**附件1**

**LDC初始化**

#ifndef LDC1314\_CMD\_H\_

#define LDC1314\_CMD\_H\_

#include "derivative.h"

//FLOAT LDC COMMANDS

#define LDC1314\_CMD\_REVID 0x00

#define LDC1314\_CMD\_RPMAX 0x01

#define LDC1314\_CMD\_RPMIN 0x02

#define LDC1314\_CMD\_SENSORFREQ 0x03 //谐振频率

#define LDC1314\_CMD\_LDCCONFIG 0x04

#define LDC1314\_CMD\_CLKCONFIG 0x05

#define LDC1314\_CMD\_THRESHILSB 0x06

#define LDC1314\_CMD\_THRESHIMSB 0x07

#define LDC1314\_CMD\_THRESLOLSB 0x08

#define LDC1314\_CMD\_THRESLOMSB 0x09

#define LDC1314\_CMD\_INTCONFIG 0x0A

#define LDC1314\_CMD\_PWRCONFIG 0x0B

#define LDC1314\_CMD\_STATUS 0x20

#define LDC1314\_CMD\_PROXLSB 0x21

#define LDC1314\_CMD\_PROXMSB 0x22

#define LDC1314\_CMD\_FREQCTRLSB 0x23

#define LDC1314\_CMD\_FREQCTRMID 0x24

#define LDC1314\_CMD\_FREQCTRMSB 0x25

//FLOAT LDC BITMASKS

#define LDC1314\_BIT\_AMPLITUDE 0x18

#define LDC1314\_BIT\_RESPTIME 0x07

#define LDC1314\_BIT\_CLKSEL 0x02

#define LDC1314\_BIT\_CLKPD 0x01

#define LDC1314\_BIT\_INTMODE 0x07

#define LDC1314\_BIT\_PWRMODE 0x01

#define LDC1314\_BIT\_STATUSOSC 0x80

#define LDC1314\_BIT\_STATUSDRDYB 0x40

#define LDC1314\_BIT\_STATUSWAKEUP 0x20

#define LDC1314\_BIT\_STATUSCOMP 0x10

void FLOAT\_LDC\_init()

{

while(orgVal[1]!=RPMAX||orgVal[2]!=RPMIN||orgVal[3]!=RFREQ)//一旦在此循环说明初始化不成功

{

FLOAT\_SPI\_init();

FLOAT\_delay\_us(3000);

FLOAT\_Singal\_SPI\_Write(LDC1314\_CMD\_RPMAX, RPMAX);

FLOAT\_Singal\_SPI\_Write(LDC1314\_CMD\_RPMIN, RPMIN);//0x14

FLOAT\_Singal\_SPI\_Write(LDC1314\_CMD\_SENSORFREQ, RFREQ); //谐振频率计算方法见《浮点科技电轨传感器调试手册》

FLOAT\_Singal\_SPI\_Write(LDC1314\_CMD\_LDCCONFIG, 0x17); /\*\*\*\*\*\*\*\*/

FLOAT\_Singal\_SPI\_Write(LDC1314\_CMD\_CLKCONFIG, 0x00); //L配置LDC1314的输出速率

FLOAT\_Singal\_SPI\_Write(LDC1314\_CMD\_THRESHILSB, 0x50); /\*\*\*\*\*\*\*\*/

FLOAT\_Singal\_SPI\_Write(LDC1314\_CMD\_THRESHIMSB, 0x14);

FLOAT\_Singal\_SPI\_Write(LDC1314\_CMD\_THRESLOLSB, 0xC0);

FLOAT\_Singal\_SPI\_Write(LDC1314\_CMD\_THRESLOMSB, 0x12);

FLOAT\_Singal\_SPI\_Write(LDC1314\_CMD\_INTCONFIG, 0x02);

FLOAT\_Singal\_SPI\_Write(LDC1314\_CMD\_PWRCONFIG, 0x01);

FLOAT\_SPI\_Read\_Buf(LDC1314\_CMD\_REVID,&orgVal[0],12);//orgVal[]对应上面写入的值说明初始化正常

}

}

int ldc\_read\_avr()

{

char rpi=0; //取rpi次平均值

for (rpi=0;rpi<rpi\_max;rpi++)

{

FLOAT\_SPI\_Read\_Buf(LDC1314\_CMD\_PROXLSB,&proximtyData[0],2);

proximtyDataTEMP = ((unsigned char)proximtyData[1]<<8) + proximtyData [0];

proximtyDataSUM += proximtyDataTEMP;

if (proximtyDataTEMP < proximtyDataMIN) //在100个proximtyDataTEMP中取最大，最小

proximtyDataMIN = proximtyDataTEMP;

if (proximtyDataTEMP > proximtyDataMAX)

proximtyDataMAX = proximtyDataTEMP;

}

proximtyDataAVE = proximtyDataSUM /rpi\_max;

proximtyDataSUM=0;

proximtyDataAVE\_LAS=proximtyDataAVE;

return proximtyDataAVE;

}

long int filter()

{

char count,i,j,count1;

char count2=0;

long int temp;

long int sum=0;

for(count=0;count<NN;count++)

{

value\_buf[count] = ldc\_read\_avr();

}

for(count1=0;count1<NN;count1++)

{

if(value\_buf[count1]<32768)

{

new\_value\_buf[count2]=value\_buf[count1];

count2++;

}

}

for (j=0;j<count2-1;j++)

{

for (i=0;i<count2-j;i++)

{

if ( new\_value\_buf[i]>new\_value\_buf[i+1] )

{

temp = new\_value\_buf[i];

new\_value\_buf[i] = new\_value\_buf[i+1];

new\_value\_buf[i+1] = temp;

}

}

}

for(count=1;count<count2-1;count++)

{

sum += new\_value\_buf[count];

}

return (long int)(sum/(count2-2));

}

**附件2**

**SPI初始化**

void FLOAT\_SPI\_init()

{

DDRA=0x0E;//MISO

// gpio\_init (PTD1 ,GPO,1);//MOSI

// gpio\_init (PTD2, GPO,1);// CSN

// gpio\_init (PTD3, GPO,0);//SCK

CSN\_H;

SCK\_L;

MOSI\_H;

}

unsigned char FLOAT\_SPI\_RW(unsigned char rwdata)

{

unsigned char spi\_rw\_i=0;

unsigned char temp=0;

for(spi\_rw\_i=0;spi\_rw\_i<8;spi\_rw\_i++) // output 8-bit

{

/\*\*\* prepare the write data of read before the coming of rising up\*\*\*\*\*\*/

if(rwdata & 0x80)

MOSI\_H;

else

MOSI\_L;

rwdata<<=1; // shift next bit to MSB

temp<<=1;

SCK\_H; //Set SCK high Rising up

if(MISO)

temp|=1;

SCK\_L; //set SCK low Falling down

}

return(temp); // return read byte

}

unsigned char FLOAT\_Singal\_SPI\_Read(unsigned char reg)

{

unsigned char rdata;

CSN\_L; // CSN low, initialize SPI communication...

FLOAT\_delay\_us(2);

reg=reg|0x80; //read com

FLOAT\_SPI\_RW(reg); // Select register to read from..

asm("nop");

asm("nop");

asm("nop");

asm("nop");

asm("nop");

asm("nop");

asm("nop");

asm("nop");

asm("nop");

asm("nop");

asm("nop");

asm("nop");

asm("nop");

asm("nop");

asm("nop");

asm("nop");

asm("nop");

asm("nop");

asm("nop");

asm("nop");

asm("nop");

rdata = FLOAT\_SPI\_RW(0); // then read registervalue

// FLOAT\_delay\_us(1700);

CSN\_H; // CSN high, terminate SPI communication

return rdata; // return register value

}

void FLOAT\_Singal\_SPI\_Write(unsigned char reg,unsigned char wdata)

{

CSN\_L; // CSN low, initialize SPI communication...

FLOAT\_delay\_us(2);//2us

reg=reg&~0x80;

FLOAT\_SPI\_RW(reg); // Select register to read from..

asm("nop");

asm("nop");

asm("nop");

asm("nop");

asm("nop");

asm("nop");

asm("nop");

asm("nop");

asm("nop");

asm("nop");

asm("nop");

asm("nop");

asm("nop");

asm("nop");

asm("nop");

asm("nop");

asm("nop");

asm("nop");

asm("nop");

asm("nop");

asm("nop");

FLOAT\_SPI\_RW(wdata); // ..then read registervalue

FLOAT\_delay\_us(1700);//875us

CSN\_H; // CSN high, terminate SPI communication

}

void FLOAT\_SPI\_Read\_Buf(unsigned char reg, unsigned char \*pBuf, unsigned char len)

{

unsigned char spi\_rw\_i;

CSN\_L; // Set CSN low, init SPI tranaction

reg=reg|0x80; //read

FLOAT\_SPI\_RW(reg); // Select register to write to and read status unsigned char

for(spi\_rw\_i=0;spi\_rw\_i<len;spi\_rw\_i++)

{

pBuf[spi\_rw\_i] = FLOAT\_SPI\_RW(0);

}

CSN\_H;

}