

main.c

//Evan Garcia

//Professor Bolding

//EE 3280

//May 23, 2017

//main.c

// This file configures the accelerometer, and sets up the I2C and Interrupts

#include <project.h>

#include <stdio.h>

#include <stdlib.h>

//Write Register Values

const uint32 SLAVEADDRESS = 0x53;

const uint32 PWR\_CTL = 0x2D;

const uint32 DATA\_FORMAT = 0x31;

const uint32 DATAX0 = 0x32;

const uint32 INT\_SOURCE = 0x30;

const uint32 INT\_MAP = 0x2F;

const uint32 INT\_ENABLE = 0x2E;

const uint32 TAP\_AXES = 0x2A;

const uint32 WINDOW = 0x23;

const uint32 LATENT = 0x22;

const uint32 DUR = 0x21;

const uint32 THRESH\_TAP = 0x1D;

//Iterator

int16 i;

//Read and Write Buffer

uint8 ReadBuffer[10];

uint8 WriteBuffer[10];

//Coordinates Buffer for Star Location

int16 starCoordinateX[150]; //Keep track of star x location

int16 starCoordinateY[150]; //Keep track of star y location

//Data Variables

int16 DATAX0Val = 0;

int16 DATAX1Val = 0;

int16 DATAY0Val = 0;

int16 DATAY1Val = 0;

//Read Variables

int16 XReadval = 0;

int16 YReadval = 0;

//X and Y coordinates for center of circle

int16 x = 62;

int16 y = 62;

//Function Prototypes

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

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

void createStarfield(int16, int16[], int16[]);

void createCrosshair(int16, int16);

void deleteCrosshair(int16, int16);

void normalBlasterExploson(int16, int16, int16);

void largeBlasterExploson(int16, int16, int16);

int main()

{

int16 randNumX; //X coordinates for stars

int16 randNumY; //Y coordinates for stars

//Start GLCD & Backlight

GLCD\_Start();

Backlight\_Write(1);

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

//Start Timer and I2C

Timer\_Start();

I2C\_Master\_Start();

//Turn on and Configure Accelerometer

WriteBuffer[0] = PWR\_CTL;

WriteBuffer[1] = 0b00001000;

WriteSensor(WriteBuffer, 2);

WriteBuffer[0] = DATA\_FORMAT;

WriteBuffer[1] = 0b00000000;

WriteSensor(WriteBuffer, 2);

WriteBuffer[0] = INT\_MAP;

WriteBuffer[1] = 0b00000000;

WriteSensor(WriteBuffer, 2);

WriteBuffer[0] = TAP\_AXES;

WriteBuffer[1] = 0b00000111;

WriteSensor(WriteBuffer, 2);

WriteBuffer[0] = THRESH\_TAP;

WriteBuffer[1] = 0x40;

WriteSensor(WriteBuffer, 2);

WriteBuffer[0] = DUR;

WriteBuffer[1] = 0x12;

WriteSensor(WriteBuffer, 2);

WriteBuffer[0] = WINDOW;

WriteBuffer[1] = 0xA0;

WriteSensor(WriteBuffer, 2);

WriteBuffer[0] = LATENT;

WriteBuffer[1] = 0x15;

WriteSensor(WriteBuffer, 2);

WriteBuffer[0] = INT\_ENABLE;

WriteBuffer[1] = 0b01100000;

WriteSensor(WriteBuffer, 2);

//Start Interrupts

TimerInterrupt\_Start();

TapInterrupt\_Start();

//Find Coordinates for starfield

for(i = 0; i < 150; i++)

{

randNumX = rand() % 121;

randNumY = rand() % 121;

starCoordinateX[i] = randNumX;

starCoordinateY[i] = randNumY;

}

//Clear GLCD

GLCD\_Clear(GLCD\_BLACK);

//Draw Starfield

createStarfield(i, starCoordinateX, starCoordinateY);

//Draw Crosshair

createCrosshair(x,y);

for(;;)

{

}

}

//Write to Sensor

void WriteSensor(uint8 \*WriteBuffer, int numvals)

{

I2C\_Master\_I2CMasterClearStatus();

I2C\_Master\_I2CMasterWriteBuf(SLAVEADDRESS,WriteBuffer,numvals,I2C\_Master\_I2C\_MODE\_COMPLETE\_XFER);

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

}

//Read from Sensor

void ReadSensor(uint8 \*ReadBuffer, int numvals)

{

I2C\_Master\_I2CMasterClearStatus();

I2C\_Master\_I2CMasterReadBuf(SLAVEADDRESS,ReadBuffer,numvals,I2C\_Master\_I2C\_MODE\_COMPLETE\_XFER);

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

}

//Draw Starfield

void createStarfield(int16 i, int16 starCoordinateX[], int16 starCoordinateY[])

{

for(i = 0; i < 150; i++)

{

GLCD\_Pixel(starCoordinateX[i], starCoordinateY[i], GLCD\_WHITE);

}

}

//Draw Crosshair

void createCrosshair(int16 x, int16 y)

{

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

GLCD\_DrawLine(x, y-12, x, y + 12, GLCD\_YELLOW);

GLCD\_DrawLine(x-12, y, x + 12, y, GLCD\_YELLOW);

}

//Delete old crosshairs

void deleteCrosshair(int16 x, int16 y)

{

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

GLCD\_DrawLine(x, y-12, x, y + 12, GLCD\_BLACK);

GLCD\_DrawLine(x-12, y, x + 12, y, GLCD\_BLACK);

}

//Normal Size explosion

void normalBlasterExploson(int16 i ,int16 x, int16 y)

{

for(i = 0; i < 25 ; i++)

{

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

}

for(i = 0; i < 26 ; i++)

{

GLCD\_DrawCircle(x, y, 25 - i, GLCD\_BLACK);

}

}

//Hyper Charged explosion

void largeBlasterExploson(int16 i ,int16 x, int16 y)

{

for(i = 0; i < 50 ; i++)

{

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

}

for(i = 0; i < 51 ; i++)

{

GLCD\_DrawCircle(x, y, 50 - i, GLCD\_BLACK);

}

}

/\* [] END OF FILE \*/

TimerInterrupt.c

//Evan Garcia

//Professor Bolding

//EE 3280

//May 23, 2017

//TimerInterrupt.c

//This file reads X and Y values from the accelerometer, and moves the crosshair accordingly

#include <cydevice\_trm.h>

#include <CyLib.h>

#include <TimerInterrupt.h>

#include "cyapicallbacks.h"

#if !defined(TimerInterrupt\_\_REMOVED) /\* Check for removal by optimization \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Place your includes, defines and code here

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* `#START TimerInterrupt\_intc` \*/

#include <project.h>

#include <stdio.h>

//Globals

extern int16 i;

extern uint8 ReadBuffer[10];

extern uint8 WriteBuffer[10];

extern int16 starCoordinateX[150]; //Keep track of star x location

extern int16 starCoordinateY[150]; //Keep track of star y location

extern const uint32 DATAX0;

extern const uint32 SLAVEADDRESS;

extern int16 DATAX0Val;

extern int16 DATAX1Val;

extern int16 DATAY0Val;

extern int16 DATAY1Val;

extern int16 XReadval;

extern int16 YReadval;

extern int16 x; //X and Y coordinates for center of circle

extern int16 y;

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

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

extern void createCrosshair(int16, int16);

extern void deleteCrosshair(int16, int16);

extern void createStarfield(int16, int16[], int16[]);

/\* `#END` \*/

extern cyisraddress CyRamVectors[CYINT\_IRQ\_BASE + CY\_NUM\_INTERRUPTS];

/\* Declared in startup, used to set unused interrupts to. \*/

CY\_ISR\_PROTO(IntDefaultHandler);

CY\_ISR(TimerInterrupt\_Interrupt)

{

#ifdef TimerInterrupt\_INTERRUPT\_INTERRUPT\_CALLBACK

TimerInterrupt\_Interrupt\_InterruptCallback();

#endif /\* TimerInterrupt\_INTERRUPT\_INTERRUPT\_CALLBACK \*/

/\* Place your Interrupt code here. \*/

/\* `#START TimerInterrupt\_Interrupt` \*/

TapInterrupt\_Disable();

//Read X and Y values

WriteBuffer[0] = DATAX0;

WriteSensor(WriteBuffer, 1);

ReadSensor(ReadBuffer, 6);

DATAX0Val = ReadBuffer[0];

DATAX1Val = ReadBuffer[1];

DATAY0Val = ReadBuffer[2];

DATAY1Val = ReadBuffer[3];

//Shift and OR to convert to decimal

XReadval = ((DATAX1Val << 8) | DATAX0Val);

YReadval = ((DATAY1Val <<8) | DATAY0Val);

//Left Slow

if((XReadval < 0 && XReadval > -50) && y > 10)

{

deleteCrosshair(x,y);

y = y - 1;

}

//Right Slow

else if((XReadval > 40 && XReadval < 90)&& y < 120 )

{

deleteCrosshair(x,y);

y = y + 1;

}

//Left Medium

if((XReadval < -50 && XReadval > -150) && y > 10)

{

deleteCrosshair(x,y);

y = y - 3;

}

//Right Medium

else if((XReadval > 90 && XReadval < 190)&& y < 120 )

{

deleteCrosshair(x,y);

y = y + 3;

}

//Left Fast

if((XReadval < -150) && y > 10)

{

deleteCrosshair(x,y);

y = y - 5;

}

//Right Fast

else if((XReadval > 190)&& y < 120 )

{

deleteCrosshair(x,y);

y = y + 5;

}

//Up Slow

if((YReadval < 0 && YReadval > -50)&& x < 121)

{

deleteCrosshair(x,y);

x = x + 1;

}

//Down Slow

else if((YReadval > 40 && YReadval < 90)&& x > 10)

{

deleteCrosshair(x,y);

x = x - 1;

}

//Up Medium

if((YReadval < -50 && YReadval > -150 )&& x < 121)

{

deleteCrosshair(x,y);

x = x + 3;

}

//Down Medium

else if((YReadval > 90 && YReadval < 190 )&& x > 10)

{

deleteCrosshair(x,y);

x = x - 3;

}

//Up Fast

if(YReadval < -150 && x < 121)

{

deleteCrosshair(x,y);

x = x + 5;

}

//Down Fast

else if(YReadval > 190 && x > 10)

{

deleteCrosshair(x,y);

x = x - 5;

}

//Rewrite Crosshair and Starfield

createCrosshair(x,y);

createStarfield(i, starCoordinateX, starCoordinateY);

TapInterrupt\_Enable();

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

/\* `#END` \*/

}

TapInterrupt.c

//Evan Garcia

//Professor Bolding

//EE 3280

//May 23, 2017

//TapInterrupt.c

//This file reads from INT\_SOURCE, and shoots the blaster accordingly

#include <cydevice\_trm.h>

#include <CyLib.h>

#include <TapInterrupt.h>

#include "cyapicallbacks.h"

#if !defined(TapInterrupt\_\_REMOVED) /\* Check for removal by optimization \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Place your includes, defines and code here

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* `#START TapInterrupt\_intc` \*/

//Globals

#include <project.h>

#include <stdio.h>

extern uint8 ReadBuffer[10];

extern uint8 WriteBuffer[10];

extern int16 i;

extern int16 x;

extern int16 y;

extern const uint32 INT\_SOURCE;

char mystringOne[20];

char mystringTwo[20];

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

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

extern void normalBlasterExploson(int16, int16, int16);

extern void largeBlasterExploson(int16, int16, int16);

/\* `#END` \*/

extern cyisraddress CyRamVectors[CYINT\_IRQ\_BASE + CY\_NUM\_INTERRUPTS];

/\* Declared in startup, used to set unused interrupts to. \*/

CY\_ISR\_PROTO(IntDefaultHandler);

CY\_ISR(TapInterrupt\_Interrupt)

{

#ifdef TapInterrupt\_INTERRUPT\_INTERRUPT\_CALLBACK

TapInterrupt\_Interrupt\_InterruptCallback();

#endif /\* TapInterrupt\_INTERRUPT\_INTERRUPT\_CALLBACK \*/

/\* Place your Interrupt code here. \*/

/\* `#START TapInterrupt\_Interrupt` \*/

TimerInterrupt\_Disable();

//Read from INT\_SOURCE

WriteBuffer[0] = INT\_SOURCE;

WriteSensor(WriteBuffer, 1);

ReadSensor(ReadBuffer, 1);

//If Single Tap do normal explosion, if double, do hyper explosion

if((ReadBuffer[0] > 0xC0 && ReadBuffer[0] < 0xC5) || ReadBuffer[0] == 0x42 )

{

normalBlasterExploson(i,x,y);

}

else if((ReadBuffer[0] > 0xE0 && ReadBuffer[0] < 0xE5) || ReadBuffer[0] == 0x62 )

{

largeBlasterExploson(i,x,y);

}

TimerInterrupt\_Enable();

/\* `#END` \*/

}