Principles of Embedded Software Project 3 Documentation

README.md

Name: Atharva Nandanwar

Source files - source folder

- source
    - main.c - main code procedures
    - main.h - header file for main
    - common.h - common declarations
    - logger/
        - logger.c - logger functions
        - logger.h - header file for logger
    - led_control/
        - led_control.c - LED Control Functions
        - led_control.h - header file for LED Control
    - mem_test/
        - allocate.c - Function for allocating memory
        - allocate.h
        - display.c - Function for displaying data
        - display.h
        - free.c - Function for freeing data

    - free.h
    - get_addr.c - Function for getting address from offset
    - get_addr.h
    - invert.c - Function for inverting data
    - invert.h
    - mem_write.c - Function for writing into memory region
    - mem_write.h
    - pattern_write.c - Function for writing pattern into memory region
    - pattern_write.h
    - verify.c - Function for verifying pattern
    - verify.h
  - pattern_gen/
    - pattern_gen.c - Function for generating pattern
    - pattern_gen.h

Compilation Instructions

1. **Target - KL25Z**

   Open the project in MCUXpresso, and in Build Targets -> KL25Z or KL25Z_LOG.

   Press Debug, and it would run the file.

   To monitor the output, open a serial monitor for appropriate port at 115200 baud, no parity, and one stop bit.

2. **Target - PC**

   Open the project in MCUXpresso, and in Build Targets -> PC or PC_LOG.

After the compilation is successful, go into the project directory -> Debug, and run the file pes_project_3.axf on Linux.

It would print all the output on standard output.

Note: if this doesn't work, then use the makefile to generate project executables. You need to have arm-none-eabi-gcc and gcc on your system to compile.

Make Commands:

```
make -r all BUILD=(BUILD_NAME)-----------------------------make -r all BUILD=
```

UML Files

Navigate to doc -> UML_Diagrams/

There are html and pdf documents, html documents are more representative of what I want to show.

Makefile

```
# Makefile for Memory Test Project
# Author : Atharva Nandanwar
# Date: 10/20/2019

######################################################################
# Build Variables

# Program for removing files
RM := rm -rf

# Program for making directories
MK := mkdir -p
```

```
# PC compiler
PC_CC := gcc

# PC linker
PC_LL := gcc

# ARM compiler
ARM_CC := arm-none-eabi-gcc

# ARM linker
ARM_LL := arm-none-eabi-gcc

#####################################################################
# PC Compiler Flags
PC_FLAGS := -c -Wall -Werror -g -DARCH_SIZE=uint64_t

# ARM Compiler Flags
ARM_FLAGS := -c \
             -std=c99 \
             -O0 \
             -g3 \
             -ffunction-sections \
             -fmessage-length=0 \
             -fno-common \
             -fdata-sections \
             -fno-builtin \
             -mcpu=cortex-m0plus \
             -mthumb \
             -DARCH_SIZE=uint32_t

# ARM Linker Flags
ARM_LL_FLAGS := -v \
                -nostdlib \
                -Xlinker -Map="./Debug/pes_project_3.map" \
                -Xlinker --gc-sections \
                -Xlinker -print-memory-usage \
                -Xlinker --sort-section=alignment \
                -Xlinker --cref \
                -mcpu=cortex-m0plus \
                -mthumb \
                -T linkerfile.ld \
                -o $(EXE)

# ARM Defines
```

```
ARM_DEFS := \
            -D__REDLIB__ \
            -DCPU_MKL25Z128VLK4 \
            -DCPU_MKL25Z128VLK4_cm0plus \
            -DSDK_OS_BAREMETAL \
            -DFSL_RTOS_BM \
            -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 \
            -DSDK_DEBUGCONSOLE=0 \
            -DSDK_DEBUGCONSOLE_UART

# Build Folders
SOURCE := ./source
DEBUG := ./Debug


# PC Include Files
PC_INCS := \
            -I"$(SOURCE)" \
            -I"$(SOURCE)/led_control" \
            -I"$(SOURCE)/logger" \
            -I"$(SOURCE)/mem_test" \
            -I"$(SOURCE)/pattern_gen" \

# PC Object Files
PC_OBJS := \
            $(DEBUG)/source/logger/logger.o \
            $(DEBUG)/source/mem_test/allocate.o \
            $(DEBUG)/source/mem_test/display.o \
            $(DEBUG)/source/mem_test/free.o \
            $(DEBUG)/source/mem_test/get_addr.o \
            $(DEBUG)/source/mem_test/invert.o \
            $(DEBUG)/source/mem_test/mem_write.o \
            $(DEBUG)/source/mem_test/pattern_write.o \
            $(DEBUG)/source/mem_test/verify.o \
            $(DEBUG)/source/pattern_gen/pattern_gen.o

 # PC Dependencies Files
 PC_DEPS := \
            $(DEBUG)/source/logger/logger.d \
            $(DEBUG)/source/mem_test/allocate.d \
            $(DEBUG)/source/mem_test/display.d \
            $(DEBUG)/source/mem_test/free.d \
            $(DEBUG)/source/mem_test/get_addr.d \
            $(DEBUG)/source/mem_test/invert.d \
            $(DEBUG)/source/mem_test/mem_write.d \
            $(DEBUG)/source/mem_test/pattern_write.d \
            $(DEBUG)/source/mem_test/verify.d \
            $(DEBUG)/source/pattern_gen/pattern_gen.d

 # ARM Include Files
 ARM_INCS := \
            -I"$(SOURCE)" \
            -I"$(SOURCE)/led_control" \
            -I"$(SOURCE)/logger" \
            -I"$(SOURCE)/mem_test" \
            -I"$(SOURCE)/pattern_gen" \
            -I"board" \
            -I"CMSIS" \
            -I"drivers" \
            -I"startup" \
            -I"utilities" \

 # ARM Object Files
 ARM_OBJS := \
            $(DEBUG)/source/logger/logger.o \
            $(DEBUG)/source/mem_test/allocate.o \
            $(DEBUG)/source/mem_test/display.o \
            $(DEBUG)/source/mem_test/free.o \
            $(DEBUG)/source/mem_test/get_addr.o \
            $(DEBUG)/source/mem_test/invert.o \
            $(DEBUG)/source/mem_test/mem_write.o \
            $(DEBUG)/source/mem_test/pattern_write.o \
            $(DEBUG)/source/mem_test/verify.o \
            $(DEBUG)/source/pattern_gen/pattern_gen.o \
            $(DEBUG)/startup/startup_mkl25z4.o \
            $(DEBUG)/CMSIS/system_MKL25Z4.o \
            $(DEBUG)/board/board.o \
            $(DEBUG)/board/clock_config.o \
            $(DEBUG)/board/peripherals.o \
            $(DEBUG)/board/pin_mux.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_smc.o \
            $(DEBUG)/drivers/fsl_uart.o \
            $(DEBUG)/utilities/fsl_debug_console.o

# ARM Dependencies Files
ARM_DEPS := \
            $(DEBUG)/source/logger/logger.d \
            $(DEBUG)/source/mem_test/allocate.d \
            $(DEBUG)/source/mem_test/display.d \
            $(DEBUG)/source/mem_test/free.d \
            $(DEBUG)/source/mem_test/get_addr.d \
            $(DEBUG)/source/mem_test/invert.d \
            $(DEBUG)/source/mem_test/mem_write.d \
            $(DEBUG)/source/mem_test/pattern_write.d \
            $(DEBUG)/source/mem_test/verify.d \
            $(DEBUG)/source/pattern_gen/pattern_gen.d \
            $(DEBUG)/startup/startup_mkl25z4.d \
            $(DEBUG)/CMSIS/system_MKL25Z4.d \
            $(DEBUG)/board/board.d \
            $(DEBUG)/board/clock_config.d \
            $(DEBUG)/board/peripherals.d \
            $(DEBUG)/board/pin_mux.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_smc.d \
            $(DEBUG)/drivers/fsl_uart.d \
            $(DEBUG)/utilities/fsl_debug_console.d

# Executable file
EXE := $(DEBUG)/pes_project_3.axf

######################################################################
# Build Rules
# Rules for making all
```

```
all : $(EXE)

######################################################################
# Selecting Platform
ifeq ($(BUILD), KL25Z)
build_option := kl25z
PLATFORM := KL25Z
else ifeq ($(BUILD), KL25Z_LOG)
build_option := kl25z_log
PLATFORM := KL25Z
else ifeq ($(BUILD), KL25Z_TESTS)
build_option := kl25z_tests
PLATFORM := KL25Z
else ifeq ($(BUILD), PC)
build_option := pc
PLATFORM := PC
else ifeq ($(BUILD), PC_LOG)
build_option := pc_log
PLATFORM := PC
else ifeq ($(BUILD), PC_TESTS)
build_option := pc_tests
PLATFORM := PC
endif
######################################################################

$(EXE) : $(build_option)

######################################################################
# Rule for making KL25Z target without logging
kl25z : directories $(ARM_OBJS) $(SOURCE)/main.c $(SOURCE)/led_control/led_con
    @$(ARM_CC) $(ARM_FLAGS) $(ARM_DEFS) $(ARM_INCS) -DKL25Z ./source/main.c -c
    @$(ARM_CC) $(ARM_FLAGS) $(ARM_DEFS) $(ARM_INCS) -DKL25Z $(SOURCE)/led_cont
    @arm-none-eabi-gcc -nostdlib -Xlinker -Map="./Debug/pes_project_3.map" -Xl
    @echo "KL25Z without logging made"

######################################################################
# Rule for making KL25Z target with logging
kl25z_log : directories $(ARM_OBJS) $(SOURCE)/main.c $(SOURCE)/led_control/led
    @$(ARM_CC) $(ARM_FLAGS) $(ARM_DEFS) $(ARM_INCS) -DKL25Z_LOG $(SOURCE)/main
    @$(ARM_CC) $(ARM_FLAGS) $(ARM_DEFS) $(ARM_INCS) -DKL25Z_LOG $(SOURCE)/led_
    @arm-none-eabi-gcc -nostdlib -Xlinker -Map="./Debug/pes_project_3.map" -Xl
    @echo "KL25Z with logging made"

######################################################################
```

```
# Rule for making PC target without logging
pc : directories $(PC_OBJS) $(SOURCE)/main.c $(SOURCE)/led_control/led_control
	@$(PC_CC) $(PC_FLAGS) $(PC_INCS) -DPC $(SOURCE)/main.c -o $(DEBUG)/source/
	@$(PC_CC) $(PC_FLAGS) $(PC_INCS) -DPC $(SOURCE)/led_control/led_control.c
	@$(PC_LL) $(DEBUG)/source/main.o $(DEBUG)/source/led_control/led_control.c
	@echo "PC without logging made"

#####################################################################
# Rule for making PC target with logging
pc_log : directories $(PC_OBJS) $(SOURCE)/main.c $(SOURCE)/led_control/led_con
	@$(PC_CC) $(PC_FLAGS) $(PC_INCS) -DPC_LOG $(SOURCE)/main.c -o $(DEBUG)/sou
	@$(PC_CC) $(PC_FLAGS) $(PC_INCS) -DPC_LOG $(SOURCE)/led_control/led_contro
	@$(PC_LL) $(DEBUG)/source/main.o $(DEBUG)/source/led_control/led_control.c
	@echo "PC with logging made"

#####################################################################
# Essesntial ARM Object Files
$(DEBUG)/board/%.o: ./board/%.c
	@echo 'Building file: $<'
	@$(ARM_CC) $(ARM_FLAGS) $(ARM_DEFS) $(ARM_INCS) -MMD -MP -MF"./$(@:%.o=%.c
	@echo 'Finished building: $<'
	@echo ' '

$(DEBUG)/CMSIS/%.o: ./CMSIS/%.c
	@echo 'Building file: $<'
	@$(ARM_CC) $(ARM_FLAGS) $(ARM_DEFS) $(ARM_INCS) -MMD -MP -MF"./$(@:%.o=%.c
	@echo 'Finished building: $<'
	@echo ' '

$(DEBUG)/drivers/%.o: ./drivers/%.c
	@echo 'Building file: $<'
	@$(ARM_CC) $(ARM_FLAGS) $(ARM_DEFS) $(ARM_INCS) -MMD -MP -MF"./$(@:%.o=%.c
	@echo 'Finished building: $<'
	@echo ' '

$(DEBUG)/startup/%.o: ./startup/%.c
	@echo 'Building file: $<'
	@$(ARM_CC) $(ARM_FLAGS) $(ARM_DEFS) $(ARM_INCS) -MMD -MP -MF"./$(@:%.o=%.c
	@echo 'Finished building: $<'
	@echo ' '

$(DEBUG)/utilities/%.o: ./utilities/%.c
	@echo 'Building file: $<'
	@$(ARM_CC) $(ARM_FLAGS) $(ARM_DEFS) $(ARM_INCS) -MMD -MP -MF"./$(@:%.o=%.c
	@echo 'Finished building: $<'
	@echo ' '

######################################################################
# Compiling files for ARM Builds
ifeq ($(PLATFORM), KL25Z)
$(DEBUG)/source/logger/logger.o : $(SOURCE)/logger/logger.c
	@echo 'Building file: $<'
	@$(ARM_CC) $(ARM_FLAGS) $(ARM_DEFS) $(ARM_INCS) -MMD -MP -MF"./$(@:%.o=%.c
	@echo 'Finished building: $<'
	@echo ' '

$(DEBUG)/source/mem_test/%.o : $(SOURCE)/mem_test/%.c
	@echo 'Building file: $<'
	@$(ARM_CC) $(ARM_FLAGS) $(ARM_DEFS) $(ARM_INCS) -MMD -MP -MF"./$(@:%.o=%.c
	@echo 'Finished building: $<'
	@echo ' '

$(DEBUG)/source/pattern_gen/%.o : $(SOURCE)/pattern_gen/%.c
	@echo 'Building file: $<'
	@$(ARM_CC) $(ARM_FLAGS) $(ARM_DEFS) $(ARM_INCS) -MMD -MP -MF"./$(@:%.o=%.c
	@echo 'Finished building: $<'
	@echo ' '
######################################################################
# Compiling files for PC Builds
else ifeq ($(PLATFORM), PC)
$(DEBUG)/source/logger/logger.o : $(SOURCE)/logger/logger.c
	@echo 'Building file: $<'
	$(PC_CC) $(PC_FLAGS) $(PC_INCS) -MMD -MP -MF"./$(@:%.o=%.d)" -MT"./$(@:%.c
	@echo 'Finished building: $<'
	@echo ' '

$(DEBUG)/source/mem_test/%.o : $(SOURCE)/mem_test/%.c
	@echo 'Building file: $<'
	$(PC_CC) $(PC_FLAGS) $(PC_INCS) -MMD -MP -MF"./$(@:%.o=%.d)" -MT"./$(@:%.c
	@echo 'Finished building: $<'
	@echo ' '

$(DEBUG)/source/pattern_gen/%.o : $(SOURCE)/pattern_gen/%.c
	@echo 'Building file: $<'
	$(PC_CC) $(PC_FLAGS) $(PC_INCS) -MMD -MP -MF"./$(@:%.o=%.d)" -MT"./$(@:%.c
	@echo 'Finished building: $<'
	@echo ' '
endif
```

```
######################################################################
# Making directories
.PHONY : directories
directories :
	$(MK) \
	$(DEBUG) \
	$(DEBUG)/board \
	$(DEBUG)/CMSIS \
	$(DEBUG)/drivers \
	$(DEBUG)/startup \
	$(DEBUG)/utilities \
	$(DEBUG)/ucunit \
	$(DEBUG)/source/led_control \
	$(DEBUG)/source/logger \
	$(DEBUG)/source/mem_test \
	$(DEBUG)/source/pattern_gen \
	$(DEBUG)/source/unit_tests

# Clean target
clean:
	@$(RM) \
	$(DEBUG)/board \
	$(DEBUG)/CMSIS \
	$(DEBUG)/drivers \
	$(DEBUG)/startup \
	$(DEBUG)/utilities \
	$(DEBUG)/source \
	$(DEBUG)/pes_project_3.axf \
	$(DEBUG)/pes_project_3.map
	@echo "Build cleaned"
```

Source Files

main.c

```
/**
  * File Name         - main.c
  * Description     - contains main program sequence
  * Author            - Atharva Nandanwar
  * Tools            - GNU C Compiler / ARM Compiler Toolchain
  * Leveraged Code     -
  * URL                -
  */

#include "main.h"

// Global Data types
logger logger_1 = {
        0,
        NULL,
        0,
        NULL,
        0,
};
logger* logger_instance = &logger_1;
ARCH_SIZE buffer_address[16];
uint8_t length = 0;

// Function declarations
uint8_t get_length(ARCH_SIZE* address, uint8_t length_of_array);
void delay(void);
#if defined(KL25Z) || defined(KL25Z_LOG)
void init(void);
#endif

// Start of main
int main(void)
{
// Board pins and peripherals initialization - KL25Z only
#if defined(KL25Z) || defined(KL25Z_LOG)
    init();
#endif

// Logger control
#if defined(KL25Z_LOG) || defined(PC_LOG)
    logger_enable();
#else
    logger_disable();
#endif

    uint32_t* base = NULL;
    ARCH_SIZE* address = NULL;
    size_t length = 16;
```

```
int8_t seed = 74;
uint8_t test_status = SUCCESS;
volatile uint8_t status;

// Starting the tests--------------------------------------------
Turn_On_Only_LED(BLUE);
// Memory allocation---------------------------------------------
base = allocate_words(length);
if(base == NULL)
{
    logger_instance->string = "Failed to allocate memory";
    log_string();
    test_status++;
}
else
{
    logger_instance->string = "Successful memory allocation";
    log_string();

}
base = (void*) 0;


// Writing pattern into allocated memory-------------------------
status = write_pattern(base, length, seed);
if(status == SUCCESS)
{
    // Display the pattern
    logger_instance->data = (ARCH_SIZE*) display_memory(base, 16);
    logger_instance->length = 16;
    log_data();
}
else
{
    logger_instance->string = "Failed to write";
    log_string();
    test_status++;
}

// Verifying the pattern-----------------------------------------
address = verify_pattern(base, length, seed);
if(address != NULL)
{
    if(address[0] == 0) // Verification successful
```

```
        {
            logger_instance->string = "Verifying Pattern - Successful verifica
            log_string();
        }
        else
        {
            logger_instance->string = "Verifying Pattern - Failure to verify";
            log_string();
            logger_instance->data = address;
            logger_instance->length = get_length(address, length);
            log_address();
            test_status++; //Since the verify pattern is supposed to fail
        }
    }
    else
    {
        logger_instance->string = "Failed - Passed NULL";
        log_string();
        test_status++; //Since NULL means failure
    }

    // Write 0xEE into a memory region-------------------------------
    if(write_memory(get_address(base, 7), 0xEE))
    {
        logger_instance->string = "Failed to write at memory location";
        log_string();
    }
    else
    {
        logger_instance->string = "Failed to write at memory location";
        log_string();
    }
    // Write 0xFF into a memory region-------------------------------
    if(write_memory(get_address(base, 8), 0xFF))
    {
        logger_instance->string = "Failed to write at memory location";
        log_string();
    }
    else
    {
        logger_instance->string = "Failed to write at memory location";
        log_string();
    }
```

```
// Display the pattern-------------------------------------------
logger_instance->data = (ARCH_SIZE*) display_memory(get_address(base, 7),
logger_instance->length = 2;
log_data();

// Verifying the pattern-----------------------------------------
address = verify_pattern(base, length, seed);
if(address != NULL)
{
    if(address[0] == 0) // Verification successful
    {
        logger_instance->string = "Verifying Pattern - Successful verifica
        log_string();
        test_status++; //Since the verify pattern is supposed to fail
    }
    else
    {
        logger_instance->string = "Verifying Pattern - Failure to verify";
        log_string();
        logger_instance->data = address;
        logger_instance->length = get_length(address, length);
        log_address();
    }
}
else
{
    logger_instance->string = "Verify Failed - Passed NULL";
    log_string();
    test_status++;
}

// Write the pattern---------------------------------------------
status = write_pattern(base, length, seed);
if(status == SUCCESS)
{
    // Displaying the pattern
    logger_instance->data = (ARCH_SIZE*) display_memory(base, 16);
    logger_instance->length = 16;
    log_data();
}
else
{
    logger_instance->string = "Failed to write";
```

```
        log_string();
        test_status++;
    }

    // Verifying the pattern-----------------------------------------
    address = verify_pattern(base, length, seed);
    if(address != NULL)
    {
        if(address[0] == 0) // Verification successful
        {
            logger_instance->string = "Verifying Pattern - Successful verifica
            log_string();
        }
        else
        {
            logger_instance->string = "Verifying Pattern - Failure to verify";
            log_string();
            logger_instance->data = address;
            logger_instance->length = get_length(address, length);
            log_address();
            test_status++; //Since the verify pattern is supposed to fail
        }
    }
    else
    {
        logger_instance->string = "Failed - Passed NULL";
        log_string();
        test_status++;
    }

    // Invert a block of memory--------------------------------------
    status = invert_block(get_address(base, 9), 4);
    if(status == SUCCESS)
    {
        // Display the pattern
        logger_instance->data = (ARCH_SIZE*) display_memory(get_address(base,
        logger_instance->length = 4;
        log_data();
    }
    else
    {
        logger_instance->string = "Failed to invert";
        log_string();
        test_status++;
```

```
}

// Verifying the pattern-----------------------------------------
address = verify_pattern(base, length, seed);
if(address != NULL)
{
    if(address[0] == 0) // Verification successful
    {
        logger_instance->string = "Verifying Pattern - Successful verifica
        log_string();
        test_status++; //Since the verify pattern is supposed to fail
    }
    else
    {
        logger_instance->string = "Verifying Pattern - Failure to verify";
        log_string();
        logger_instance->data = address;
        logger_instance->length = get_length(address, length);
        log_address();
    }
}
else
{
    logger_instance->string = "Failed - Passed NULL";
    log_string();
    test_status++;
}


// Inverting a block of memory-----------------------------------
status = invert_block(get_address(base, 9), 4);
if(status == SUCCESS)
{
    logger_instance->data = (ARCH_SIZE*) display_memory(get_address(base,
    logger_instance->length = 4;
    log_data();
}
else
{
    logger_instance->string = "Failed to invert";
    log_string();
    test_status++;
}
```

```
    // Verifying the pattern-----------------------------------------
    address = verify_pattern(base, length, seed);
    if(address != NULL)
    {
        if(address[0] == 0) // Verification successful
        {
            logger_instance->string = "Verifying Pattern - Successful verifica
            log_string();
        }
        else
        {
            logger_instance->string = "Verifying Pattern - Failure to verify";
            log_string();
            logger_instance->data = address;
            logger_instance->length = get_length(address, length);
            log_address();
            test_status++; //Since the verify pattern is supposed to fail
        }
    }
    else
    {
        logger_instance->string = "Failed - Passed NULL";
        log_string();
        test_status++;
    }

    // LED Test Status------------------------------------------------
    delay();
    if (test_status == SUCCESS)
    {
        Turn_On_Only_LED(GREEN);
    }
    else
    {
        Turn_On_Only_LED(RED);
    }
    free_words(base);
    return 0;
}


/* Function definitions */
// To determine how many defunct addresses are present
uint8_t get_length(ARCH_SIZE* address, uint8_t length_of_array)
```

```
{
    uint8_t length = 0;
    for (uint8_t i = 0; i < length_of_array; i++)
    {
        if (*(address + i) != 0)
            length++;
    }
    return length;
}

// Just some minor delay
void delay(void)
{
    volatile uint32_t i = 2300 * 3000;
    while(i != 0)
    {
        i--;
        __asm volatile ("nop");
    }
}

#if defined(KL25Z) || defined(KL25Z_LOG)
void init(void)
{
    /* Init board hardware. */
    BOARD_InitBootPins();
    BOARD_InitBootClocks();
    BOARD_InitBootPeripherals();
    /* Init FSL debug console. */
    BOARD_InitDebugConsole();
}
#endif
```

main.h

```
/**
  * File Name         - main.h
  * Description     - header file for main.c
  * Author            - Atharva Nandanwar
  * Tools            - GNU C Compiler / ARM Compiler Toolchain
  * Leveraged Code     -
  * URL                -
  */


#ifndef MAIN_H_
#define MAIN_H_
#include <stdlib.h>
#include "common.h"
#if defined(KL25Z) || defined(KL25Z_LOG)
#include "board.h"
#include "peripherals.h"
#include "pin_mux.h"
#include "clock_config.h"
#include "MKL25Z4.h"
#include "fsl_debug_console.h"
#endif
#include "mem_test/allocate.h"
#include "mem_test/free.h"
#include "mem_test/pattern_write.h"
#include "mem_test/mem_write.h"
#include "mem_test/verify.h"
#include "mem_test/display.h"
#include "mem_test/get_addr.h"
#include "mem_test/invert.h"
#include "logger/logger.h"
#include "led_control/led_control.h"
#endif /* MAIN_H_ */
```

common.h

```
/**
  * File Name         - common.h
  * Description     - common header file with global data structures
  * Author            - Atharva Nandanwar
  * Tools            - GNU C Compiler / ARM Compiler Toolchain
  * Leveraged Code     -
  * URL                -
  */
```

```
#ifndef COMMON_H_
#define COMMON_H_
#include <stdint.h>
typedef enum mem_status
{
    SUCCESS = 0,
    FAIL,
    OUT_OF_MEMORY
} mem_status;

typedef struct logger{
    uint8_t status;
    char* string;
    uint32_t integer;
    ARCH_SIZE * data;
    size_t length;
}logger;
#endif /* COMMON_H_ */
```

led_control/led_control.c

```
/**
  * File Name         - led_control.c
  * Description     - contains function for turning on LEDs
  * Author            - Atharva Nandanwar
  * Tools            - GNU C Compiler / ARM Compiler Toolchain
  * Leveraged Code     -
  * URL                -
  */

#include "led_control.h"

void Turn_On_Only_LED(uint8_t LED)
{
    // LED_string is used to print messages
    char* LED_string = NULL;
    // KL25Z board specific LED operations
#if defined(KL25Z) || defined(KL25Z_LOG)
    LED_RED_INIT(LOGIC_LED_OFF);
    LED_BLUE_INIT(LOGIC_LED_OFF);
    LED_GREEN_INIT(LOGIC_LED_OFF);
#endif
    if(LED == RED)
    {
        LED_string = "RED";
        printf("LED %s is ON\n\r", LED_string);
#if defined(KL25Z) || defined(KL25Z_LOG)
        LED_RED_ON();
        LED_GREEN_OFF();
        LED_BLUE_OFF();
#endif
    }
    else if (LED == BLUE)
    {
        LED_string = "BLUE";
        printf("LED %s is ON\n\r", LED_string);
#if defined(KL25Z) || defined(KL25Z_LOG)
        LED_RED_OFF();
        LED_GREEN_OFF();
        LED_BLUE_ON();
#endif

    }
    else if (LED == GREEN)
    {
        LED_string = "GREEN";
        printf("LED %s is ON\n\r", LED_string);
#if defined(KL25Z) || defined(KL25Z_LOG)
        LED_RED_OFF();
        LED_GREEN_ON();
        LED_BLUE_OFF();
#endif
    }
}
```

led_control/led_control.h

```
/**
  * File Name         - led_control.h
  * Description     - header file for led_control.c
  * Author            - Atharva Nandanwar
  * Tools              - GNU C Compiler / ARM Compiler Toolchain
  * Leveraged Code       -
  * URL                  -
  */


#ifndef LED_CONTROL_H_
#define LED_CONTROL_H_
#include <stdio.h>
#include <stdint.h>
#if defined(KL25Z) || defined(KL25Z_LOG)
#include "board.h"
#endif
#define RED     0
#define BLUE      1
#define GREEN      2
void Turn_On_Only_LED(uint8_t LED_Macro);
#endif /* LED_CONTROL_H_ */
```

logger/logger.c

```
/**
  * File Name          - logger.c
  * Description      - contains logger functions
  * Author             - Atharva Nandanwar
  * Tools              - GNU C Compiler / ARM Compiler Toolchain
  * Leveraged Code       -
  * URL                  -
  */


#include "logger.h"

void logger_enable(void)
{
    logger_instance->status = 1;
    printf("Logger Instance - Logger ON-----------------\n\r");
}

void logger_disable(void)
{
    logger_instance->status = 0;
    printf("Logger Instance - Logger OFF-----------------\n\r");
}

uint8_t logger_status(void)
{
    return logger_instance->status;
}

void log_string(void)
{
    if(logger_instance -> status == 1)
    {
        printf("%s\n\r", logger_instance->string);
    }
}

// Used to print byte data from given memory address
void log_data(void)
{
    if(logger_instance -> status == 1)
    {
        printf("Logger Instance - dumping data----------\n\r");
        uint8_t* temp = (uint8_t *) logger_instance->data;
        volatile uint8_t i;
        printf("Address      Data\n\r");
        for (i = 0; i < logger_instance->length; i++)
        {
            printf("%p - %#02x\n\r", (temp + i), *(temp + i));
        }
    }
}

// Used to print addresses from given memory address
// Use case - verify pattern
void log_address(void)
{
    if(logger_instance -> status == 1)
    {
        printf("Logger Instance - defunct addresses----------\n\r");
        ARCH_SIZE* temp = logger_instance->data;
        volatile uint8_t i;
        printf("Addresses\n\r");
        for (i = 0; i < logger_instance->length; i++)
        {
            printf("%#lx\n\r", *(temp + i));
        }
    }
}

void log_int()
{
    if(logger_instance -> status == 1)
    {
        printf("Logger Instance - printing integer------\n\r");
        printf("%d\n\r", logger_instance->integer);
    }
}
```

logger/logger.h

```
/**
  * File Name         - logger.h
  * Description     - header file for logger.c
  * Author            - Atharva Nandanwar
  * Tools            - GNU C Compiler / ARM Compiler Toolchain
  * Leveraged Code     -
  * URL                -
  */

#ifndef LOGGER_H_
#define LOGGER_H_
#include <stdio.h>
#include <stdint.h>
#include "common.h"
void logger_enable(void);
void logger_disable(void);
uint8_t logger_status(void);
void log_string(void);
void log_data(void);
void log_address(void);
void log_int(void);
extern logger* logger_instance;
#endif /* LOGGER_H_ */
```

mem_test/allocate.c

```
/**
  * File Name         - allocate.c
  * Description     - contains function which allocates memory
  * Author            - Atharva Nandanwar
  * Tools            - GNU C Compiler / ARM Compiler Toolchain
  * Leveraged Code     -
  * URL                -
  */

#include "allocate.h"

uint32_t * allocate_words(size_t length)
{
    uint32_t * p = (uint32_t *)malloc(length);
    return p;
}
```

mem_test/allocate.h

```
/**
  * File Name         - allocate.h
  * Description     - header file for allocate.c
  * Author            - Atharva Nandanwar
  * Tools            - GNU C Compiler / ARM Compiler Toolchain
  * Leveraged Code     -
  * URL                -
  */

#ifndef MEM_TEST_ALLOCATE_H_
#define MEM_TEST_ALLOCATE_H_
#include <stdint.h>
#include <stdlib.h>
uint32_t * allocate_words(size_t length);
#endif /* MEM_TEST_ALLOCATE_H_ */
```

mem_test/display.c

```
/**
  * File Name         - display.c
  * Description     - returns a pointer which accesses bytes in the
  *                   allocated memory
  * Author            - Atharva Nandanwar
  * Tools            - GNU C Compiler / ARM Compiler Toolchain
  * Leveraged Code     -
  * URL                -
  */

#include "display.h"

uint8_t * display_memory(uint32_t *loc, size_t length)
{
    return (uint8_t*) loc;
}
```

mem_test/display.h

```
/**
  * File Name         - display.h
  * Description     - Header file for display.c
  * Author            - Atharva Nandanwar
  * Tools            - GNU C Compiler / ARM Compiler Toolchain
  * Leveraged Code     -
  * URL                -
  */

#ifndef MEM_TEST_DISPLAY_H_
#define MEM_TEST_DISPLAY_H_
#include <stdint.h>
#include <stdlib.h>
uint8_t * display_memory(uint32_t *loc, size_t length);
#endif /* MEM_TEST_DISPLAY_H_ */
```

mem_test/free.c

```
/**
```

```
  * File Name         - free.c
  * Description     - header file for free.c
  * Author            - Atharva Nandanwar
  * Tools            - GNU C Compiler / ARM Compiler Toolchain
  * Leveraged Code     -
  * URL                -
  */

#include "free.h"

void free_words(uint32_t* src)
{
    free(src);
}
```

mem_test/free.h

```
/**
  * File Name         - free.h
  * Description     - contains free_words functions which frees memory
  * Author            - Atharva Nandanwar
  * Tools            - GNU C Compiler / ARM Compiler Toolchain
  * Leveraged Code     -
  * URL                -
  */

#ifndef MEM_TEST_FREE_H_
#define MEM_TEST_FREE_H_
#include <stdint.h>
#include <stdlib.h>
void free_words(uint32_t* src);
#endif /* MEM_TEST_FREE_H_ */
```

mem_test/get_addr.c

```
/**
  * File Name         - get_addr.c
  * Description     - contains get_address function which gives address from 
  * Author            - Atharva Nandanwar
  * Tools            - GNU C Compiler / ARM Compiler Toolchain
  * Leveraged Code     -
  * URL                -
  */

#include "get_addr.h"

uint32_t* get_address(uint32_t* base, uint16_t offset)
{
    if(base == NULL)
    {
        return NULL;
    }
    else
    {
        // uint8 because we want to increment by one byte
        uint8_t* temp = (uint8_t*) base;
        base = (uint32_t*)(temp + offset);
        return base;
    }
}
```

mem_test/get_addr.h

```
/**
  * File Name         - get_addr.h
  * Description      - header file for get_addr.c
  * Author            - Atharva Nandanwar
  * Tools            - GNU C Compiler / ARM Compiler Toolchain
  * Leveraged Code     -
  * URL                -
  */

#ifndef MEM_TEST_GET_ADDR_H_
#define MEM_TEST_GET_ADDR_H_
#include <stdint.h>
#include <stdlib.h>
uint32_t* get_address(uint32_t* base, uint16_t offset);
#endif /* MEM_TEST_GET_ADDR_H_ */
```

mem_test/invert.c

```
/**
  * File Name         - invert.c
  * Description     - contains function which inverts a block of memory
  * Author            - Atharva Nandanwar
  * Tools            - GNU C Compiler / ARM Compiler Toolchain
  * Leveraged Code     -
  * URL                -
  */

#include "invert.h"

mem_status invert_block(uint32_t* loc, size_t length)
{
    if (loc == NULL)
    {
        return FAIL;
    }
    uint8_t* temp = (uint8_t*) loc;
    volatile uint8_t i;
    // Going byte by byte
    for (i = 0; i < length; i++)
    {
        // XOR to invert the memory
        *(temp + i) ^= 0xFF;
    }
    return SUCCESS;
}
```

mem_test/invert.h

```
/**
  * File Name         - invert.h
  * Description     - header file for invert.c
  * Author            - Atharva Nandanwar
  * Tools            - GNU C Compiler / ARM Compiler Toolchain
  * Leveraged Code     -
  * URL                -
  */

#ifndef MEM_TEST_INVERT_H_
#define MEM_TEST_INVERT_H_
#include <stdlib.h>
#include <stdint.h>
#include "../common.h"
mem_status invert_block(uint32_t* loc, size_t length);
#endif /* MEM_TEST_INVERT_H_ */
```

mem_test/mem_write.c

```
/**
  * File Name         - mem_write.c
  * Description     - contains function that writes individual bytes
  * Author            - Atharva Nandanwar
  * Tools           - GNU C Compiler / ARM Compiler Toolchain
  * Leveraged Code     -
  * URL                -
  */

#include "mem_write.h"

mem_status write_memory(uint32_t* loc, uint8_t value)
{
    if (loc == NULL)
    {
        return FAIL;
    }
    // Writing into the individual byte
    uint8_t * temp = (uint8_t *) loc;
    *temp = value;
    return SUCCESS;
}
```

mem_test/mem_write.h

```
/**
  * File Name         - mem_write.h
  * Description     - header file for mem_write.c
  * Author            - Atharva Nandanwar
  * Tools            - GNU C Compiler / ARM Compiler Toolchain
  * Leveraged Code     -
  * URL                -
  */

#ifndef MEM_TEST_MEM_WRITE_H_
#define MEM_TEST_MEM_WRITE_H_
#include <stdint.h>
#include <stdlib.h>
#include "../common.h"
mem_status write_memory(uint32_t* loc, uint8_t value);
#endif /* MEM_TEST_MEM_WRITE_H_ */
```

mem_test/pattern_write.c

```
/**
  * File Name         - pattern_write.c
  * Description     - contains function which writes pattern from
  *                   pattern generator into memory block
  * Author            - Atharva Nandanwar
  * Tools            - GNU C Compiler / ARM Compiler Toolchain
  * Leveraged Code     -
  * URL                -
  */

#include "pattern_write.h"

mem_status write_pattern(uint32_t * loc, size_t length, int8_t seed)
{
    if(loc == NULL)
    {
        return FAIL;
    }
    uint8_t* byte_array = (uint8_t *) loc;
    pattern_generator(byte_array, length, seed);
    return SUCCESS;
}
```

mem_test/pattern_write.h

```
/**
  * File Name         - pattern_write.h
  * Description     - header file for pattern_write.c
  * Author            - Atharva Nandanwar
  * Tools            - GNU C Compiler / ARM Compiler Toolchain
  * Leveraged Code     -
  * URL                -
  */

#ifndef MEM_TEST_PATTERN_WRITE_H_
#define MEM_TEST_PATTERN_WRITE_H_
#include <stdint.h>
#include <stdlib.h>
#include "../common.h"
#include "pattern_gen/pattern_gen.h"
mem_status write_pattern(uint32_t * loc, size_t length, int8_t seed);
#endif /* MEM_TEST_PATTERN_WRITE_H_ */
```

mem_test/verify.c

```
/**
  * File Name         - verify.c
  * Description     - contains function which verifies pattern and
  *                   return defunct addresses
  * Author            - Atharva Nandanwar
  * Tools            - GNU C Compiler / ARM Compiler Toolchain
  * Leveraged Code     -
  * URL                -
  */
#include "verify.h"

ARCH_SIZE * verify_pattern(uint32_t * loc, size_t length, int8_t seed)
{
    // For byte wise operations
```

```
        uint8_t* temp = (uint8_t*) loc;
        // global array to hold defunct addresses
        extern ARCH_SIZE buffer_address[16];
        // local array to hold pattern generator values
        uint8_t pattern_holder[length];
        pattern_generator(pattern_holder, length, seed);
        // i for looping through the length of pattern,
        // j for storing defunct addresses if any
        if(loc != NULL)
        {
            volatile uint8_t i, j = 0;
            for (i = 0; i < length; i++)
            {
                // If pattern matches
                if (*(temp + i) == pattern_holder[i])
                {
                    continue;
                }
                // If pattern doesn't match
                else if (*(temp + i) != pattern_holder[i])
                {
                    *(buffer_address + j) = (ARCH_SIZE) (temp + i);
                    j++;
                }
            }
            // If verify pattern sucessful, empty buffer for extra
            // precautions
            if(j == 0)
            {
                for (i = 0; i < length; i++)
                {
                    buffer_address[i] = 0;
                }
            }
            return buffer_address;
        }
        else
        {
            return NULL;
        }
    }
```

mem_test/verify.h

```c
/**
  * File Name         - verify.h
  * Description     - header file for verify.c
  * Author            - Atharva Nandanwar
  * Tools            - GNU C Compiler / ARM Compiler Toolchain
  * Leveraged Code     -
  * URL                -
  */

#ifndef MEM_TEST_VERIFY_H_
#define MEM_TEST_VERIFY_H_
#include <stdint.h>
#include <stdlib.h>
#include "pattern_gen/pattern_gen.h"
#include "common.h"
extern uint8_t length;
ARCH_SIZE * verify_pattern(uint32_t * loc, size_t length, int8_t seed);
#endif /* MEM_TEST_VERIFY_H_ */
```

pattern_gen/pattern_gen.c

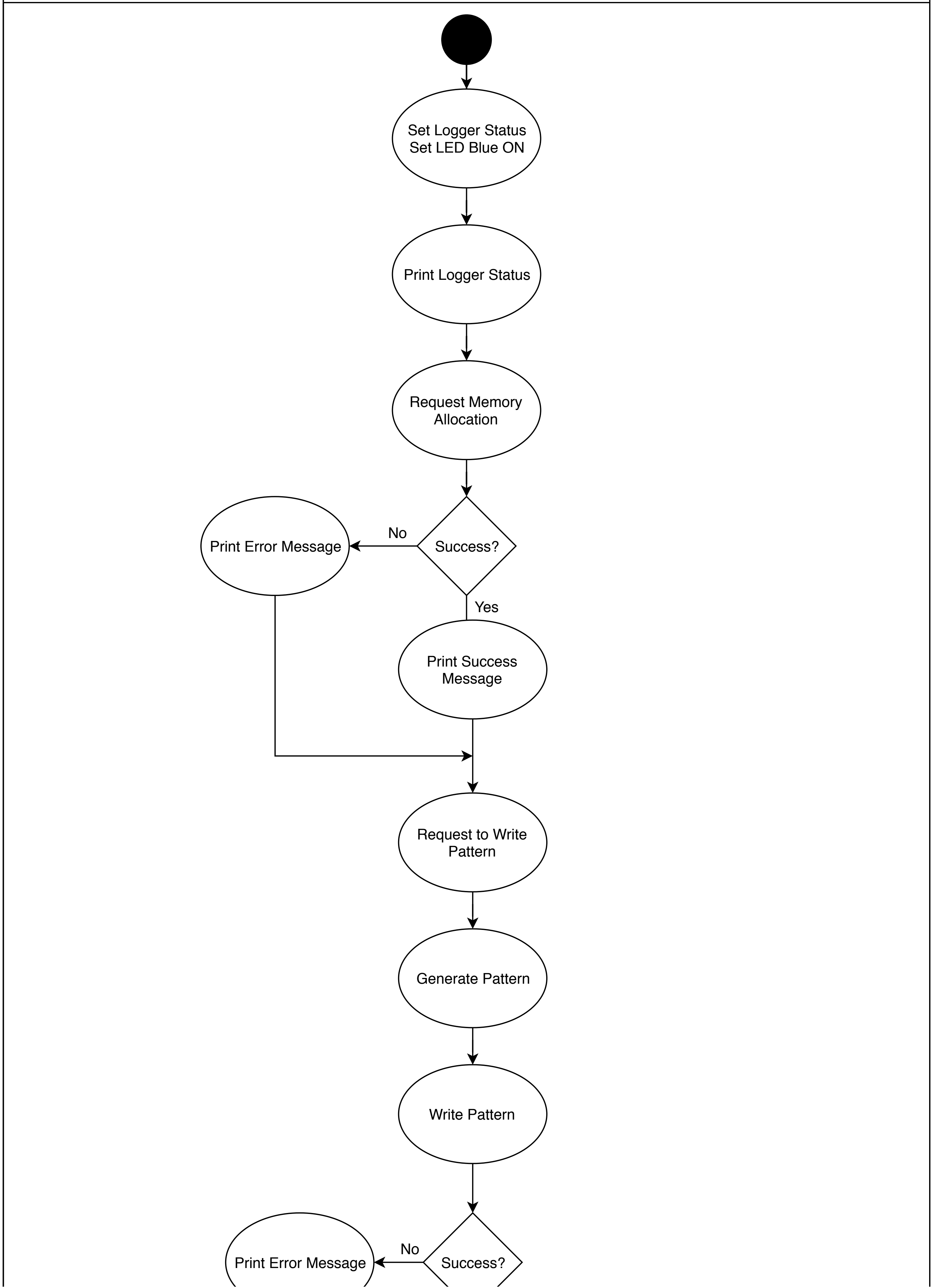
```c
/**
  * File Name         - pattern_gen.c
  * Description     - contains function generating pattern from a seed
  * Author            - Atharva Nandanwar
  * Tools            - GNU C Compiler / ARM Compiler Toolchain
  * Leveraged Code     -
  * URL                -
  */


#include "pattern_gen.h"

void pattern_generator(uint8_t *pattern, uint8_t length, int8_t seed)
{
    volatile uint8_t i, j;
    // Random lookup table for calculations
    uint8_t lookup[15] = {17, 2, 32, 66, 1, 99, 30, 23, 53, 6, 14, 67, 59, 89,
    uint8_t temporary[length];
    for (i = 0, j = 0; i < length; i++, j++)
    {
        // Random function to calculate random values
        temporary[i] = seed * seed + lookup[j] + (i % 13);
        // Lookup table operated circularly
        if (j == 14)
        {
            j = 0;
        }
    }
    for (i = 0; i < length; i++)
    {
        *(pattern + i) = temporary[i];
    }
}
```

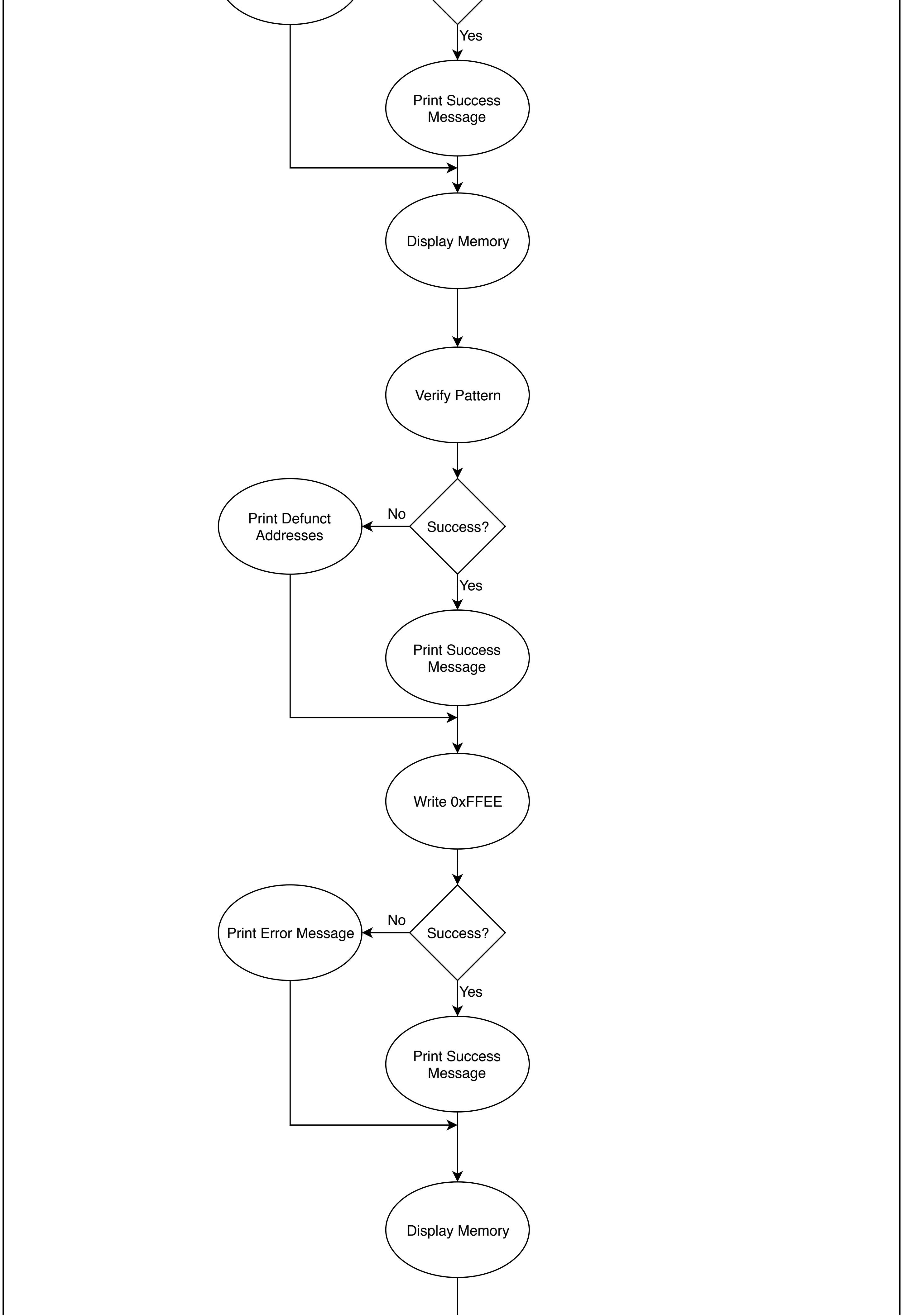
pattern_gen/pattern_gen.h

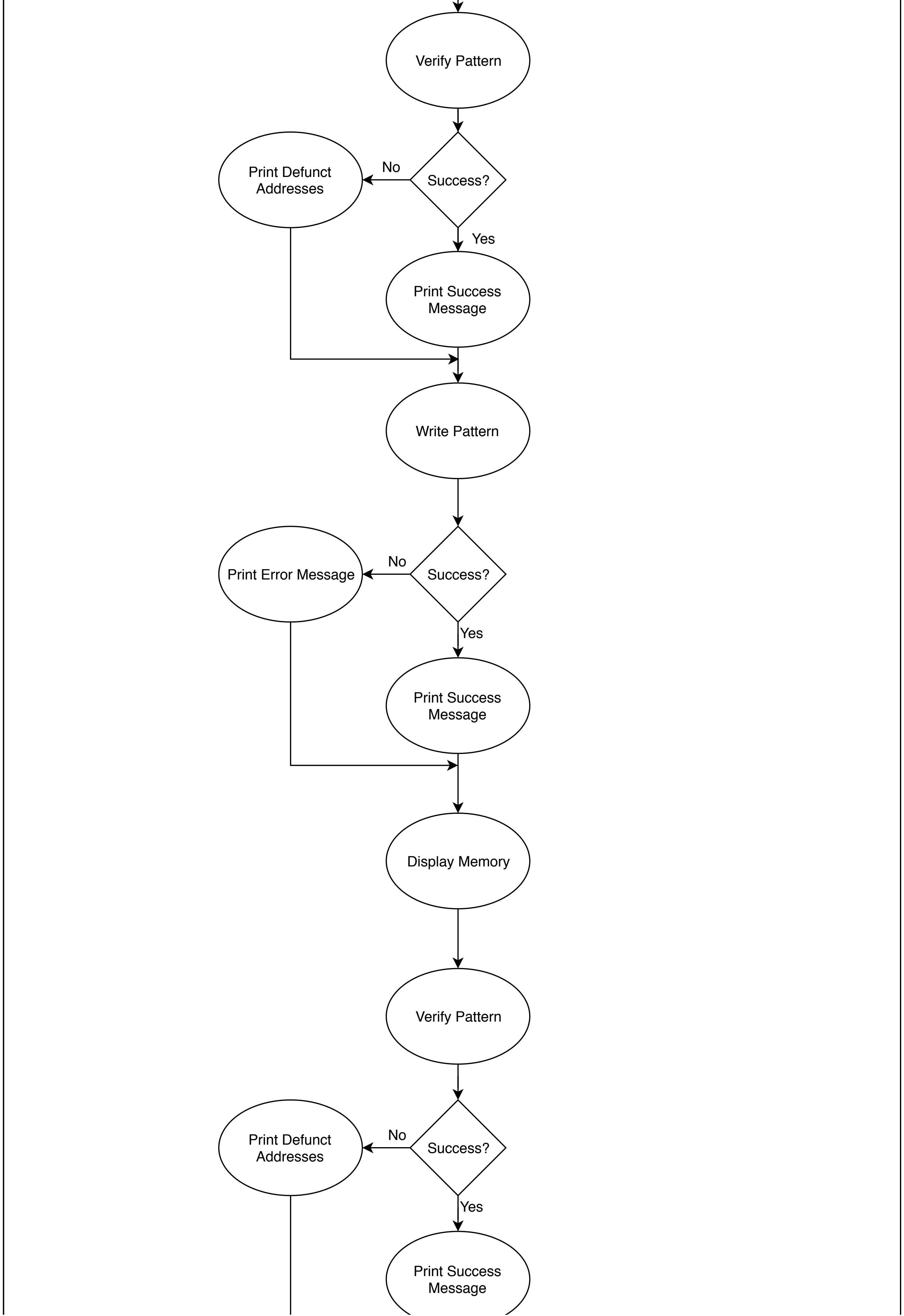
```
/**
  * File Name         - pattern_gen.h
  * Description     - header file for pattern_gen.c
  * Author            - Atharva Nandanwar
  * Tools            - GNU C Compiler / ARM Compiler Toolchain
  * Leveraged Code     -
  * URL                -
  */


#ifndef PATTERN_GEN_H_
#define PATTERN_GEN_H_
#include <stdint.h>
void pattern_generator(uint8_t *pattern, uint8_t length, int8_t seed);
#endif /* PATTERN_GEN_H_ */
```

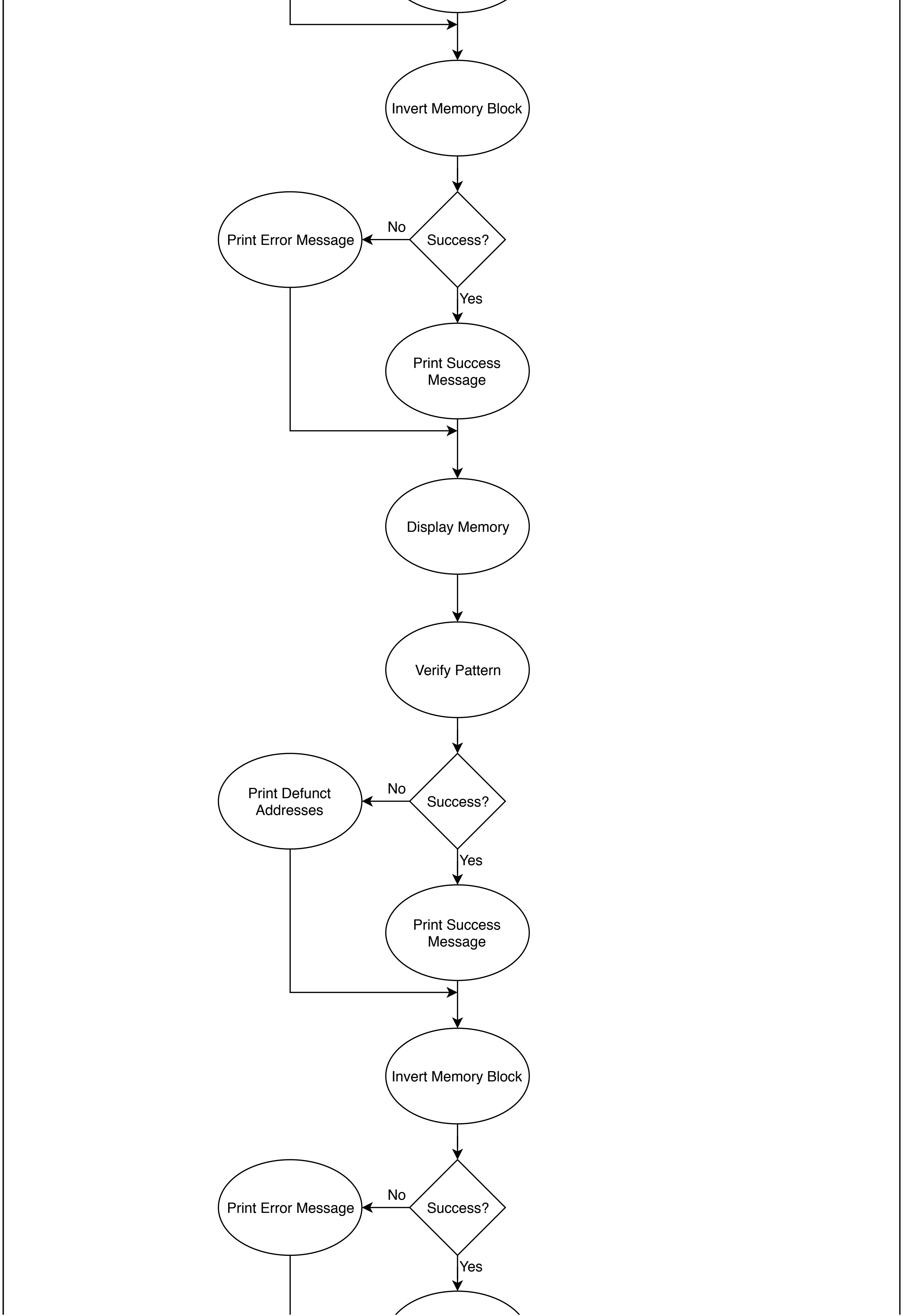
Principles of Embedded Software Project 3 Activity Diagram

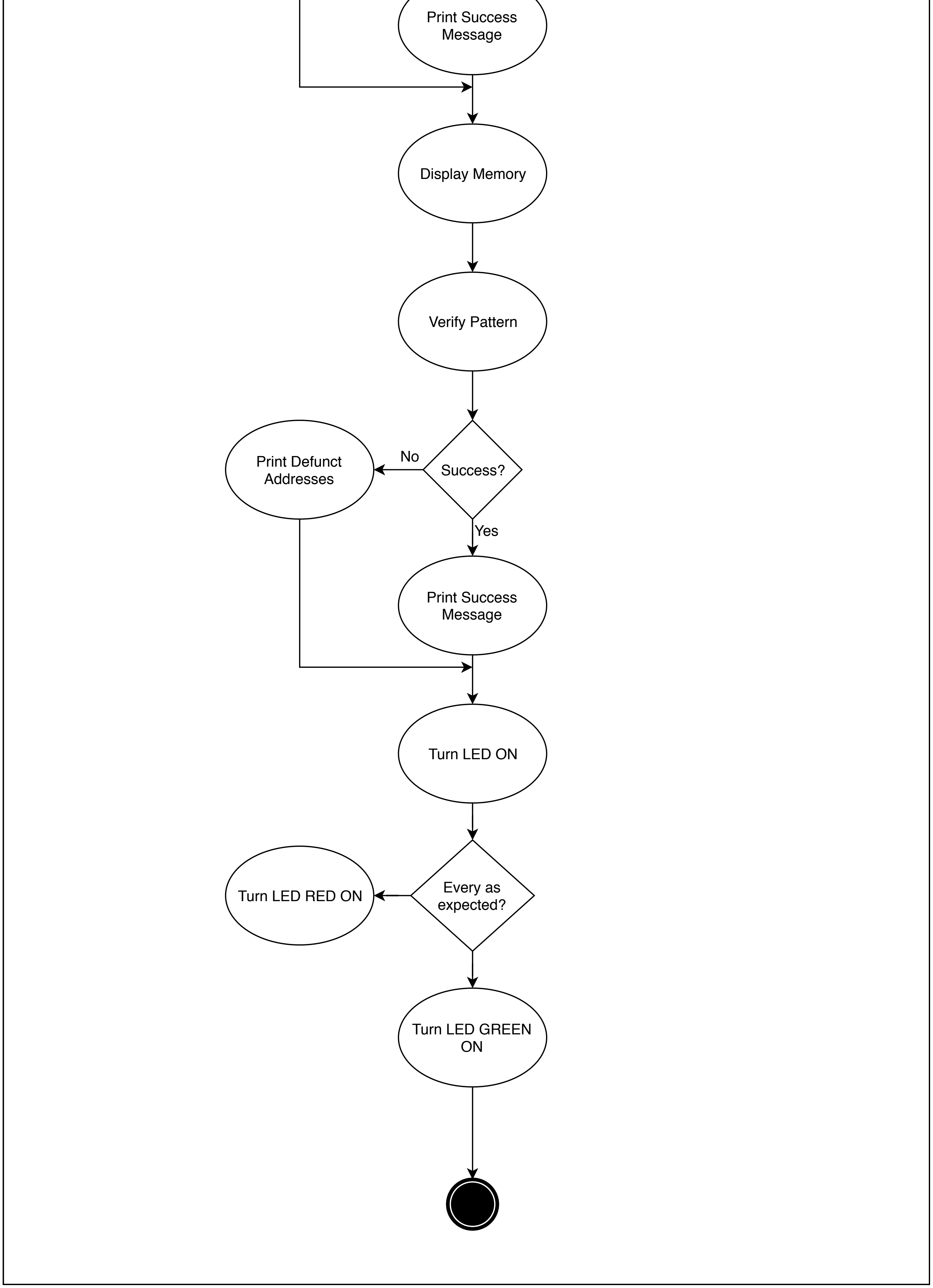
MAIN PROGRAM FLOW



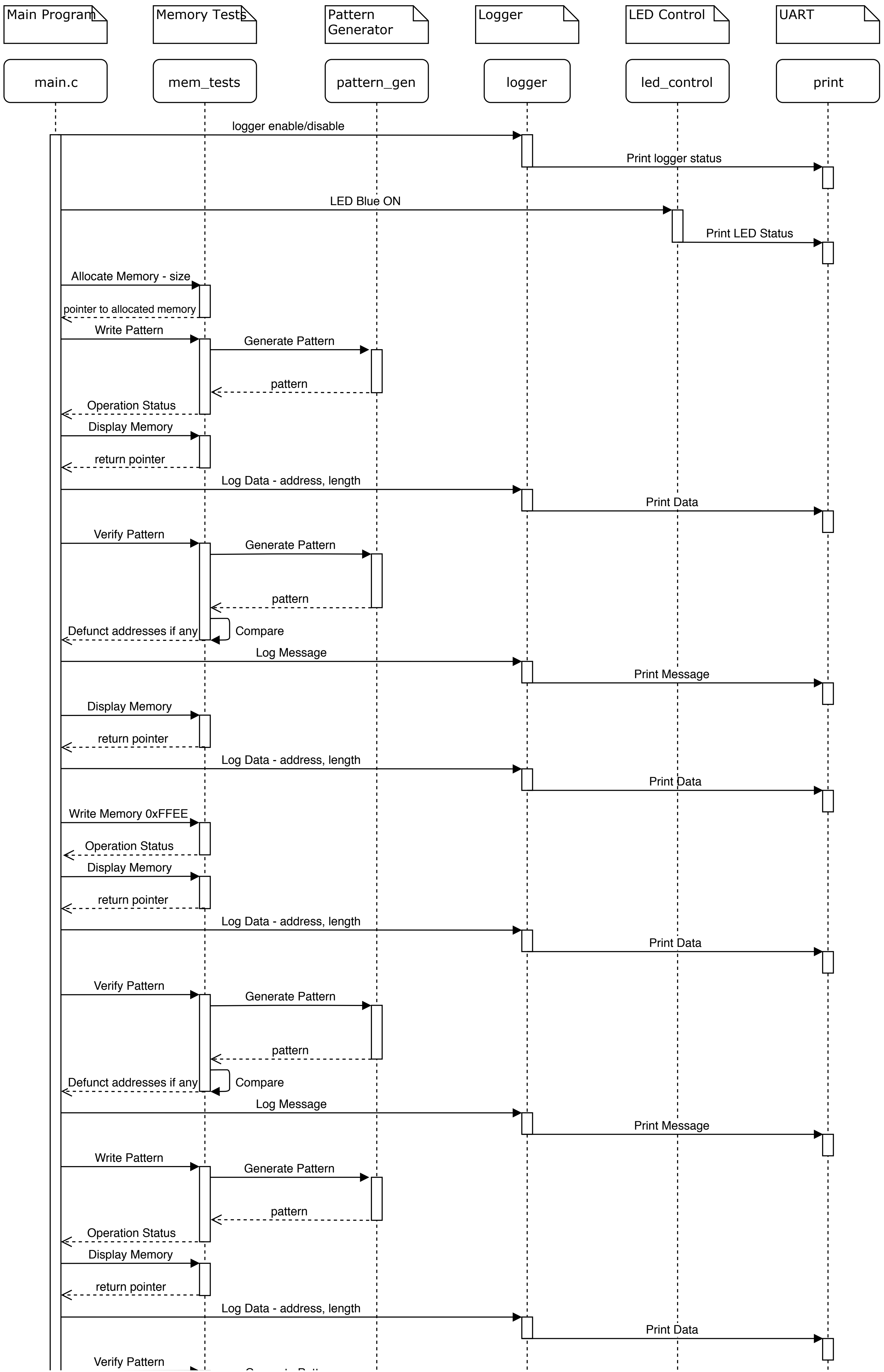

Yes
Print Success Message
Display Memory
Verify Pattern
Success?
No
Print Defunct Addresses
Yes
Print Success Message
Write 0xFFEE
Success?
No
Print Error Message
Yes
Print Success Message
Display Memory

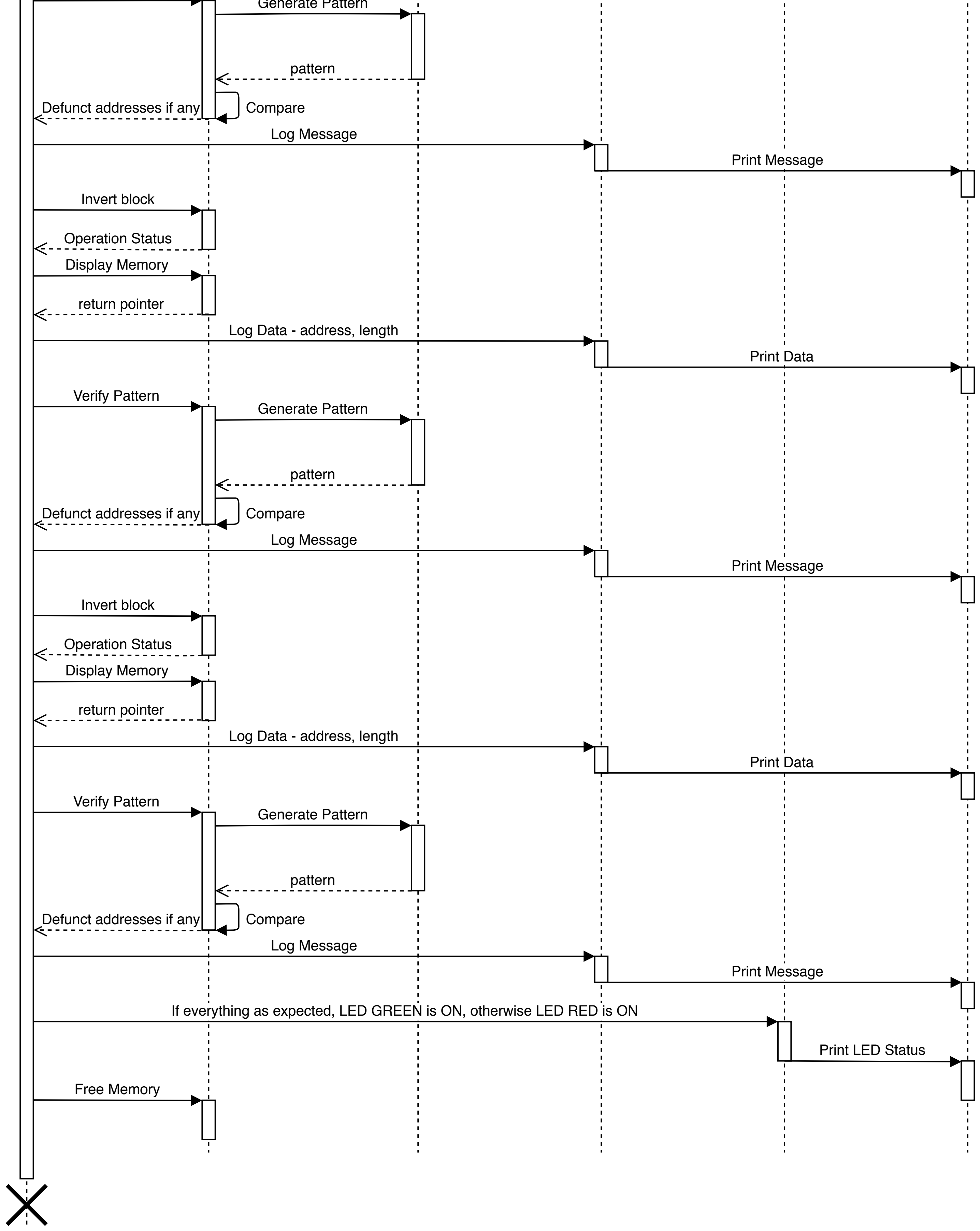

Verify Pattern
Success?
No
Print Defunct Addresses
Yes
Print Success Message
Write Pattern
Success?
No
Print Error Message
Yes
Print Success Message
Display Memory
Verify Pattern
Success?
No
Print Defunct Addresses
Yes
Print Success Message


Invert Memory Block
Success?
No
Print Error Message
Yes
Print Success Message
Display Memory
Verify Pattern
Success?
No
Print Defunct Addresses
Yes
Print Success Message
Invert Memory Block
Success?
No
Print Error Message
Yes


Print Success Message
Display Memory
Verify Pattern
Success?
No
Print Defunct Addresses
Yes
Print Success Message
Turn LED ON
Every as expected?
Turn LED RED ON
Turn LED GREEN ON

Principles of Embedded Systems Project 3 Sequence Diagram




Generate Pattern
pattern
Compare
Defunct addresses if any
Log Message
Print Message
Invert block
Operation Status
Display Memory
return pointer
Log Data - address, length
Print Data
Verify Pattern
Generate Pattern
pattern
Compare
Defunct addresses if any
Log Message
Print Message
Invert block
Operation Status
Display Memory
return pointer
Log Data - address, length
Print Data
Verify Pattern
Generate Pattern
pattern
Compare
Defunct addresses if any
Log Message
Print Message
If everything as expected, LED GREEN is ON, otherwise LED RED is ON
Print LED Status
Free Memory