```
================================================================================
=======
<Blinkinlights.c>
================================================================================
=======
/*
 * Copyright 2016-2020 NXP
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without modification,
 * are permitted provided that the following conditions are met:
 *
 * o Redistributions of source code must retain the above copyright notice, this
list
 *   of conditions and the following disclaimer.
 *
 * o Redistributions in binary form must reproduce the above copyright notice, this
 *   list of conditions and the following disclaimer in the documentation and/or
 *   other materials provided with the distribution.
 *
 * o Neither the name of NXP Semiconductor, Inc. nor the names of its
 *   contributors may be used to endorse or promote products derived from this
 *   software without specific prior written permission.
 *
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
 * WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
 * DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
 * ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
 * (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
 * ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
 * (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
 * SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
 */

/**
 * @file    Blinkenlights.c
 * @brief   Blinks Light in a with a start run sequence and then continues in an
infinitee sequence with set delays
 *
 * @author Arpit Savarkar
 * @date September 10 2020
 * @version 1.0
 *
 *
 * Sources of Reference :
 * Online Links :
https://github.com/alexander-g-dean/ESF/tree/master/NXP/Misc/Touch%20Sense
 * Online Links :
https://github.com/alexander-g-dean/ESF/blob/master/NXP/Code/Chapter_2/Source/
main.c
 * Textbooks : Embedded Systems Fundamentals with Arm Cortex-M based
MicroControllers
 * I would like to thank the SA's of the course Rakesh Kumar, Saket Penurkar and
Howdy Pierece for their
 * support for helping to understand the IDE functioning
 */
#include <stdio.h>
```

```
#include "board.h"
#include "peripherals.h"
#include "pin_mux.h"
#include "clock_config.h"
#include "MKL25Z4.h"
#include "fsl_debug_console.h"
#include "led.h"
#include "delay.h"
#include "cap_sensor.h"
#include <stdint.h>
#include <stdbool.h>
#include <time.h>

/* ************************************************************************
 * Definitions
 ************************************************************************/

//#define sec  6000000
#define SEC  5000

static enum {ST_ON, ST_OFF, ST_TSI, ST_DEAD} next_state = ST_ON;
// ----------------------------------------------------------------------

/* ************************************************************************
 * Prototypes
 ************************************************************************/

/*
 * @brief   Application entry point.
 */
int main(void) {

      /* Init board hardware. */
    BOARD_InitBootPins();
    BOARD_InitBootClocks();
    BOARD_InitBootPeripherals();
#ifndef BOARD_INIT_DEBUG_CONSOLE_PERIPHERAL
    /* Init FSL debug console. */
    BOARD_InitDebugConsole();
#endif

#ifdef FLAG
    PRINTF("\nThe custom build via IDE\n");
#endif

#ifdef DEBUG
    PRINTF("\n\r Debug Mode");
#endif

    /* Initialize the LED Pins as OUTPUT Pins */
    LED_Init();
    CAP_Init();
#ifdef DEBUG
    PRINTF("\n\r Initialization Routine Complete");
#endif


    // Initially Clear all the LED Bits on Ports
    LED_Control(0, 0, 0);
```

```
#ifdef DEBUG
    PRINTF("\n\r All LED's OFF ");
#endif

    int val = 0;
    bool LED_Status[3] = {RED_ON, GREEN_ON, BLUE_ON};


    // Startup Pattern
    LED_Startup(SEC);
#ifdef DEBUG
    PRINTF("\n\r StartUp Routine Complete");
#endif

#ifdef DEBUG
    PRINTF("\n\r Begin Infinite Loop");
#endif

    for(int i=0 ; ; i++) {
      if(i == 4) {
            i = 0;
      }
#ifdef DEBUG
    PRINTF("\n\r START TIMER : %u", delayTimes[i]);
#endif
            if( next_state == ST_ON) {
                  LED_Control(LED_Status[0], LED_Status[1] , LED_Status[2]);

                  conditional_Delay(delayLoopUp[i], delay100MSECLoopUp[i], &val,
LED_Status);
                  next_state = ST_OFF;
            }
            if (next_state == ST_OFF) {
                  LED_Control(0,0,0);
                  Delay(SEC/2);
                  next_state = ST_ON;
            }
    }

    // End of Main
    return 0 ;
}


================================================================================
=======
<cap_sensor.h>
================================================================================
=======
/*
 * cap_sensor.h
 *
 *  Created on: Sep 19, 2020
 *      Author: Arpit Savarkar / arpit.savarkar@colorado.edu
 */

#ifndef CAP_SENSOR_H_
#define CAP_SENSOR_H_
```

```
#include "MKL25Z4.h"
#include <stdint.h>
#include <stdio.h>
#include "fsl_debug_console.h"

/* ************************************************************************
 * Macros
 ************************************************************************/
#define SCAN_OFFSET 588  // Offset for scan range
#define SCAN_DATA TSI0->DATA & 0xFFFF // Accessing the bits held in
TSI0_DATA_TSICNT
#define NOISE_LOW 25
#define NOISE_HIGH 60000

/* ************************************************************************
 * Prototypes
 ************************************************************************/

/*   @brief  Initialization of Capacitive Sensor (Clocks Gating etc) */
void CAP_Init(void);


/**
 *   @brief  Helper Function to return back the Value sensed by the capacitive
sensor
 *
 *
 *   @param  void
 *
 *   @return  Scanned Value with OFFSET Subtracted
 */
uint16_t CAP_Scan(void);



//void helper_CAP_Scan(volatile uint16_t val);

#endif /* CAP_SENSOR_H_ */
```

```
================================================================================
=======
<cap_sensor.c>
================================================================================
=======
/*
 * cap_sensor.c
 *
 *  Created on: Sep 19, 2020
 *      Author: Arpit Savarkar / arpit.savarkar@colorado.edu
 *
 *  References:
 *
https://www.digikey.com/eewiki/display/microcontroller/Using+the+Capacitive+Touch+S
ensor+on+the+FRDM-KL46Z
 */

#include "MKL25Z4.h"
#include "cap_sensor.h"
```

```
/* ************************************************************************
 * Code
 ************************************************************************/

/*   @brief  Initialization of Capacitive Sensor (Clocks Gating etc) */
void CAP_Init(void) {

      // Enable clock for TSI PortB 16 and 17
            SIM->SCGC5 |= SIM_SCGC5_TSI_MASK;

            TSI0->GENCS = TSI_GENCS_OUTRGF_MASK |  // Out of range flag, set to 1
to clear
                  TSI_GENCS_MODE(0U) |  // Set at 0 for capacitive sensing.  Other
settings are 4 and 8 for threshold detection, and 12 for noise detection
                  TSI_GENCS_REFCHRG(0U) | // 0-7 for Reference charge
                  TSI_GENCS_DVOLT(0U) | // 0-3 sets the Voltage range
                  TSI_GENCS_EXTCHRG(0U) | //0-7 for External charge
                  TSI_GENCS_PS(0U) | // 0-7 for electrode prescaler
                  TSI_GENCS_NSCN(31U) | // 0-31 + 1 for number of scans per
electrode
                  TSI_GENCS_TSIEN_MASK | // TSI enable bit
                  TSI_GENCS_STPE_MASK | // Enables TSI in low power mode
                  TSI_GENCS_EOSF_MASK ; // End of scan flag, set to 1 to clear
#ifdef DEBUG
      PRINTF("\n\r Clock Gating and Initialization of Capacitive Sensor Complete
");
#endif
}


/**
 *   @brief  Helper Function to return back the Value sensed by the capacitive
sensor
 *
 *
 *   @param  void
 *
 *   @return  Scanned Value with OFFSET Subtracted
 */
uint16_t CAP_Scan(void) {
      uint16_t scan;
//#ifdef DEBUG
//    PRINTF("\n\r Capacitive Touch Scanning with OFFSET %d: ", SCAN_OFFSET);
//#endif
            TSI0->DATA = TSI_DATA_TSICH(9U); // Using channel 9 of the TSI
            TSI0->DATA |= TSI_DATA_SWTS_MASK; // Software trigger for scan
            scan = SCAN_DATA;
            TSI0->GENCS |= TSI_GENCS_EOSF_MASK ; // Reset end of scan flag
            return scan - SCAN_OFFSET;
}
================================================================================
=======
<delay.h>
================================================================================
=======
/*
 * delay.h
 *
```

```
 *  Created on: Sep 19, 2020
 *      Author: Arpit Savarkar (arpit.savarkar@colorado.edu)
 */

#ifndef DELAY_H_
#define DELAY_H_

#include <stdint.h>
#include <stdbool.h>
#include "led.h"
#include "cap_sensor.h"


/* ************************************************************************
 * Static Globals
 ************************************************************************/
static const int delayTimes[4] =
{
            // 500, 1000, 2000, 3000 mSec
            500, 1000, 2000, 3000
};

static const int delayLoopUp[4] =
{
            // 500, 1000, 2000, 3000 mSec
            312500, 625000, 900000, 1200000
};

static const int delay100MSECLoopUp[4] =
{
            // 500, 1000, 2000, 3000 mSec
            62500, 62500, 40000, 333334
};


/* ************************************************************************
 * Prototypes
 ************************************************************************/


extern void Delay(volatile uint32_t delay_time);
void conditional_Delay(volatile uint32_t delay_time, int MSEC100 , int* val, bool*
LED_array);

#endif /* DELAY_H_ */

================================================================================
=======
<delay.c>
================================================================================
=======
/*
 * delay.c
 *
 *  Created on: Sep 19, 2020
 *      Author: Arpit Savarkar / arpit.savarkar@colorado.edu
 */
#include "delay.h"
```

```
/**
 *   @brief  Busy Waiting Function upto specified Delay Limit
 *
 *   @param  delay_time: Delay Time in in Seconds
 *
 *   @return  void
 */
void Delay(volatile unsigned int delay_time) {
      delay_time = delay_time*1000;
      while(delay_time--) {
            ;
      }
}


/**
 *   @brief  Busy Waiting Function , also checks if the capacitive touch sensor
provides a
 *           Non Noisy value read every 100 msec
 *
 *   @param  delay_time: Delay Time in in milliseconds
 *   @param  MSEC100: 100 milliseconds delay absolute value
 *   @param  val : Value from the capacitive sensor
 *   @param  LED_array : Pointer to an LED_Status Register
 *
 *   @return  void
 */
void conditional_Delay(volatile uint32_t delay_time, int MSEC100 , int* val, bool*
LED_array) {

      while(delay_time-- != 0) {
            if((delay_time % MSEC100) == 0) {
                  *val = CAP_Scan();
                  if( (*val > NOISE_LOW) && (*val < NOISE_HIGH) ) {
                        helper_COLOR(val, LED_array);
                  }
            }
            else {
                  ;
            }
      }
}
================================================================================
=======
<led.h>
================================================================================
=======
/*
 * led.h
 *
 *  Created on: Sep 19, 2020
 *      Author: Arpit Savarkar / arpit.savarkar@colorado.edu
 */

#ifndef LED_H_
#define LED_H_

#include "delay.h"
```

```
#include "MKL25Z4.h"
#include <stdbool.h>
#include <stdint.h>
#include <stdio.h>
#include "fsl_debug_console.h"

/* ************************************************************************
 * Definitions
 ************************************************************************/

#define RED_LED_PIN_POS (18)
#define GREEN_LED_PIN_POS (19)
#define BLUE_LED_PIN_POS (1)

#define MASK(x) (1UL << (x))

#define RED_ON 1
#define RED_OFF 0
#define BLUE_ON 1
#define BLUE_OFF 0
#define GREEN_ON 1
#define GREEN_OFF 0
/* ************************************************************************
 * Prototypes
 ************************************************************************/

/**
 *   @brief  Initializes the GPIO Ports and Clock Gating and Direction of Pins
 *
 *   @param  void
 *
 *   @return  void
 */
void LED_Init(void);

/**
 *   @brief  Clears and Sets the bits required for Digital On/Off of LEDs
 *
 *   PCOR -> Clears Bit for the set Mask Pin (Active Low)
 *   PSOR -> Sets Bit for the set Mask Pin
 *
 *   @param  red: Boolean 1 to set High, 0 to set Low for Red Led
 *   @param  green: Boolean 1 to set High, 0 to set Low for Green Led
 *   @param  blue: Boolean 1 to set High, 0 to set Low for Blue Led
 *
 *   @return  void
 */
void LED_Control(unsigned int red, unsigned int green, unsigned int blue);

/**
 *   @brief  Implement Startup Routine
 *
 *   Implements the following routine
 *    RED for 500 msec, OFF for 100 msec,
 *      GREEN for 500 msec, OFF for 100 msec,
 *      BLUE for 500 msec, OFF for 100 msec
 *      WHITE (that is, RED, GREEN, and BLUE all on) for 100 msec, OFF for 100 msec
 *    WHITE for 100 msec, OFF for 100 MSEC
 *
```

```
 *   @param  sec :  Delay period
 *
 *   @return  void
 */
void LED_Startup(uint32_t sec);


//void LED_Flash(uint32_t sec);
//void LED_TurnOFF(void);

/**
 *   @brief  Helper Function to Set the LED's based on Input of Helper Function
 *
 *   Given a value as an input from the Capacitive Sensor it Sets the LED Status
array
 *   to the desired config
 *
 *   @param  val :  Value with OFFSET from the capacitive Sensor
 *   @param  LED_array : LED Status Array
 *
 *   @return  void
 */
void helper_COLOR(int* val, bool* LED_array);

#endif /* LED_H_ */
```

```
================================================================================
=======
<led.c>
================================================================================
=======
/*
 * led.c
 *
 *  Created on: Sep 19, 2020
 *      Author: Arpit Savarkar (arpit.savarkar@colorado.edu)
 */


#include "led.h"


/**
 *   @brief  Initializes the GPIO Ports and Clock Gating and Direction of Pins
 *
 *   @param  void
 *
 *   @return  void
 */
void LED_Init(void) {

#ifdef DEBUG
      PRINTF("\n\r Enabling Clock for PortB and PortD");
#endif
      // Clock Signal is required to Enable Ports
      SIM->SCGC5 |= SIM_SCGC5_PORTB_MASK | SIM_SCGC5_PORTD_MASK;

      /* Setting the required pins as GPIO Pins */
#ifdef DEBUG
```

```
      PRINTF("\n\r Setting the Pins to be used as GPIO ");
#endif
      // Red
      PORTB->PCR[RED_LED_PIN_POS] &= ~PORT_PCR_MUX_MASK;
      PORTB->PCR[RED_LED_PIN_POS] |= PORT_PCR_MUX(1);

      // Green
      PORTB->PCR[GREEN_LED_PIN_POS] &= ~PORT_PCR_MUX_MASK;
      PORTB->PCR[GREEN_LED_PIN_POS] |= PORT_PCR_MUX(1);

      //Blue
      PORTD->PCR[BLUE_LED_PIN_POS] &= ~PORT_PCR_MUX_MASK;
      PORTD->PCR[BLUE_LED_PIN_POS] |= PORT_PCR_MUX(1);

#ifdef DEBUG
      PRINTF("\n\r Setting the Pin direction to OUTPUT");
#endif
      /* Pin Direction to Set as Output   */
      PTB->PDDR |= MASK(RED_LED_PIN_POS) | MASK(GREEN_LED_PIN_POS);
      PTD->PDDR |= MASK(BLUE_LED_PIN_POS);

#ifdef DEBUG
      PRINTF("\n\r LED Initialization Successful");
#endif
}


/**
 *   @brief  Clears and Sets the bits required for Digital On/Off of LEDs
 *
 *   PCOR -> Clears Bit for the set Mask Pin (Active Low)
 *   PSOR -> Sets Bit for the set Mask Pin
 *
 *   @param  red: Boolean 1 to set High, 0 to set Low for Red Led
 *   @param  green: Boolean 1 to set High, 0 to set Low for Green Led
 *   @param  blue: Boolean 1 to set High, 0 to set Low for Blue Led
 *
 *   @return  void
 */
void LED_Control(unsigned int red, unsigned int green, unsigned int blue) {

#ifdef DEBUG
      PRINTF("\n\r LED CHANGE STATUS LOG ; 1-> Active, 0->Disabled");
      PRINTF("\n\r RED LED : %u", red);
      PRINTF("\n\r GREEN LED : %u", green);
      PRINTF("\n\r BLUE LED : %u", blue);
#endif

      // BLue
      if(blue){
            PTD->PCOR = MASK(BLUE_LED_PIN_POS);
      }
      else {
            PTD->PSOR = MASK(BLUE_LED_PIN_POS);
      }

      // Red
      if(red){
            PTB->PCOR = MASK(RED_LED_PIN_POS);
```

```
	}
	else {
		PTB->PSOR = MASK(RED_LED_PIN_POS);
	}

	// Green
	if(green){
		PTB->PCOR = MASK(GREEN_LED_PIN_POS);
	}
	else {
		PTB->PSOR = MASK(GREEN_LED_PIN_POS);
	}

}

//void LED_TurnOFF(void) {
//    PTD->PDDR &= ~ MASK(BLUE_LED_PIN_POS);
//    PTB->PDDR &= ~ MASK(RED_LED_PIN_POS);
//    PTB->PDDR &= ~ MASK(GREEN_LED_PIN_POS);
//}


/**
 *   @brief  Implement Startup Routine
 *
 *   Implements the following routine
 *    RED for 500 msec, OFF for 100 msec,
 *      GREEN for 500 msec, OFF for 100 msec,
 *      BLUE for 500 msec, OFF for 100 msec
 *      WHITE (that is, RED, GREEN, and BLUE all on) for 100 msec, OFF for 100 msec
 *    WHITE for 100 msec, OFF for 100 MSEC
 *
 *   @param  sec :  Delay period
 *
 *   @return  void
 */
void LED_Startup(uint32_t sec) {
	// Red n for 500 msec

	LED_Control(RED_ON, GREEN_OFF, BLUE_OFF);
#ifdef DEBUG
	PRINTF("\n\r LED RED ON for 500 msec");
#endif
	Delay(sec/2);

	// All off for 100 msec
	LED_Control(RED_OFF, GREEN_OFF, BLUE_OFF);
#ifdef DEBUG
	PRINTF("\n\r ALL LED DISABLED for 100 msec");
#endif
	Delay(sec/10);

	// Green for 500 msec
	LED_Control(RED_OFF, GREEN_ON, BLUE_OFF);
#ifdef DEBUG
	PRINTF("\n\r LED GREEN ON for 500 msec");
#endif
	Delay(sec/2);
```

```
      // All off for 100 msec
      LED_Control(RED_OFF, GREEN_OFF, BLUE_OFF);
#ifdef DEBUG
      PRINTF("\n\r ALL LED DISABLED for 100 msec");
#endif
      Delay(sec/10);

      // Blue for 500 msec
      LED_Control(RED_OFF, GREEN_OFF, BLUE_ON);
#ifdef DEBUG
      PRINTF("\n\r LED BLUE ON for 500 msec");
#endif
      Delay(sec/2);

      // All off for 100 msec
      LED_Control(RED_OFF, GREEN_OFF, BLUE_OFF);
#ifdef DEBUG
      PRINTF("\n\r ALL LED DISABLED for 100 msec");
#endif
      Delay(sec/10);

      // White for 100 msec
      LED_Control(RED_ON, GREEN_ON, BLUE_ON);
#ifdef DEBUG
      PRINTF("\n\r LED WHITE for 100 msec");
#endif
      Delay(sec/10);

      // All off for 100 msec
      LED_Control(RED_OFF, GREEN_OFF, BLUE_OFF);
#ifdef DEBUG
      PRINTF("\n\r ALL LED DISABLED for 100 msec");
#endif
      Delay(sec/10);

      // White for 100 msec
      LED_Control(RED_ON, GREEN_ON, BLUE_ON);
#ifdef DEBUG
      PRINTF("\n\r LED WHITE for 100 msec");
#endif
      Delay(sec/10);

      // All off for 100 msec
      LED_Control(RED_OFF, GREEN_OFF, BLUE_OFF);
#ifdef DEBUG
      PRINTF("\n\r ALL LED DISABLED for 100 msec");
#endif
      Delay(sec/10);

}


/**
 *   @brief  Helper Function to Set the LED's based on Input of Helper Function
 *
 *   Given a value as an input from the Capacitive Sensor it Sets the LED Status
array
 *   to the desired config
 *
```

```
 *   @param  val :  Value with OFFSET from the capacitive Sensor
 *   @param  LED_array : LED Status Array
 *
 *   @return  void
 */
void helper_COLOR(int* val, bool* LED_array) {

#ifdef DEBUG
      PRINTF("\n\r SLIDER VALUE: %u", *val);
#endif

      if(*val<= 20 || *val > 2000) {
            LED_Control(RED_OFF, GREEN_OFF, BLUE_OFF);
            LED_array[0] = LED_array[1] = LED_array[2] = 0;
      }
      else if (*val < 350) {
            LED_Control(RED_ON, 0 , 0);
            LED_array[0] = RED_ON;
            LED_array[1] = LED_array[2] = 0;
      }
      else if(*val < 750) {
            LED_Control(0, GREEN_ON , 0);
            LED_array[1] = GREEN_ON;
            LED_array[0] = LED_array[2] = 0;
      }
      else if(*val >= 750) {
            LED_Control(0, 0, BLUE_ON);
            LED_array[2] = BLUE_ON;
            LED_array[0] = LED_array[1] = 0;
      }
}
```