#include <SoftwareSerial.h>

//#include <Serial.h>

// P1 and P2 threshold values

#define P1\_THR\_0 0xB0

#define P1\_THR\_1 0x88

#define P1\_THR\_2 0x88

#define P1\_THR\_3 0x88

#define P1\_THR\_4 0x88

#define P1\_THR\_5 0x88

#define P1\_THR\_6 0xF9

#define P1\_THR\_7 0xCE

#define P1\_THR\_8 0x73

#define P1\_THR\_9 0x9C

#define P1\_THR\_10 0xE7

#define P1\_THR\_11 0x3F

#define P1\_THR\_12 0x3F

#define P1\_THR\_13 0x3F

#define P1\_THR\_14 0x3F

#define P1\_THR\_15 0x07

#define P2\_THR\_0 0xB0

#define P2\_THR\_1 0x88

#define P2\_THR\_2 0x88

#define P2\_THR\_3 0x88

#define P2\_THR\_4 0x88

#define P2\_THR\_5 0x88

#define P2\_THR\_6 0xF9

#define P2\_THR\_7 0xCE

#define P2\_THR\_8 0x73

#define P2\_THR\_9 0x9C

#define P2\_THR\_10 0xE7

#define P2\_THR\_11 0x3F

#define P2\_THR\_12 0x3F

#define P2\_THR\_13 0x3F

#define P2\_THR\_14 0x3F

#define P2\_THR\_15 0x07

#define syncByte 0x55 // Synch byte

#define THRBW 0x80 // Threshold bulk write command

#define SRW 0x50 // Serial write command

#define AFEGAINRANGE 0x0F // AFEGAINRANGE ,holds range Ox0F

#define TVGBW 0x70 // Threshold voltage gain bulk write

#define TVGAIN0 0x88 // TVG commands

#define TVGAIN1 0x88 //TVG commands

#define TVGAIN2 0x88 //TVG commands

#define TVGAIN3 0x41 //TVG commands

#define TVGAIN4 0x04 //TVG commands

#define TVGAIN5 0x10 //TVG commands

#define TVGAIN6 0x40 //TVG commands

#define P1BL 0x00 //Preset and busrt listen command

#define numObj 0x01 //To define object number

//#define SD 0x40

#define UMR 0x28 //Ultrasonic measurement command

#define DECPL\_TEMP 0x26

unsigned char calcChecksum(unsigned char); // calcchecksum function

unsigned char ChecksumInput[35]; //checksum array to hold values

unsigned char THBUFF[35]; //Threshold buffer array for P1 and P2

unsigned char AFEGAIN[5]; //AFEGAIN array

unsigned char cmd; //cmd denotes command in checksum function

unsigned char TVG[10]; //Threshold voltage gain array

unsigned char PRE1\_BUR\_LIS[4]; //Preset and burst listen array

unsigned char ULTRA\_MEAS[3]; //Ultrasonic measurement array

unsigned char UMRData[35]; //holds Ultrasonic measurement data i:e data,width,amplitude

unsigned char byte1; //Denotes Object distance byte1+byte2

unsigned char byte2;

unsigned char byte3; //Object width measurement

unsigned char byte4; //Peak amplitude measurement

//SoftwareSerial Serial(10,11); //Rx, Tx

void setup()

{

// put your setup code here, to run once:

//Serial.begin(9600); // Begin Serial communication

Serial.begin(19200, SERIAL\_8N2);

// CONFIGURATION SETTINGS

// Initialise Threshold values to an array

THBUFF[0]=syncByte;

THBUFF[1]=THRBW;

THBUFF[2]=P1\_THR\_0;

THBUFF[3]=P1\_THR\_1;

THBUFF[4]=P1\_THR\_2;

THBUFF[5]=P1\_THR\_3;

THBUFF[6]=P1\_THR\_4;

THBUFF[7]=P1\_THR\_5;

THBUFF[8]=P1\_THR\_6;

THBUFF[9]=P1\_THR\_7;

THBUFF[10]=P1\_THR\_8;

THBUFF[11]=P1\_THR\_9;

THBUFF[12]=P1\_THR\_10;

THBUFF[13]=P1\_THR\_11;

THBUFF[14]=P1\_THR\_12;

THBUFF[15]=P1\_THR\_13;

THBUFF[16]=P1\_THR\_14;

THBUFF[17]=P1\_THR\_15;

THBUFF[18]=P2\_THR\_0;

THBUFF[19]=P2\_THR\_1;

THBUFF[20]=P2\_THR\_2;

THBUFF[21]=P2\_THR\_3;

THBUFF[22]=P2\_THR\_4;

THBUFF[23]=P2\_THR\_5;

THBUFF[24]=P2\_THR\_6;

THBUFF[25]=P2\_THR\_7;

THBUFF[26]=P2\_THR\_8;

THBUFF[27]=P2\_THR\_9;

THBUFF[28]=P2\_THR\_10;

THBUFF[29]=P2\_THR\_11;

THBUFF[30]=P2\_THR\_12;

THBUFF[31]=P2\_THR\_13;

THBUFF[32]=P2\_THR\_14;

THBUFF[33]=P2\_THR\_15;

THBUFF[34]= calcChecksum(THRBW);

// Initialise AFEGAIN values in array

AFEGAIN[0]=syncByte;

AFEGAIN[1]=SRW;

AFEGAIN[2]=DECPL\_TEMP;

AFEGAIN[3]=AFEGAINRANGE;

AFEGAIN[4]=calcChecksum(SRW);

// Initialise TVG values in array

TVG[0]=syncByte;

TVG[1]=TVGBW;

TVG[2]=TVGAIN0;

TVG[3]=TVGAIN1;

TVG[4]=TVGAIN2;

TVG[5]=TVGAIN3;

TVG[6]=TVGAIN4;

TVG[7]=TVGAIN5;

TVG[8]=TVGAIN6;

TVG[9]=calcChecksum(TVGBW);

//ISSUE P1BL to initiate burst pulse to measure Real time of flight

PRE1\_BUR\_LIS[0]=syncByte;

PRE1\_BUR\_LIS[1]=P1BL;

PRE1\_BUR\_LIS[2]=numObj;

PRE1\_BUR\_LIS[3]=calcChecksum(P1BL);

//ULTRASONIC MEASUREMENT RESULT

ULTRA\_MEAS[0]=syncByte;

ULTRA\_MEAS[1]=UMR;

ULTRA\_MEAS[2]=calcChecksum(UMR);

byte1=UMRData[0]; //Object distance byte1+byte2

byte2=UMRData[1];

byte3=UMRData[2]; //Object width measurement

byte4=UMRData[3]; //Peak amplitude measure

}

void loop() {

// delay of 4 seconds

delay(4000);

Serial.write(THBUFF, 35); // Write THRESHOLD BULK WRITE //BURST LISTEN SETTINGS COMMIT

delay(50); // 50mS delay

Serial.write(AFEGAIN,5); //Write AFEGAIN //RECEIVE SIGNAL SETTINGS COMMIT

delay(50); // 50mS delay

Serial.write(TVG, 10); //Write TVGAIN //AMPLIFY RECEIVED SIGNAL SETTINGS COMMIT

delay(50); // 50mS delay

Serial.println("Initialization done");

// issue burst pulse and read UMR value continuously

while(1u)

{

Serial.write(PRE1\_BUR\_LIS,4); //Issue preset and burst listen //CHOOSING PRESET1 BURST LISTEN

delay(50); // delay 50mS

Serial.write(ULTRA\_MEAS,3); //Issue ultrasonic measurement command

Serial.println("Pinged US for data");

if(Serial.available()> 0) // check if data received in the arduino UART buffer //CHECKING IF THERE’S DATA

{

//Serial.println("Data received");

byte1 = Serial.read(); //Reads object distance

byte2 = Serial.read(); //Reads object distance

byte3 = Serial.read(); //Reads object width

byte4 = Serial.read(); //Reads object peak amplitude

Serial.print(byte1);

Serial.print(" ");

Serial.print(byte2);

Serial.print(" ");

Serial.print(byte3);

Serial.print(" ");

Serial.print(byte4);

Serial.println(" ");

}

}

}

//FUNCTION FOR CALCULATING CHECKSUM

unsigned char calcChecksum(unsigned char cmd)

{

unsigned char checksumLoops =0;

unsigned char DataCount =0;

unsigned int carry=0;

switch(cmd)

{

case P1BL : //P1BL // PRESET 1 BURST AND LISTEN - NUMOBJECT IS 1 CHECKSUM

ChecksumInput[0] = cmd;

ChecksumInput[1] = numObj;

checksumLoops = 2;

break;

case UMR : //UMR // ULTRASONIC MEASUREMENT RESULT CHECKSUM

//case 2 : //SD //SYSTEM DIAGNOSTICS

ChecksumInput[0] = cmd;

checksumLoops = 1;

break;

case SRW : //RW //REGISTER WRITE CHECKSUM

ChecksumInput[0] = cmd;

ChecksumInput[1] = DECPL\_TEMP; //0x26

ChecksumInput[2] = AFEGAINRANGE; //gain\_range=0x0F

checksumLoops = 3;

break;

case TVGBW : //TVGBW //TIME VARYING GAIN BULK WRITE CHECKSUM

ChecksumInput[0] = cmd;

ChecksumInput[1] = TVGAIN0;

ChecksumInput[2] = TVGAIN1;

ChecksumInput[3] = TVGAIN2;

ChecksumInput[4] = TVGAIN3;

ChecksumInput[5] = TVGAIN4;

ChecksumInput[6] = TVGAIN5;

ChecksumInput[7] = TVGAIN6;

checksumLoops = 8;

break;

case THRBW : //THRBW //THRESHOLD BULK WRITE CHECKSUM

ChecksumInput[0] = cmd;

ChecksumInput[1] = P1\_THR\_0;

ChecksumInput[2] = P1\_THR\_1;

ChecksumInput[3] = P1\_THR\_2;

ChecksumInput[4] = P1\_THR\_3;

ChecksumInput[5] = P1\_THR\_4;

ChecksumInput[6] = P1\_THR\_5;

ChecksumInput[7] = P1\_THR\_6;

ChecksumInput[8] = P1\_THR\_7;

ChecksumInput[9] = P1\_THR\_8;

ChecksumInput[10] = P1\_THR\_9;

ChecksumInput[11] = P1\_THR\_10;

ChecksumInput[12] = P1\_THR\_11;

ChecksumInput[13] = P1\_THR\_12;

ChecksumInput[14] = P1\_THR\_13;

ChecksumInput[15] = P1\_THR\_14;

ChecksumInput[16] = P1\_THR\_15;

ChecksumInput[17] = P2\_THR\_0;

ChecksumInput[18] = P2\_THR\_1;

ChecksumInput[19] = P2\_THR\_2;

ChecksumInput[20] = P2\_THR\_3;

ChecksumInput[21] = P2\_THR\_4;

ChecksumInput[22] = P2\_THR\_5;

ChecksumInput[23] = P2\_THR\_6;

ChecksumInput[24] = P2\_THR\_7;

ChecksumInput[25] = P2\_THR\_8;

ChecksumInput[26] = P2\_THR\_9;

ChecksumInput[27] = P2\_THR\_10;

ChecksumInput[28] = P2\_THR\_11;

ChecksumInput[29] = P2\_THR\_12;

ChecksumInput[30] = P2\_THR\_13;

ChecksumInput[31] = P2\_THR\_14;

ChecksumInput[32] = P2\_THR\_15;

checksumLoops = 33;

break;

default:

break;

}

carry = 0;

for ( DataCount = 0; DataCount < checksumLoops; DataCount++)

{

if ((ChecksumInput[DataCount] + carry) < carry)

{

carry = carry + ChecksumInput[DataCount] + 1;

}

else

{

carry = carry + ChecksumInput[DataCount];

}

if (carry > 0xFF)

{

carry = carry - 255;

}

}

carry = (~carry & 0x00FF);

return carry;

}