```c
/*******************************************************************************
*                                                                              *
*                       PRINCIPLES OF EMBEDDED SOFTWARE                       *
*                                  PROJECT 2                                   *
*   Project By:GITANJALI SURESH (GISU9983) & RUCHA BORWANKAR (RUBO1268)       *
*                   Cross Platform IDE: MCUXpresso IDE v11                     *
*                   Compiler: MinGW/GNU gcc v8.2.0 (PC version)                *
*                       Cross-Compiler: ARM GCC (FB version)                   *
*                                Main file                                    */
/*************** Header files ***************/
#include <stdio.h>                      /***** For printf() *****/
#include <stdint.h>                     /***** For standard int datatypes *****/
#include <time.h>                       /***** For delay and timestamps *****/

/*************** Header files and functions specific for FB_RUN ***************/

#ifdef FB_RUN
	#include "led.h"
	void Blink(void);
	void led_Initialize(void);
#endif

/*************** Header files and functions specific for FB_DEBUG ***************/
#ifdef FB_DEBUG
	#include "led.h"
	void Blink(void);
	void led_Initialize(void);
#endif

/*************** Global variable declarations ***************/

uint32_t t_cyc[20]=
{3000,1000,2000,600,1000,400,1000,200,500,100,500,100,500,100,1000,200,1000,400,2000,600};
                    /***** Timing Loop *****/
volatile static uint32_t clr;    /***** variable for 3 on and off cycles *****/
uint32_t count;                  /***** variable for 10 complete cycles *****/

/*************** Function prototype declarations ***************/
void delay(uint32_t dly);
void timestamp(void);

/*************** Main Function ***************/
int main(void)
{
	/*************** FB Run version ***************/
	#ifdef FB_RUN
		led_Initialize();          /***** Initialization of LEDs on board *****/
		Blink();                   /***** Blinking of LEDs *****/
	#endif
	/*************** FB Debug version ***************/
	#ifdef FB_DEBUG
		led_Initialize();          /***** Initialization of LEDs on board *****/
		Blink();                   /***** Blinking of LEDs *****/
	#endif

/*************** PC Run version ***************/
	#ifdef PC_RUN
		count = 9;
		uint32_t cyc = 0;       /***** Variable indexing the timing loop *****/
		while(count!=0)
		{/***** Checking for the start of program *****/
```

```c
                    if(count == 9 && cyc == 0)
                    {
                          printf("\nPC Run Version");
                          printf("\nProgram Start... \n");
                          cyc = 0;
                          printf("\n------------------------------------------\n");
                    /***** Printing the number of the cycle ****/
                          printf("\nCycle %u.\n\r",(10-count));
                          printf("\n----------------------------------------\n");
                    }

                    /***** Looping 3 on and off cycles for every color *****/
                    /***** Color Pattern - RED GREEN BLUE *****/

                    for(clr = 3;clr > 0; cyc++,clr--)
                    {/***** Reinitializing the index of timing loop at end *****/

                          if(cyc == 20)
                                {
                                      cyc = 0;
                                      count--;
                                      printf("\n--------------------------\n");
                                      printf("\nCycle %u.\n\r",(10-count));
                                      printf("\n--------------------------\n");
                                }
                    /***** Printing the LED status along with their wait time *****/
                          printf("\nRED LED ON for %u ms.",t_cyc[cyc]);
                          delay(t_cyc[cyc]);
                          printf("\nRED LED OFF for %u ms.",t_cyc[++cyc]);
                          delay(t_cyc[cyc]);
                          printf("\n");
                    }
                    for(clr = 3;clr > 0; cyc++,clr--)
                    {
                          if(cyc == 20)
                                {
                                      cyc = 0;
                                      count--;
                                      printf("\n---------------------------\n");
                                      printf("\nCycle %u.\n\r",(10-count));
                                      printf("\n---------------------------\n");
                                }
                          printf("\nGREEN LED ON for %u ms.",t_cyc[cyc]);
                          delay(t_cyc[cyc]);
                          printf("\nGREEN LED OFF for %u ms.",t_cyc[++cyc]);
                          delay(t_cyc[cyc]);
                          printf("\n");
                    }

                    for(clr = 3;clr > 0; cyc++,clr--)
                    {
                          if(cyc == 20)
                                {
                                      cyc = 0;
                                      count--;
                                      printf("\n----------------------------\n");
                                      printf("\nCycle %u.\n\r",(10-count));
                                      printf("\n----------------------------\n");
                                }
                          printf("\nBLUE LED ON for %u ms.",t_cyc[cyc]);
                          delay(t_cyc[cyc]);
```

```
                        printf("\nBLUE LED OFF for %u ms.",t_cyc[++cyc]);
                        delay(t_cyc[cyc]);
                        printf("\n");
                 }
                 /***** For finishing the timing cycles in the last count *****/
                if(count == 0)
                {
                        while(cyc!=20)
                                {
                                    printf("\nRED LED ON for %u ms.",t_cyc[cyc]);
                                    delay(t_cyc[cyc]);
                                    printf("\nRED LED OFF for %u Ms.",t_cyc[++cyc]);
                                    delay(t_cyc[cyc]);
                                    printf("\n");
                                    cyc++;
                                }
                 }
                /***** For terminating the PC Run version *****/
               if(count == 0 && cyc == 20)
                {
                        printf("\nPC Run Version");
                        printf("\nProgram End! ");

                 }
        }
        #endif


        /*************** PC Debug version ****************/
    #ifdef PC_DEBUG
        count = 9;
        uint32_t cyc = 0;
        /***** Variable indexing the timing loop *****/
        while(count!=0)
        {
                if(count == 9 && cyc == 0)
        /***** Checking for the start of program *****/
                {
                        printf("\nPC Run Version");
                        printf("\nProgram Start... \n");
                        cyc = 0;
                        printf("\n----------------------------------------\n");
                        printf("\nCycle %u.\n\r",(10-count));
                        /***** Printing the number of the cycle ****/
                        printf("\n----------------------------------------\n");
                }
        /***** Looping 3 on and off cycles for every color *****/
        /***** Color Pattern - RED GREEN BLUE *****/
                for(clr = 3;clr > 0; cyc++,clr--)
                {
                        if(cyc == 20)
        /***** Reinitializing the index of timing loop at end *****/
                        {
                                cyc = 0;
                                count--;
                                printf("\n----------------------------\n");
                                printf("\nCycle %u.\n\r",(10-count));
                                printf("\n----------------------------\n");
                        }
    /*****Printing the LED status along with the timestamp & time since previous event *****/
                printf("\nRED LED ON");
                timestamp();
                printf(" %u ms",t_cyc[cyc-1]);
                delay(t_cyc[cyc]);
                if(cyc == 19)
/***** for looping around the time loop once it reaches the end *****/
                {
                    printf("\nRED LED OFF");
                    timestamp();
                    printf(" %u ms",t_cyc[19]);
                    delay(t_cyc[0]);
                    printf("\n");
                    cyc = 0;
                    count--;
                    if(count != 0)
                        {
                            printf("\n------------\n");
                            printf("\nCycle %u.\n\r",(10-count));
                            printf("\n------------\n");
                        }
                }
                else
                {
                    printf("\nRED LED OFF");
                    ++cyc;
                    timestamp();
                    printf(" %u ms",t_cyc[cyc-1]);
                    delay(t_cyc[cyc]);
                    printf("\n");
                }
            }
            for(clr = 3;clr > 0; cyc++,clr--)
            {
                    if(cyc == 20)
                    {
                        cyc = 0;
                        count--;
                        printf("\n---------------------------------\n");
                        printf("\nCycle %u.\n\r",(10-count));
                        printf("\n---------------------------------\n");
                    }
                printf("\nGREEN LED ON");
                timestamp();
                printf(" %u ms",t_cyc[cyc-1]);
                delay(t_cyc[cyc]);
                    if(cyc == 19)
                    {
                        printf("\nGREEN LED OFF");
                        timestamp();
                        printf(" %u ms",t_cyc[19]);
                        delay(t_cyc[0]);
                        printf("\n");
                    }
                    else
                    {
                        printf("\nGREEN LED OFF");
                        ++cyc;
                        timestamp();
                        printf(" %u ms",t_cyc[cyc-1]);
                        delay(t_cyc[cyc]);
                        printf("\n");
                    }
            }
            for(clr = 3;clr > 0; cyc++,clr--)
            {
                    if(cyc == 20)
                    {
                        cyc = 0;
                        count--;
                        printf("\n----------------------------------\n");
                        printf("\nCycle %u.\n\r",(10-count));
                        printf("\n----------------------------------\n");
                    }
                printf("\nBLUE LED ON");
                timestamp();
                printf(" %u ms",t_cyc[cyc-1]);
                delay(t_cyc[cyc]);
                    if(cyc == 19)
                    {
                        printf("\nBLUE LED OFF");
                        timestamp();
                        printf(" %u ms",t_cyc[19]);
                        delay(t_cyc[0]);
                        printf("\n");
                    }
                    else
                    {
                        printf("\nBLUE LED OFF");
                        ++cyc;
                        timestamp();
                        printf(" %u ms",t_cyc[cyc-1]);
                        delay(t_cyc[cyc]);
                        printf("\n");
                    }
            }
        /***** For finishing the timing cycles in the last count *****/
        if(count == 0)
        {
            while(cyc!=20)
            {
                printf("\nRED LED ON");
                timestamp();
                printf(" %u ms",t_cyc[cyc-1]);
                delay(t_cyc[cyc]);
                    if(cyc == 19)
                    {
                        printf("\nRED LED OFF");
                        timestamp();
                        printf(" %u ms",t_cyc[19]);
                        delay(t_cyc[0]);
                        printf("\n");
                         cyc = 0;
                          count--;
                            if(count != 0)
                            {
                                printf("\n----------------------\n");
                                printf("\nCycle %u.\n\r",(10-count));
                                printf("\n----------------\n");
                            }
                    }
                    else
```

```
                                {
                                        printf("\nRED LED OFF");
                                        ++cyc;
                                        timestamp();
                                        printf(" %u ms",t_cyc[cyc-1]);
                                        delay(t_cyc[cyc]);
                                        printf("\n");
                                }
                                cyc++;
                        }
                }
                    /***** For terminating the PC Debug version *****/
                if(count == 0 && cyc == 20)
                        {
                                printf("\nPC Run Version");
                                printf("\nProgram End! ");
                         }
                }
   #endif
   return 0 ;
}

/*************** Delay function using clock() for 1ms delay [1] ****************/
void delay(uint32_t dly)
{
        clock_t start_time = clock();
        while (clock() < start_time + dly)
        {
                ;               /***** Wasting CPU cycles to get the desired delay ******/
        }
}

/*************** Function for generating time stamps [2] ****************/
void timestamp(void)
{
        time_t ts = time(NULL);
        struct tm *pts = localtime(&ts);
        printf(" %02d:%02d:%02d", pts->tm_hour, pts->tm_min, pts->tm_sec);
}

/********************************* References **************************************
[1] https://www.geeksforgeeks.org/time-delay-c/
[2] http://zetcode.com/articles/cdatetime/
***********************************************************************************/
```

```
/***************************************************************************
*                                                                         *
*                       PRINCIPLES OF EMBEDDED SOFTWARE                   *
*                                  PROJECT 2                              *
*   Project By:GITANJALI SURESH (GISU9983) & RUCHA BORWANKAR (RUBO1268)   *
*                   Cross Platform IDE: MCUXpresso IDE v11                *
*                   Compiler: MinGW/GNU gcc v8.2.0 (PC version)           *
*                       Cross-Compiler: ARM GCC (FB version)              *
*                                 LED.c                                   */
/*************** Header files ***************/
#include "led.h"
#include <stdint.h>
#include <stdio.h>
#include "fsl_debug_console.h"
#include "board.h"
#include "peripherals.h"
#include "pin_mux.h"
#include "clock_config.h"
#include "MKL25Z4.h"

void Delay(uint32_t dly);
void Blink(void);
void led_Initialize(void);

/*************** Global variable declarations ***************/
uint32_t t_cycle[20] =
{3000,1000,2000,600,1000,400,1000,200,500,100,500,100,500,100,1000,200,1000,400,2000,600};
volatile static uint32_t color;
uint32_t cycle, ct;

/*************** Function prototype declarations ***************/

void led_Initialize(void)
{
      BOARD_InitBootPins();
      BOARD_InitBootClocks();
      BOARD_InitBootPeripherals();
      BOARD_InitDebugConsole();
      LED_RED_INIT(1);
      LED_GREEN_INIT(1);
      LED_BLUE_INIT(1);
}

void Blink(void)
{
       cycle = 0;
       ct = 9;
       while(ct!=0)
       {
             if(ct == 9 && cycle == 0)
             {/*************** FB Run version ***************/

                    #ifdef FB_RUN
                          PRINTF("FB Run Version\n\r");
                    #endif
             /*************** FB Debug version ***************/

                    #ifdef FB_DEBUG
                          PRINTF("FB Debug Version\n\r");
                    #endif
                    PRINTF("Program Start... \n\r");
```

```
                cycle = 0;
                PRINTF("====================================================\n\r");
                PRINTF("Cycle %u.\n\r",(10-ct));
                PRINTF("======================================================\n\r");
        }

        for(color = 3;color > 0; cycle++,color--)
        {
                if(cycle == 20)
                {
                        cycle = 0;
                        ct--;
                        PRINTF("=============================================\n\r");
                        PRINTF("Cycle %u.\n\r",(10-ct));
                        PRINTF("=============================================\n\r");
                }
                #ifdef FB_RUN
                        LED_RED_ON();
                        PRINTF("RED LED ON for %u ms.\n\r",t_cycle[cycle]);
                        Delay(t_cycle[cycle]);
                        LED_RED_OFF();
                        PRINTF("RED LED OFF for %u ms.\n\r",t_cycle[++cycle]);
                        Delay(t_cycle[cycle]);
                        PRINTF("----------------------------------------------\n\r");
                #endif
                #ifdef FB_DEBUG
                        LED_RED_ON();
                        PRINTF("RED LED ON %u\n\r",t_cycle[cycle]);
                        Delay(t_cycle[cycle]);
                        LED_RED_OFF();
                        PRINTF("RED LED OFF %u\n\r",t_cycle[++cycle]);
                        Delay(t_cycle[cycle]);
                        PRINTF("----------------------------------------------\n\r");
                #endif
        }
        for(color = 3;color > 0; cycle++,color--)
        {
                if(cycle == 20)
                        {
                                cycle = 0;
                                ct--;
                                PRINTF("=========================================\n\r");
                                PRINTF("Cycle %u.\n\r",(10-ct));
                                PRINTF("=========================================\n\r");
                        }
                #ifdef FB_RUN
                        LED_GREEN_ON();
                        PRINTF("GREEN LED ON for %u ms.\n\r",t_cycle[cycle]);
                        Delay(t_cycle[cycle]);
                        LED_GREEN_OFF();
                        PRINTF("GREEN LED OFF for %u ms.\n\r",t_cycle[++cycle]);
                        Delay(t_cycle[cycle]);
                        PRINTF("----------------------------------------------\n\r");
                #endif
```

```
			#ifdef FB_DEBUG
				LED_GREEN_ON();
				PRINTF("GREEN LED ON %u\n\r",t_cycle[cycle]);
				Delay(t_cycle[cycle]);
				LED_GREEN_OFF();
				PRINTF("GREEN LED OFF %u \n\r",t_cycle[++cycle]);
				Delay(t_cycle[cycle]);
				PRINTF("-----------------------------------------------\n\r");
			#endif
		}
		for(color = 3;color > 0; cycle++,color--)
		{
			if(cycle == 20)
			{
				cycle = 0;
				ct--;
				PRINTF("=================================================\n\r");
				PRINTF("Cycle %u.\n\r",(10-ct));
				PRINTF("=================================================\n\r");
			}
			#ifdef FB_RUN
				LED_BLUE_ON();
				PRINTF("BLUE LED ON for %u ms.\n\r",t_cycle[cycle]);
				Delay(t_cycle[cycle]);
				LED_BLUE_OFF();
				PRINTF("BLUE LED OFF for %u ms.\n\r",t_cycle[++cycle]);
				Delay(t_cycle[cycle]);
				PRINTF("-----------------------------------------------\n\r");
			#endif
			#ifdef FB_DEBUG
				LED_BLUE_ON();
				PRINTF("BLUE LED ON %u \n\r",t_cycle[cycle]);
				Delay(t_cycle[cycle]);
				LED_BLUE_OFF();
				PRINTF("BLUE LED OFF %u \n\r",t_cycle[++cycle]);
				Delay(t_cycle[cycle]);
				PRINTF("-----------------------------------------------\n\r");
			#endif
		}
		if(ct == 0)
		{
			while(cycle != 20)
			{
				#ifdef FB_RUN
					LED_RED_ON();
					PRINTF("RED LED ON for %u ms.\n\r",t_cycle[cycle]);
					Delay(t_cycle[cycle]);
					LED_RED_OFF();
					PRINTF("RED LED OFF for %u ms.\n\r",t_cycle[++cycle]);
					Delay(t_cycle[cycle]);
					PRINTF("-----------------------------------------------\n\r");
					cycle++;
				#endif
```

```
                          #ifdef FB_DEBUG
                                LED_RED_ON();
                                PRINTF("RED LED ON %u\n\r",t_cycle[cycle]);
                                Delay(t_cycle[cycle]);
                                LED_RED_OFF();
                                PRINTF("RED LED OFF %u\n\r",t_cycle[++cycle]);
                                Delay(t_cycle[cycle]);
                                PRINTF("-----------------------------------------------\n\r");
                          #endif
                    }
             }
/***** For terminating the FB version *****/

             if(ct == 0 && cycle == 20)
             {
                    PRINTF("===========================================================\n\r");
                    PRINTF("PC Run Version\n\r");
                    PRINTF("Program End! \n\r");
                    PRINTF("===========================================================\n\r");

             }
      }
}
/*************** Delay function for 1ms delay approximately [1] ***************/
void Delay(uint32_t dly)
{
      uint32_t i = dly*7000;    /****** As clock is 8MHz *****/
      while(i!=0)
      {
             i-    /***** Wasting MCU cycles to get the desired delay ******/
      }
}

/*****************************************************************************
*                                                                            *
*                       PRINCIPLES OF EMBEDDED SOFTWARE                      *
*                                  PROJECT 2                                 *
*   Project By:GITANJALI SURESH (GISU9983) & RUCHA BORWANKAR (RUBO1268)      *
*                   Cross Platform IDE: MCUXpresso IDE v11                   *
*                   Compiler: MinGW/GNU gcc v8.2.0 (PC version)              *
*                       Cross-Compiler: ARM GCC (FB version)                 *
*                                   LED.h                                    */
******************************************************************************/

#ifndef LED_H_

#define LED_H_


#endif







/*****************************************************************************
```

```
*                                                                              *
*                       PRINCIPLES OF EMBEDDED SOFTWARE                        *
*                                  PROJECT 2                                   *
*   Project By:GITANJALI SURESH (GISU9983) & RUCHA BORWANKAR (RUBO1268)        *
*                   Cross Platform IDE: MCUXpresso IDE v11                     *
*                   Compiler: MinGW/GNU gcc v8.2.0 (PC version)                *
*                       Cross-Compiler: ARM GCC (FB version)                   *
*                                Make file                                     */
############################
# Command for removing files
RM := rm -rf

############################
# Include files
INCLUDES := \
	-I"CMSIS" \
	-I"source" \
	-I"board" \
	-I"drivers" \
	-I"utilities"

PC_INCLUDES := \
	-I"C:\MinGW\include"
############################
# Compiler for FB versions
CC := arm-none-eabi-gcc

############################
# Linker for FB versions
LL := arm-none-eabi-gcc

###########################################
# Binary/exectable to build for FB versions
EXE := \
  ./debug/PES_Project_2.axf

#######################################
# List of object files for FB versions
OBJS := ./debug/PES_Project_2.o \
	./debug/led.o \
	./debug/startup_mkl25z4.o \
	./debug/system_MKL25Z4.o   \
	./debug/board.o \
	./debug/clock_config.o \
	./debug/peripherals.o \
	./debug/pin_mux.o \
	./debug/fsl_clock.o \
	./debug/fsl_common.o \
	./debug/fsl_flash.o \
	./debug/fsl_gpio.o \
	./debug/fsl_lpsci.o \
	./debug/fsl_smc.o \
	./debug/fsl_uart.o ./debug/fsl_debug_console.o

############################
# List of dependency files
C_DEPS = \
	./debug/PES_Project_2.d \
	./debug/led.d \
	./debug/startup_mkl25z4.d \
	./debug/system_MKL25Z4.d \
	./debug/board.d \
	./debug/clock_config.d \
	./debug/peripherals.d \
	./debug/pin_mux.d \
	./debug/fsl_clock.d \
	./debug/fsl_common.d \
	./debug/fsl_flash.d \
	./debug/fsl_gpio.d \
	./debug/fsl_lpsci.d \
	./debug/fsl_smc.d \
	./debug/fsl_uart.d ./debug/fsl_debug_console.d

##############################################################
# Include generated dependency files (only if not clean target)
ifneq ($(MAKECMDGOALS),clean)
ifneq ($(strip $(C_DEPS)),)
-include $(C_DEPS)
endif
endif

##################################
# Compiler options for FB versions
CC_OPTIONS := \
	-c \
	-std=gnu99 \
	-O0 \
	-g3 \
	-ffunction-sections \
	-fmessage-length=0\
	-fno-common \
	-fdata-sections \
	-fno-builtin \
	-mcpu=cortex-m0plus \
	-mthumb \
	-Wall \
	-Werror

################################
# Build options for FB versions
B_OPTIONS := \
	-D__REDLIB__ \
	-DCPU_MKL25Z128VLK4 \
	-DCPU_MKL25Z128VLK4_cm0plus \
	-DSDK_OS_BAREMETAL \
	-DFSL_RTOS_BM \
	-DSDK_DEBUGCONSOLE=1 \
	-DCR_INTEGER_PRINTF \
	-DPRINTF_FLOAT_ENABLE=0 \
	-DSCANF_FLOAT_ENABLE=0 \
	-DPRINTF_ADVANCED_ENABLE=0 \
	-DSCANF_ADVANCED_ENABLE=0 \
	-D__MCUXPRESSO \
	-D__USE_CMSIS \
	-DDEBUG \
	-DFRDM_KL25Z \
	-DFREEDOM \
	-specs=redlib.specs \

################################
# Linker Options for FB versions
```

```
LL_OPTIONS := -nostdlib -Xlinker -Map="debug/MyMakeProject.map" -Xlinker --gc-sections -Xlinker -
print-memory-usage -mcpu=cortex-m0plus -mthumb -T linkerfile.ld -o $(EXE)

###########################################################
# Checking different versions and enabling necessary macros
ifeq ($(VERSION), FB_RUN)
./debug/%.o: ./source/%.c
	@echo 'Building file: $<'
	$(CC) $(CC_OPTIONS) $(B_OPTIONS) -DFB_RUN $(INCLUDES) -MMD -MP -MF"$(@:%.o=%.d)" -
MT"$(@:%.o=%.o)" -MT"$(@:%.o=%.d)" -o "$@" "$<"
	@echo 'Finished building: FB_RUN $<'
	@echo ' '

else ifeq ($(VERSION), FB_DEBUG)
./debug/%.o: ./source/%.c
	@echo 'Building file: $<'
	$(CC) $(CC_OPTIONS) $(B_OPTIONS) -DFB_DEBUG $(INCLUDES) -MMD -MP -MF"$(@:%.o=%.d)" -
MT"$(@:%.o=%.o)" -MT"$(@:%.o=%.d)" -o "$@" "$<"
	@echo 'Finished building: FB_DEBUG $<'
	@echo ' '

else ifeq ($(VERSION), PC_RUN)
CC := gcc                                    #MinGW gcc compiler [1]
EXE := \
  ./debug/PES_Project_2.exe
OBJS := ./debug/PES_Project_2.o
LL := gcc
LL_OPTIONS := -g -o $(EXE)       #Flags for converting an object file into executable file [1]
./debug/PES_Project_2.o: ./source/PES_Project_2.c
	@echo 'Building file: $<'
	$(CC) -c -std=gnu99 -DPC_RUN $(PC_INCLUDES) -MMD -MP -MF"$(@:%.o=%.d)" -MT"$(@:%.o=%.o)" -
MT"$(@:%.o=%.d)" -o "$@" "$<"
	@echo 'Finished building: PC_RUN $<'
	@echo ' '

else ifeq ($(VERSION), PC_DEBUG)
CC := gcc
EXE := \
  ./debug/PES_Project_2.exe
OBJS := ./debug/PES_Project_2.o
LL := gcc
LL_OPTIONS := -g -o $(EXE)
./debug/PES_Project_2.o: ./source/PES_Project_2.c
	@echo 'Building file: $<'
	$(CC) -c -std=gnu99 -DPC_DEBUG $(PC_INCLUDES) -MMD -MP -MF"$(@:%.o=%.d)" -MT"$(@:%.o=%.o)"
-MT"$(@:%.o=%.d)" -o "$@" "$<"
	@echo 'Finished building: PC_DEBUG $<'
	@echo ' '

endif

############################
# Main (all) target
all: $(EXE)
	@echo "*** finished building ***"

############################
# Clean target
clean:
	-$(RM) $(EXECUTABLES) $(OBJS) $(EXE)
	-$(RM) ./debug/*.map
	-@echo "*******Cleaned!*******"

############################
# Rule to link the executable
$(EXE): $(OBJS) linkerfile.ld
	@echo 'Building target: $@'
	@echo 'Invoking: Linker'
	$(LL) $(LL_OPTIONS) $(OBJS)
	@echo 'Finished building target: $@'
	@echo ' '

#############################################
# Rule to build the files in the CMSIS folder
./debug/%.o: ./CMSIS/%.c
	@echo 'Building file: $<'
	$(CC) $(CC_OPTIONS) $(B_OPTIONS) $(INCLUDES) -MMD -MP -MF"$(@:%.o=%.d)" -MT"$(@:%.o=%.o)" -
MT"$(@:%.o=%.d)" -o "$@" "$<"
	@echo 'Finished building: $<'
	@echo ' '

#############################################
# Rule to build the files in the board folder
./debug/%.o: ./board/%.c
	@echo 'Building file: $<'
	$(CC) $(CC_OPTIONS) $(B_OPTIONS) $(INCLUDES) -MMD -MP -MF"$(@:%.o=%.d)" -MT"$(@:%.o=%.o)" -
MT"$(@:%.o=%.d)" -o "$@" "$<"
	@echo 'Finished building: $<'
	@echo ' '

###############################################
# Rule to build the files in the drivers folder
./debug/%.o: ./drivers/%.c
	@echo 'Building file: $<'
	$(CC) $(CC_OPTIONS) $(B_OPTIONS) $(INCLUDES) -MMD -MP -MF"$(@:%.o=%.d)" -MT"$(@:%.o=%.o)" -
MT"$(@:%.o=%.d)" -o "$@" "$<"
	@echo 'Finished building: $<'
	@echo ' '

################################################
# Rule to build the files in the utilities folder
./debug/%.o: ./utilities/%.c
	@echo 'Building file: $<'
	$(CC) $(CC_OPTIONS) $(B_OPTIONS) $(INCLUDES) -MMD -MP -MF"$(@:%.o=%.d)" -MT"$(@:%.o=%.o)" -
MT"$(@:%.o=%.d)" -o "$@" "$<"
	@echo 'Finished building: $<'
	@echo ' '\
###############################################
# Rule to build the files in the startup folder
./debug/%.o: ./startup/%.c
	@echo 'Building file: $<'
	$(CC) $(CC_OPTIONS) $(B_OPTIONS) $(INCLUDES) -o $@ $<
	@echo 'Finished building: $<'
	@echo ' '
############################ References ################################
# [1] https://www3.ntu.edu.sg/home/ehchua/programming/cpp/gcc_make.html
#########################################################################
```