;;/\*----------------------------------------------------------------------------

Code for Lab 6

In this project the PIT and the TPM are used to create a tone.

\*---------------------------------------------------------------------------\*/

#include "cmsis\_os2.h"

#include <MKL25Z4.H>

#include "../include/gpio.h"

#include "../include/pit.h"

#include "../include/tpmPwm.h"

#include "..\include\adc\_defs.h"

osEventFlagsId\_t evtFlags ; // event flags

osEventFlagsId\_t evtFlags2 ; // second event flags

bool pressed = true;

volatile float measured\_voltage;

/\*--------------------------------------------------------------

\* Tone task - switch tone on and off

\*--------------------------------------------------------------\*/

// Audio tone states

osThreadId\_t t\_tone; //Thread to control tone

void toneTask (void \*arg) {

const uint32\_t array[] = {20040,18915,17853,16851,15905,15013,14170,13375,12624,11916,11247,10616};

bool alarm\_state = 0;

int i=0;

while (1) {

osEventFlagsWait (evtFlags, MASK(PRESS\_EVT), osFlagsWaitAny, osWaitForever);

alarm\_state = !alarm\_state;

if (alarm\_state == 1)

{

blueLEDOnOff(LED\_ON);

infraLEDOnOff(LED\_ON);

initial\_sound\_on();

pressed = 1;

while(pressed)

{

MeasureVoltage() ;

measured\_voltage = ((VREF \* sres) / ADCRANGE);

voltage\_range();

if(isPressed())

{

pressed = false;

}

}

}

else if (alarm\_state == 0)

{

blueLEDOnOff(LED\_OFF);

infraLEDOnOff(LED\_OFF);

initial\_sound\_off();

}

}

//osEventFlagsWait (evtFlags, MASK(PRESS\_EVT), osFlagsWaitAny, osWaitForever);

}

/\*osThreadId\_t t\_volume; //Thread to control volume

void volumeTask(void \*arg){

const uint32\_t array\_volume[] = {0,1,2,4,8,16,32,64,128};

int j=0;

while (1) {

osEventFlagsWait (evtFlags2, MASK(PRESS\_EVT2), osFlagsWaitAny, osWaitForever);

setPWMDuty(array\_volume[j]);

if (j == 8)

{

j=-1;

}

j++;

}

}\*/

/\*------------------------------------------------------------

\* Button task - poll button and send signal when pressed

\*------------------------------------------------------------\*/

osThreadId\_t t\_button; /\* task id of task to read the first button \*/

void buttonTask (void \*arg) {

int bState = UP ;

int bCounter = 0 ;

while (1) {

osDelay(10) ;

if (bCounter) bCounter-- ;

switch (bState) {

case UP:

if (isPressed()) {

osEventFlagsSet(evtFlags, MASK(PRESS\_EVT));

bState = DOWN ;

}

break ;

case DOWN:

if (!isPressed()) {

bCounter = BOUNCEP ;

bState = BOUNCE ;

}

break ;

case BOUNCE:

if (isPressed()) {

bCounter = BOUNCEP ;

bState = DOWN ;

} else {

if (!bCounter) {

bState = UP ;

}

}

break ;

}

}

}

/\*osThreadId\_t t\_button2; task id of task to read the second button

void buttonTask2 (void \*arg) {

int bState = UP ;

int bCounter = 0 ;

while (1) {

osDelay(10) ;

if (bCounter) bCounter-- ;

switch (bState) {

case UP:

if (isPressed2()) {

osEventFlagsSet(evtFlags2, MASK(PRESS\_EVT2));

bState = DOWN ;

}

break ;

case DOWN:

if (!isPressed2()) {

bCounter = BOUNCEP ;

bState = BOUNCE ;

}

break ;

case BOUNCE:

if (isPressed2()) {

bCounter = BOUNCEP ;

bState = DOWN ;

} else {

if (!bCounter) {

bState = UP ;

}

}

break ;

}

}

}\*/

void initial\_sound\_on()

{

startTimer(0);

int i = 0;

int array[] = {20040,18915,17853};

for(i=0;i<=2;i++){

setTimer(0,array[i]);

osDelay(500);

}

stopTimer(0);

}

void initial\_sound\_off()

{

startTimer(0);

int i = 0;

int array[] = {10616,11247,11916};

for(i=0;i<=2;i++){

setTimer(0,array[i]);

osDelay(500);

}

stopTimer(0);

}

void voltage\_range ()

{

if(measured\_voltage <= 3.29 && measured\_voltage >= 3)

{

playtone(0);

}

else if(measured\_voltage < 3 && measured\_voltage >= 2.5)

{

playtone(1);

}

else if(measured\_voltage < 2.5 && measured\_voltage >= 2)

{

playtone(2);

}

else if(measured\_voltage < 2 && measured\_voltage >= 1.5)

{

playtone(3);

}

else if(measured\_voltage < 1.5 && measured\_voltage >= 1)

{

playtone(4);

}

else if(measured\_voltage < 1 && measured\_voltage >= 0.5)

{

playtone(5);

}

else if(measured\_voltage < 0.5 && measured\_voltage >= 0.25)

{

playtone(6);

}

}

void playtone(int i)

{

int array\_vol[] = {0,4,8,16,32,64,128};

setPWMDuty(array\_vol[i]);

}

/\*----------------------------------------------------------------------------

\* Main: Initialize and start RTX Kernel

\*---------------------------------------------------------------------------\*/

volatile uint8\_t calibrationFailed ; // zero expected

int main (void) {

SIM->SCGC5 |= SIM\_SCGC5\_PORTB\_MASK | SIM\_SCGC5\_PORTD\_MASK | SIM\_SCGC5\_PORTE\_MASK ;

// init\_LED() ; // initialise LED

//init\_ButtonGPIO() ; // initialise GPIO input

//init\_ButtonState() ; // initialise button state variables

Init\_ADC() ; // Initialise ADC

calibrationFailed = ADC\_Cal(ADC0) ; // calibrate the ADC

while (calibrationFailed) ; // block progress if calibration failed

Init\_ADC() ; // Reinitialise ADC

//Init\_MeasureState() ; // Initialise measure state

//Init\_SysTick(1000) ; // initialse SysTick every 1ms

configureGPIOinput() ; // Initialise button

configureGPIOoutput() ; // Initialise output

configurePIT(0) ; // Configure PIT channel 0

setTimer(0, 20040) ; // Frequency for MIDI 60 - middle C

configureTPM0forPWM() ;

setPWMDuty(32) ; // 50% volume

// Max is 128; off is 0

infraLEDOnOff(LED\_OFF);

SystemCoreClockUpdate() ;

// Initialize CMSIS-RTOS

osKernelInitialize();

// Create event flags

evtFlags = osEventFlagsNew(NULL);

evtFlags2 = osEventFlagsNew(NULL);

// Create threads

t\_tone = osThreadNew(toneTask, NULL, NULL) ;

//t\_volume = osThreadNew(volumeTask, NULL, NULL);

t\_button = osThreadNew(buttonTask, NULL, NULL) ;

//t\_button2 = osThreadNew(buttonTask2, NULL, NULL) ;

osKernelStart(); // Start thread execution - DOES NOT RETURN

for (;;) {} // Only executed when an error occurs

}