PES Project 2 Document

Makefile

```
#
Compilers
            ARM_CC = arm-none-eabi-gcc
            ARM_LL = arm-none-eabi-gcc
            PC_CC = gcc

            # Executable file
            EXE := \
                   ./Debug/pes_project_2.axf

            # Compiler Flags
            PC_FLAGS := -Wall -Werror -c -std=c99 -O0 -g3

            ARM_FLAGS := -c -std=gnu99 -O0 \
                   -g3 -fmessage-length=0 -ffunction-sections -fdata-sections \
                   -fno-builtin -fno-common -mcpu=cortex-m0plus -mthumb

            # Linker Flags
            LL_FLAGS := -nostdlib -Xlinker -Map="Debug/pes_project_2.map" \
                   -Xlinker --gc-sections -Xlinker -print-memory-usage \
                   -mcpu=cortex-m0plus -mthumb -T linkerfile.ld -o $(EXE)

            # Include directories
            ARM_INCS := -I"./board" -I"./CMSIS" -I"./drivers" -I"./source" -I"./startup" -
            I"./utilities" -I"./source/fb" -I"./include"

            PC_INCS := -I"./source" -I"./source/pc"

            # ARM defines
            ARM_DEFS := -D__REDLIB__ -DDEBUG -DCPU_MKL25Z128VLK4 \
                   -DFRDM_KL25Z -DFREEDOM -DCPU_MKL25Z128VLK4_cm0plus \
                   -DSDK_DEBUGCONSOLE=0 -DCR_INTEGER_PRINTF -D__MCUXPRESSO \
                   -D__USE_CMSIS

            # ARM Object Files
            ARM_OBJS = \
                   ./Debug/board/board.o \
                   ./Debug/board/clock_config.o \
                   ./Debug/board/peripherals.o \
        ./Debug/board/pin_mux.o \
        ./Debug/CMSIS/system_MKL25Z4.o \
        ./Debug/drivers/fsl_clock.o \
        ./Debug/drivers/fsl_common.o \
        ./Debug/drivers/fsl_flash.o \
        ./Debug/drivers/fsl_gpio.o \
        ./Debug/drivers/fsl_lpsci.o \
        ./Debug/drivers/fsl_rtc.o \
        ./Debug/drivers/fsl_smc.o \
        ./Debug/drivers/fsl_uart.o \
        ./Debug/startup/startup_mkl25z4.o \
        ./Debug/utilities/fsl_debug_console.o \
        ./Debug/source/fb/fb_led.o \
        ./Debug/source/fb/fb_loop.o \
        ./Debug/source/fb/fb_debug.o

# ARM Dependency Files
ARM_DEPS = \
        ./Debug/board/board.d \
        ./Debug/board/clock_config.d \
        ./Debug/board/peripherals.d \
        ./Debug/board/pin_mux.d \
        ./Debug/CMSIS/system_MKL25Z4.d \
        ./Debug/drivers/fsl_clock.d \
        ./Debug/drivers/fsl_common.d \
        ./Debug/drivers/fsl_flash.d \
        ./Debug/drivers/fsl_gpio.d \
        ./Debug/drivers/fsl_lpsci.d \
        ./Debug/drivers/fsl_rtc.d \
        ./Debug/drivers/fsl_smc.d \
        ./Debug/drivers/fsl_uart.d \
        ./Debug/startup/startup_mkl25z4.d \
        ./Debug/utilities/fsl_debug_console.d \
        ./Debug/source/fb/fb_led.d \
        ./Debug/source/fb/fb_loop.d \
        ./Debug/source/fb/fb_debug.d

# PC Object Files
PC_OBJS = \
        ./Debug/source/pc/pc_print.o \
        ./Debug/source/pc/pc_loop.o \
        ./Debug/source/pc/pc_debug.o
```

```
# PC Dependencies Files
PC_DEPS = \
	./Debug/source/pc/pc_print.d \
	./Debug/source/pc/pc_loop.d \
	./Debug/source/pc/pc_debug.d

# Conditional Execution
ifeq ($(BV),FB_RUN)
build_version := fb_run
else ifeq ($(BV),FB_DEBUG)
build_version := fb_debug
else ifeq ($(BV),PC_RUN)
build_version := pc_run
else ifeq ($(BV),PC_DEBUG)
build_version := pc_debug
endif

# Rule for all
all: $(EXE)

# Rule for executable
$(EXE): $(build_version)

# Different targets
pc_run: directories   $(PC_OBJS)
	$(PC_CC) $(PC_FLAGS) $(PC_INCS) -DPC_RUN -o ./Debug/source/main.o
./source/main.c
	$(PC_CC) $(PC_OBJS) ./Debug/source/main.o -o $(EXE)

pc_debug: directories $(PC_OBJS)
	$(PC_CC) $(PC_FLAGS) $(PC_INCS) -DPC_DEBUG -o ./Debug/source/main.o
./source/main.c
	$(PC_CC) $(PC_OBJS) ./Debug/source/main.o -o $(EXE)

fb_run: directories   $(ARM_OBJS) linkerfile.ld
	$(ARM_CC) $(ARM_FLAGS) $(ARM_INCS) -DFB_RUN -o ./Debug/source/main.o
./source/main.c
	$(ARM_LL) $(LL_FLAGS) $(ARM_OBJS) ./Debug/source/main.o -o $(EXE)

fb_debug: directories $(ARM_OBJS) linkerfile.ld
	$(ARM_CC) $(ARM_FLAGS) $(ARM_INCS) -DFB_DEBUG -o ./Debug/source/main.o
./source/main.c
	$(ARM_LL) $(LL_FLAGS) $(ARM_OBJS) ./Debug/source/main.o  -o $(EXE)
```

```
# Target for making directories that are needed
# https://stackoverflow.com/questions/1950926/create-directories-using-make-
file
# Leveraged code
OUT_DIR := Debug Debug/CMSIS Debug/drivers Debug/board Debug/source \
       Debug/source/fb Debug/source/pc Debug/utilities Debug/startup

MK := mkdir -p
directories:
       $(MK) $(OUT_DIR)


# ARM Targets
# Targets for compiling required object files
# https://mcuoneclipse.com/2017/07/22/tutorial-makefile-projects-with-eclipse/
# Leveraged code
./Debug/CMSIS/%.o: ./CMSIS/%.c
       $(ARM_CC) $(ARM_FLAGS) $(ARM_DEFS) $(ARM_INCS) -MMD -MP -
MF"./$(@:%.o=%.d)" -MT"./$(@:%.o=%.o)" -MT"./$(@:%.o=%.d)" -o "$@" "$<"

./Debug/drivers/%.o: ./drivers/%.c
       $(ARM_CC) $(ARM_FLAGS) $(ARM_DEFS) $(ARM_INCS) -MMD -MP -
MF"./$(@:%.o=%.d)" -MT"./$(@:%.o=%.o)" -MT"./$(@:%.o=%.d)" -o "$@" "$<"

./Debug/board/%.o: ./board/%.c
       $(ARM_CC) $(ARM_FLAGS) $(ARM_DEFS) $(ARM_INCS) -MMD -MP -
MF"./$(@:%.o=%.d)" -MT"./$(@:%.o=%.o)" -MT"./$(@:%.o=%.d)" -o "$@" "$<"

./Debug/startup/%.o: ./startup/%.c
       $(ARM_CC) $(ARM_FLAGS) $(ARM_DEFS) $(ARM_INCS) -MMD -MP -
MF"./$(@:%.o=%.d)" -MT"./$(@:%.o=%.o)" -MT"./$(@:%.o=%.d)" -o "$@" "$<"

./Debug/utilities/%.o: ./utilities/%.c
       $(ARM_CC) $(ARM_FLAGS) $(ARM_DEFS) $(ARM_INCS) -MMD -MP -
MF"./$(@:%.o=%.d)" -MT"./$(@:%.o=%.o)" -MT"./$(@:%.o=%.d)" -o "$@" "$<"

./Debug/source/fb/%.o: ./source/fb/%.c
       $(ARM_CC) $(ARM_FLAGS) $(ARM_DEFS) $(ARM_INCS) -MMD -MP -
MF"./$(@:%.o=%.d)" -MT"./$(@:%.o=%.o)" -MT"./$(@:%.o=%.d)" -o "$@" "$<"
```

```
# PC Targets
# Targets for compiling required object files
./Debug/source/pc/%.o: ./source/pc/%.c
	$(PC_CC) $(PC_FLAGS) -MMD -MP -MF"./$(@:%.o=%.d)" -MT"./$(@:%.o=%.o)" -
MT"./$(@:%.o=%.d)" -o "$@" "$<"


# Target for cleaning builds
clean:
	rm -rf Debug
```

Source Code

main.c

```
/*
 * main.c
 *
 *  Created on: Sep 28, 2019
 *  Author: Atharva Nandanwar
 *  Email: Atharva.Nandanwar@Colorado.EDU
 *
 */

#include "main.h"

uint16_t time_table[20] = {3000, 1000, 2000, 600, 1000, 400, 1000,\
                                  200, 500, 100, 500, 100, 500, 100, 1000,\
                                              200, 1000, 400, 2000, 600};

/*
 * Function - main
 * Arguments - none
 * Brief - Manages the logic to toggle the LEDs
 *
 */
int main(void)
{
	// Variable to hold RBG loop iterations
	uint8_t looper = 0;
	// Initialize system
	proc_init();

	// Loop for doing 10 cycles
	for (uint8_t i = 0; i < 10; i++)
	{
		// Loop for cycling in look up table
		for (uint8_t j = 0; j < 20; j++)
		{
			// Switch between alternative ON and OFF
			mode = looper % 2;
			// Change color after 6 led_execute
			flag = looper / 6;
			// Function for LED execution
			led_execute();
			// Delay
			loop(time_table[j]);

			looper++;
			(looper == 18)?looper = 0:looper;

			// Conditional execution for Debug
#if defined(FB_DEBUG) || defined(PC_DEBUG)
			debug(time_table[j]);
#endif
		}
	}
	return 0;
}
```

---

main.h

```
/*
 * main.h
 *
 *  Created on: Sep 28, 2019
 *  Author: Atharva Nandanwar
 *  Email: Atharva.Nandanwar@Colorado.EDU
 *
 */

#ifndef SOURCE_MAIN_H_
```

```c
#define SOURCE_MAIN_H_
#endif /* SOURCE_MAIN_H_ */

// Standard inclusions to support fixed width integers,
// and time constructs
#include <stdint.h>
#include <time.h>

// Data types to store processor clock counts,
// and difference between two events
clock_t prevEvent;
clock_t thisEvent;
double diffEvent;

// Struct to store time_data
struct tm *time_data;

// Data types to store flag value, and mode
uint8_t flag;
uint8_t mode;

// Conditional execution according to different
// builds
#ifdef PC_RUN
#include "pc_loop.h"  // contains function with PC loop logic
#include "pc_print.h" // contains function with PC print logic
#elif PC_DEBUG
#include "pc_loop.h"
#include "pc_print.h"
#include "pc_debug.h" // contains function with debug logic
#elif FB_RUN
#include "fb_loop.h"    // contains function with Freedom Board
                                            // loop logic
#include "fb_led.h"          // contains function with Freedom Board
                                            // led logic
#elif FB_DEBUG
#include "fb_loop.h"
#include "fb_led.h"
#include "fb_debug.h" // contains function with Freedom Board
                                            // debug logic
#endif
```

pc_print.h

```
/*
         * pc_print.h
         *
         *  Created on: Oct 1, 2019
         *  Author: Atharva Nandanwar
         *  Email: Atharva.Nandanwar@Colorado.EDU
         *
         */

        #ifndef PC_PC_PRINT_H_
        #define PC_PC_PRINT_H_
        #endif /* PC_PC_PRINT_H_ */

        #include <stdio.h>
        #include <stdint.h>

        // Macros to simplify understanding which color
        // is referred
        #define RED (0)
        #define BLUE (1)
        #define GREEN (2)

        // Variables declared in main
        extern uint8_t flag;
        extern uint8_t mode;

        void proc_init(void);
        void led_execute(void);
```

---

pc_print.c

```
/*
         * pc_print.c
         *
         *  Created on: Sep 28, 2019
         *  Author: Atharva Nandanwar
         *  Email: Atharva.Nandanwar@Colorado.EDU
         *
         */

        #include "pc_print.h"
```

```c
/*
 * Function - proc_init
 * Arguments - none
 * Brief - Just a placeholder function
 *
 */
void proc_init(void)
{
        while(0)
        {
                ;
        }
}

/*
 * Function - led_execute
 * Arguments - none
 * Brief - PC version of execution - just prints LED ON/OFF
 *
 */
void led_execute(void)
{
        // char pointers to hold strings
        char *led = NULL;
        char *state = NULL;

        // Logic for color
        if (flag == RED)
        {
                led = "RED";
        }
        else if (flag == BLUE)
        {
                led = "BLUE";
        }
        else if (flag == GREEN)
        {
                led = "GREEN";
        }

        // Logic for ON/OFF
        if (mode == 1)
```

```
			{
				state = "OFF";
			}
			else if (mode == 0)
			{
				state = "ON";
			}

			// Printing them
			printf("\nLED %5s %3s\t", led, state);
		}
```

---

pc_loop.h

```
 /*
	 * pc_loop.h
	 *
	 *  Created on: Sep 28, 2019
	 *  Author: Atharva Nandanwar
	 *  Email: Atharva.Nandanwar@Colorado.EDU
	 *
	 */

	#ifndef PC_PC_LOOP_H_
	#define PC_PC_LOOP_H_
	#endif /* PC_PC_LOOP_H_ */

	#include <stdint.h>
	#include <time.h>

	void loop(uint16_t number);
```

---

pc_loop.c

```
 /*
	 * pc_loop.c
	 *
	 *  Created on: Sep 28, 2019
	 *  Author: Atharva Nandanwar
	 *  Email: Atharva.Nandanwar@Colorado.EDU
	 *
	 */
```

```c
#include "pc_loop.h"

/*
 * Function - loop
 * Arguments -
 * number : taking number to generate desirable delay time
 * Brief - creates a delay based on number
 *
 */
void loop(uint16_t number)
{
        // loop_var will be used to generate delay
        uint64_t loop_var = number * 680000;
        while(loop_var != 0)
        {
                loop_var--;
                // Assembly instruction to do nothing
                __asm volatile ("nop");
        }
}
```

---

pc_debug.h

```c
/*
 * pc_debug.h
 *
 *  Created on: Oct 1, 2019
 *  Author: Atharva Nandanwar
 *  Email: Atharva.Nandanwar@Colorado.EDU
 *
 */

#ifndef PC_PC_DEBUG_H_
#define PC_PC_DEBUG_H_
#endif /* PC_PC_DEBUG_H_ */

#include <stdio.h>
#include <time.h>
#include <stdint.h>

// Variables declared in main
extern clock_t prevEvent;
extern clock_t thisEvent;
extern double diffEvent;
extern struct tm *time_data;

void debug(uint16_t loop_num);
```

---

pc_debug.c

```
/*
 * pc_debug.c
 *
 *  Created on: Oct 1, 2019
 *  Author: Atharva Nandanwar
 *  Email: Atharva.Nandanwar@Colorado.EDU
 *
 */


#include "pc_debug.h"

// Leveraged Code - https://stackoverflow.com/questions/5248915/execution-time-of-
c-program
// Time Difference

/*
 * Function - debug
 * Arguments -
 * loop_num : for pc version, this number means nothing
 * Brief - prints out debug information
 *
 */
void debug(uint16_t loop_num)
{
	// Algorithm to calculate time difference
	prevEvent = thisEvent;
	thisEvent = clock();
	diffEvent = (double)(thisEvent - prevEvent)/CLOCKS_PER_SEC;
	diffEvent = diffEvent * 1000;

	// Code to get timestamp
	time_t set_time = time(NULL);
	time_data = localtime(&set_time);
	printf("%02d:%02d:%02d  %-4.3lf\n", time_data -> tm_hour, \
			time_data -> tm_min, time_data -> tm_sec, \
                        diffEvent);
        }
```

---

fb_loop.h

```
 /*
         * fb_loop.h
         *
         *  Created on: Sep 28, 2019
         *  Author: Atharva Nandanwar
         *  Email: Atharva.Nandanwar@Colorado.EDU
         *
         */

        #ifndef FB_FB_LOOP_H_
        #define FB_FB_LOOP_H_
        #endif /* FB_FB_LOOP_H_ */

        #include <stdint.h>

        void loop(uint16_t number);
```

---

fb_loop.c

```
 /*
         * fb_loop.c
         *
         *  Created on: Sep 28, 2019
         *  Author: Atharva Nandanwar
         *  Email: Atharva.Nandanwar@Colorado.EDU
         *
         */

        #include "fb_loop.h"

        /*
         * Function - loop
         * Arguments -
         * number : used to generate variable delay
         * Brief - produces variable delay for freedom board
         *
         */
        void loop(uint16_t number)
        {
                uint64_t loop_var = number * 2300;
                while((loop_var) != 0)
                {
                        loop_var--;
                        __asm volatile ("nop");
                }
        }
```

---

fb_led.h

```
 /*
          * fb_led.h
          *
          *  Created on: Sep 28, 2019
          *  Author: Atharva Nandanwar
          *  Email: Atharva.Nandanwar@Colorado.EDU
          *
          */

        #ifndef FB_FB_LED_H_
        #define FB_FB_LED_H_
        #endif /* FB_FB_LED_H_ */

        #include <stdint.h>

        // Macros to simplify things
        #define RED             (0)
        #define BLUE            (1)
        #define GREEN           (2)
        #define RED_GPIO        BOARD_LED_RED_GPIO
        #define BLUE_GPIO       BOARD_LED_BLUE_GPIO
        #define GREEN_GPIO      BOARD_LED_GREEN_GPIO
        #define RED_PIN         BOARD_LED_RED_GPIO_PIN
        #define BLUE_PIN        BOARD_LED_BLUE_GPIO_PIN
        #define GREEN_PIN       BOARD_LED_GREEN_GPIO_PIN
        #define LOW             1
        #define HIGH            0

        // Variables declared in main
        extern uint8_t flag;
        extern uint8_t mode;
```

```
void proc_init(void);
void led_execute(void);
```

---

fb_led.c

```
/*
 * fb_led.c
 *
 *  Created on: Sep 28, 2019
 *  Author: Atharva Nandanwar
 *  Email: Atharva.Nandanwar@Colorado.EDU
 *
 */

#include "fb_led.h"
#include "pin_mux.h"
#include "fsl_gpio.h"
#include "board.h"

/*
 * Function - proc_init
 * Arguments - none
 * Brief - Initialize system peripherals in desired state
 *
 */
void proc_init(void)
{
	// Initialize Pins, Clocks, and DebugConsole
	BOARD_InitPins();
	BOARD_BootClockRUN();
	//BOARD_InitDebugConsole();
	gpio_pin_config_t red_led = {
			kGPIO_DigitalOutput, 1,
	    };
	gpio_pin_config_t blue_led = {
			kGPIO_DigitalOutput, 1,
	    };
	gpio_pin_config_t green_led = {
			kGPIO_DigitalOutput, 1,
	    };
	GPIO_PinInit(RED_GPIO, RED_PIN, &red_led);
	GPIO_WritePinOutput(RED_GPIO, RED_PIN, LOW);
	GPIO_PinInit(BLUE_GPIO, BLUE_PIN, &blue_led);
```

```
		GPIO_WritePinOutput(BLUE_GPIO, BLUE_PIN, LOW);
		GPIO_PinInit(GREEN_GPIO, GREEN_PIN, &green_led);
		GPIO_WritePinOutput(GREEN_GPIO, GREEN_PIN, LOW);
	}

	/*
	 * Function - led_execute
	 * Arguments - none
	 * Brief - Freedom Board version of execution
	 *
	 */
	void led_execute(void)
	{
		char *state = NULL;
		if (flag == RED)
		{
			GPIO_WritePinOutput(RED_GPIO, RED_PIN, mode);
		}
		else if (flag == BLUE)
		{
			GPIO_WritePinOutput(BLUE_GPIO, BLUE_PIN, mode);
		}
		else if (flag == GREEN)
		{
			GPIO_WritePinOutput(GREEN_GPIO, GREEN_PIN, mode);
		}
		state = (mode == 1)?"ON":"OFF";
	}
```

---

fb_debug.h

```
 /*
	 * pc_debug.h
	 *
	 *  Created on: Oct 1, 2019
	 *  Author: Atharva Nandanwar
	 *  Email: Atharva.Nandanwar@Colorado.EDU
	 *
	 */

	#ifndef FB_FB_DEBUG_H_
	#define FB_FB_DEBUG_H_
	#endif /* FB_FB_DEBUG_H_ */
```

```
#include <stdint.h>

#define RED (0)
#define BLUE (1)
#define GREEN (2)

extern uint8_t flag;
extern uint8_t mode;

void debug(uint16_t loop_num);
static void print(void);
```

---

fb_debug.c

```
/*
 * fb_debug.c
 *
 *  Created on: Oct 1, 2019
 *  Author: Atharva Nandanwar
 *  Email: Atharva.Nandanwar@Colorado.EDU
 *
 */

// Including all the required header files
#include "fb_debug.h"
#include "board.h"
#include "fsl_debug_console.h"

/*
 * Function - debug
 * Arguments -
 * loop_num : used to print the loop counter value used when creating
 * delay
 * Brief - printing debug information
 *
 */
void debug(uint16_t loop_num)
{
	print();
	PRINTF("\t%d\n", loop_num * 2300);
}
```

```c
/*
 * Function - print
 * Arguments - void
 * Brief - collects and prints correct debug information
 * This is logically similar to led_execute function in pc_print.c
 * however, I didn't choose to abstract it out
 */
static void print(void)
{
        // char pointers to hold strings
        char *led = NULL;
        char *state = NULL;

        // Logic for color
        if (flag == RED)
        {
                led = "RED";
        }
        else if (flag == BLUE)
        {
                led = "BLUE";
        }
        else if (flag == GREEN)
        {
                led = "GREEN";
        }

        // Logic for ON/OFF
        if (mode == 1)
        {
                state = "OFF";
        }
        else if (mode == 0)
        {
                state = "ON";
        }

        // Printing them to console
        PRINTF("LED %5s %3s\t", led, state);
}
```

README

PES Project 2

Readme

This repository contains all the required files for PES Project 2.

Recommended compiler for build targets fb_run fb_debug - use MCUXpresso to build and debug pc_run pc_debug - use gcc, and bash terminal to execute

Note - makefile included, and these builds can be built using `make BV=FB_RUN` or `make BV=PC_DEBUG`
Folder Structure:

1. source - contains source files
2. source/fb - contains source files for freedom board targets
3. source/pc - contains source files for pc based targets
4. include - include files only for freedom board targets
5. startup - source file for startup code
6. utilities, CMSIS, drivers, board - files required for Freedom board configuration, and drivers
7. linkerfile.ld - file used by arm-linker
8. makefile - make file for the project