);
    setup_pwm4(PWM_ENABLED|PWM_ACTIVE_LOW|PWM_TIMER2);
    set_pwm4_duty(64);//active low for 1us
    //ADC//
    setup_adc_ports(sAN7,VSS_FVR);
    setup_adc(ADC_LEGACY_MODE|ADC_CLOCK_DIV_128);
    setup_vref(VREF_ON|VREF_ADC_1v024);
    set_adc_channel(7);
    set_adc_trigger(ADC_TRIGGER_TIMER2);
    enable_interrupts(INT_AD);
    //GLOBAL//
    enable_interrupts(GLOBAL);

    for(i=0;i<32;i++) CosTable[i]=CosTable[i]+4096;
    while(1)
    {
/**************ADC CONTROL*****************/
        if (STRICMP(CommandString,collectdata)==0)
        {
            puts("OK");
            pause=0;
            count=0;
            Multi_1coscos=0; Multi_1cossin=0;
             Multi_2coscos=0; Multi_2cossin=0;
            while(pause==0)
            {
                if(count==128)
                {
                    long ii;
                    printf("[");
                    for(ii=0;ii<128;ii++) if(ii<128) printf("%d ",adctable[ii]);
            }
                    printf("];");
                    putc(13);
                    putc(10);
                    putc(13);
                    putc(10);
                    cos_1k_scaled = (float)Multi_1coscos/370727;
                    sin_1k_scaled = (float)Multi_1cossin/370727;
                    cos_2k_scaled = (float)Multi_2coscos/370727;
                    sin_2k_scaled = (float)Multi_2cossin/370727;
                    float sign_direction= (cos_1k_scaled^2-
                  sin_1k_scaled ^2)* cos_2k_scaled-2* cos_1k_scaled*
                  sin_1k_scaled*(- sin_2k_scaled);
                    printf("%ld%ld%ld%ld\n\r",Multi_1coscos,Multi_1coss
                  in,Multi_2coscos,Multi_2cossin);
                    printf("1kreal: %f      1kimag: %f      2kreal: %f
2kimag: %f",cos_1k_scaled,sin_1k_scaled,cos_2k_scaled,sin_2k_scaled);
                    putc(13); putc(10);pause=1;
                }
             }
        CommandString[0]=0;//reset CommandString
        }
    }
}
```