

**MAIN**

/\* ========================================

\*

\* Copyright YOUR COMPANY, THE YEAR

\* All Rights Reserved

\* UNPUBLISHED, LICENSED SOFTWARE.

\*

\* CONFIDENTIAL AND PROPRIETARY INFORMATION

\* WHICH IS THE PROPERTY OF your company.

\*

\* ========================================

\*/

#include "project.h"

#include "stdlib.h"

#include "stdbool.h"

#include "stdio.h"

//Function that is used to draw the stars from the array

void drawStars(int16 starArray[])

{

CyGlobalIntDisable;

int16 i=0;

for(i=0;i<300;i++)

{

GLCD\_DrawLine(starArray[i],starArray[i+1],starArray[i],starArray[i+1],GLCD\_WHITE);

}

CyGlobalIntEnable;

}

//Draws the scope

//X and Y is the center of the circle

//Color is the color of the scope

void drawScope(int16 x,int16 y,int32 color)

{

CyGlobalIntDisable;

GLCD\_DrawCircle(x,y,8,color);

GLCD\_DrawLine((x-15),(y),(x+15),y,color);

GLCD\_DrawLine(x,(y-15),x,(y+15),color);

CyGlobalIntEnable;

}

//Draws the blast

//X and Y is the center of the circle

//Size for normal is 0 and bigger is a greater specified number

void drawBlast(int16 x,int16 y,int16 size,int32 color,int16 starArray[])

{

CyGlobalIntDisable;

//Draws the disired color blast

int16 i=0;

for(i=0;i<(40+size);i++)

{

GLCD\_DrawCircle(x,y,i,color);

}

//Draws over in black

for(i=0;i<(40+size);i++)

{

GLCD\_DrawCircle(x,y,i,GLCD\_BLACK);

}

//Put stars back

drawStars(starArray);

//Put back the scope

drawScope(x,y,GLCD\_YELLOW);

CyGlobalIntEnable;

}

//Function to set the led color with r,g,b values

void ledSet(int16 r,int16 g, int16 b)

{

R\_LED\_Write(r);

G\_LED\_Write(g);

B\_LED\_Write(b);

}

void WriteSensor(uint8 \*WriteBuffer, int numvals)

{

I2C\_I2CMasterClearStatus();

I2C\_I2CMasterWriteBuf(0x53,WriteBuffer,numvals,I2C\_I2C\_WRITE\_XFER\_MODE);

while(!(I2C\_I2CMasterStatus()&I2C\_I2C\_MSTAT\_WR\_CMPLT));

}

void ReadSensor(uint8 \*ReadBuffer, int numvals)

{

I2C\_I2CMasterClearStatus();

I2C\_I2CMasterReadBuf(0x53,ReadBuffer,numvals,I2C\_I2C\_READ\_XFER\_MODE);

while(!(I2C\_I2CMasterStatus()&I2C\_I2C\_MSTAT\_RD\_CMPLT));

}

int16 stars [300]={0};

int16 twinkleStars=0;

int16 i=0;

int16 x=62;

int16 y=62;

int16 xOld=62;

int16 yOld=62;

uint32 counter=0;

bool blasterReady=true;

bool hyperblasterReady=true;

//For accel setup

uint8 PWR\_CTL=0x2D;

uint8 DATA\_FORMAT=0x31;

uint8 TAP\_AXES=0x2A;

uint8 INT\_ENABLE=0x2E;

uint8 INT\_MAP=0x2F;

uint8 INT\_SOURCE=0x30;

uint8 DATAX0=0x32;

uint8 DATAX1=0x33;

uint8 DATAY0=0x34;

uint8 DATAY1=0x35;

uint8 DATAZ0=0x36;

uint8 DATAZ1=0x37;

//For accel tap

uint8 THRESH\_TAP=0x1D;

uint8 DUR=0x21;

uint8 Latent=0x22;

uint8 Window=0x23;

//Buffers for i2C

uint8 ReadBuffer[10];

uint8 WriteBuffer[10];

//Variables for the strings for testing

char stringX[10];

char stringY[10];

char stringZ[10];

int main(void)

{

CyGlobalIntEnable;

//Turn stuff on

Timer\_Start();

I2C\_Start();

Clock\_1\_Start();

GLCD\_Start();

Backlight\_Write(1);

GLCD\_Clear(GLCD\_BLACK);

//Draw the startup screen

drawStars(stars);

drawScope(x,y,GLCD\_YELLOW);

ledSet(1,1,0); //Set to blue

//Turns on the accel

WriteBuffer[0] = PWR\_CTL;

WriteBuffer[1] = 0b00001000;

WriteSensor(WriteBuffer,2); //Write two bytes of data

//Set data format

WriteBuffer[0] = DATA\_FORMAT;

WriteBuffer[1] = 0b00000011;

WriteSensor(WriteBuffer,2); //Write two bytes of data

//Enable

WriteBuffer[0] = INT\_ENABLE;

WriteBuffer[1] = 0b01110000;

WriteSensor(WriteBuffer,2); //Write two bytes of data

//map values

WriteBuffer[0] = INT\_MAP;

WriteBuffer[1] = 0b00000000;

WriteSensor(WriteBuffer,2); //Write two bytes of data

//Set the tap threshold

WriteBuffer[0] = THRESH\_TAP;

WriteBuffer[1] = 0xc0;

WriteSensor(WriteBuffer,2); //Write two bytes of data

//Sets the tap threshold

WriteBuffer[0] = TAP\_AXES;

WriteBuffer[1] = 0b00000111;

WriteSensor(WriteBuffer,2); //Write two bytes of data

//Sets the tap duration

WriteBuffer[0] = DUR;

WriteBuffer[1] = 0x20;

WriteSensor(WriteBuffer,2); //Write two bytes of data

//Set the latency

WriteBuffer[0] = Latent;

WriteBuffer[1] = 0x20;

WriteSensor(WriteBuffer,2); //Write two bytes of data

//Setup window

WriteBuffer[0] = Window;

WriteBuffer[1] = 0x80;

WriteSensor(WriteBuffer,2); //Write two bytes of data

//Start interupts

accData\_Start();

tapInterrupt\_Start();

accData\_ClearPending();

//Loads the array with x,y for the stars

for(i=0;i<300;i++)

{

stars[i]=rand()%132;

}

/\* Place your initialization/startup code here (e.g. MyInst\_Start()) \*/

//Draws the stars

drawStars(stars);

CyGlobalIntEnable; /\* Enable global interrupts. \*/

while(1)

{

}

}

/\* [] END OF FILE \*/

INTR Includes

#include "project.h"

#include "stdlib.h"

#include "stdbool.h"

#include "stdio.h"

extern int16 stars[300];

extern int16 twinkleStars;

extern int16 i;

extern int16 x;

extern int16 y;

extern int16 xOld;

extern int16 yOld;

extern uint32 counter;

extern uint8 ReadBuffer[10];

extern uint8 WriteBuffer[10];

//For accel setup

extern uint8 PWR\_CTL;

extern uint8 DATA\_FORMAT;

extern uint8 TAP\_AXES;

extern uint8 INT\_ENABLE;

extern uint8 INT\_MAP;

extern uint8 INT\_SOURCE;

extern uint8 DATAX0;

extern uint8 DATAX1;

extern uint8 DATAY0;

extern uint8 DATAY1;

extern uint8 DATAZ0;

extern uint8 DATAZ1;

//For accel tap

extern uint8 THRESH\_TAP;

extern uint8 DUR;

extern uint8 Latent;

extern uint8 Window;

extern void WriteSensor(uint8 \*WriteBuffer, int numvals);

extern void ReadSensor(uint8 \*ReadBuffer, int numvals) ;

extern void drawScope(int16 x,int16 y,int32 color);

extern void drawStars(int16 starArray[]);

//For testing

extern char stringX[10];

extern char stringY[10];

extern char stringZ[10];

**accData int**

//Read acc data

WriteBuffer[0] = 0x32;

WriteSensor(WriteBuffer,1); // Write register number

ReadSensor(ReadBuffer,6); // Reads DataX0 through Data Z1

//Get the data for x,y,z by shifting and or the readbuffers

int16 xData= -((ReadBuffer[1]<<8 )| ReadBuffer[0]);

int16 yData= ((ReadBuffer[3]<<8) | ReadBuffer[2]);

int16 zData= ((ReadBuffer[5]<<8 )| ReadBuffer[4]);

//format the data for testing

//CyGlobalIntDisable;

// sprintf(stringX, "X: %5d",xData);

// GLCD\_PrintString(stringX,30,30,GLCD\_BLUE,GLCD\_BLACK);

// sprintf(stringY, "Y: %5d",yData);

// GLCD\_PrintString(stringY,50,30,GLCD\_BLUE,GLCD\_BLACK);

// sprintf(stringZ, "Z: %5d",zData);

//GLCD\_PrintString(stringZ,70,30,GLCD\_BLUE,GLCD\_BLACK);

// CyGlobalIntEnable;

if((xData>2) &&xData !=xOld ||yData!=yOld) //Move up

{

drawScope(x,y,GLCD\_BLACK);//Draw over old scope

drawStars(stars); //draw stars back

if(x<10) //keep from going off screen

{

x=10;

}

else

{

x=x-(xData/2); // moves in proportion to the tilt

}

//These two if statments help with going at an angle

// Move on the y axis

if(yData>0&&yData!=yOld)

{

if(y>120) // keeps from going off screen

{

y=120;

}

else

{

y=y+(yData/2); //moves in proportion to the tilt

}

}

else if(yData<0&&yData!=yOld)

{

if(y<10) //keeps from going off screen

{

y=10;

}

else

{

y=y+(yData/2); //move in proportion to the tilt

}

}

drawScope(x,y,GLCD\_YELLOW); //draw the scope

}

if((xData<2) &&xData !=xOld||yData!=yOld) //Move down

{

drawScope(x,y,GLCD\_BLACK);

drawStars(stars);

if(x>120) //Keeps from going off screen

{

x=120;

}

else

{

x=x-(xData/2); // Move in proportion to the tilt

}

//These two if statments help with going at an angle

// Move on the y axis

if(yData>0&&yData!=yOld)

{

if(y>120) //Keep from going off screen

{

y=120;

}

else

{

y=y+(yData/2);// Move in proportion to the tilt

}

}

else if(yData<0&&yData!=yOld)

{

if(y<10)// Keep from going off screen

{

y=10;

}

else

{

y=y+(yData/2);// Move in proportion to the tilt

}

}

drawScope(x,y,GLCD\_YELLOW);// Draw scope

}

//Store last value to be used to check if it has moved

yOld=yData;

xOld=xData;

//

// drawScope(x,y,GLCD\_YELLOW);

CyGlobalIntEnable;

Timer\_ClearInterrupt(Timer\_INTR\_MASK\_TC);

**tapInterrupt**

//Send the code to get the tap

WriteBuffer[0]=0x30;

WriteSensor(WriteBuffer,1);

ReadSensor(ReadBuffer,1);

//shift the readbuffer into int

int tap=ReadBuffer[0]>>5;

tap=tap&0b011; // and with the binary to make into int

//For normal blast

if(tap==2)

drawBlast(x,y,0,GLCD\_GREEN,stars);

//Hyper blast

if(tap==3)

drawBlast(x,y,10,GLCD\_BLUE,stars);