Readme For Project 3

Team Members :

1) Atharv Desai (atharv.desai@colorado.edu)
2) Suraj Thite (suraj.thite@colorado.edu)

This is a readme file for the third project assignment in the Principles of Embedded Software Course for FALL '19.

The below enumerated files are contained in the repository

1. PES_PROJECT3
   - i) Project Settings
   - ii) Build Targets
   - iii) Binaries
   - iv) Includes
   - v) CMSIS
   - vi) Board
   - vii) Drivers
   - viii) Source
     - a) main.c & main.h
     - b) logger.c & logger.h
     - c) RGBled.c & RGBled.h
     - d) memory.c & memory.h
     - e) unittest.c & unittest.h
   - ix) Startup
   - x) Utilities
   - xi) Debug
   - xii) PESProject3 PE Debug.launch
2. UML Activity Diagram
3. UML Sequence Diagram
4. Readme File (Readme Markdown File)
5. WBS for project 3

INSTALLATION & EXECUTION NOTES

The code is tested on the enviornment below:

1. MCUXpresso IDE which is an easy to use Eclipse-based development environment for NXP® MCUs based on Arm® Cortex®-M cores. Fb_debug and fb_run modules were compiled and executed on this IDE.

2. MCUXpresso IDE which is an easy to use Eclipse-based development environment for NXP® MCUs based on Arm® Cortex®-M cores. Fb_debug and fb_run modules were compiled and executed on this IDE.

3. During this project, this IDE was used to code and execute memory tests on FRDM-KL25z and print their output on the IDE's serial terminal.

4. Also, we were able to check and verify the memory blocks allocation in heap using the memory map options in the IDE.

5. TO execute the project on both, the development system and linux/windows ,we developed a MakeFile for the code of the whole project and built on MCUXpresso IDE with all build options and flags mandatory to execute and build the project on development system as well as linux.

6. The hardware used in this project was FRDM-KL25Z board which has been designed by NXP and built on ARM® Cortex™-M0+ processor.

7. The editor used to build the code is gedit version 2.3.8.1 on Linux Mint Machine.

8. To execute the executable file simply type ./(filename) in linux gcc environment while click on debug (bug icon) and then resume button to execute the file on MCUXpresso.

9. To compile and create executable file in linux environment , type gcc (filename).c -o (filename) -Wall - Werror -lm

10. Kindly use notepad++ for viewing .out files ,particularly for first output since they have been misaligned due to character "Space or Tab" encoding.

11. Using Drawio online tool, we made UML sequence diagram and UML activity diagram for demonstrating the flow of memory test cases.

```c
/*
 * logger.c
 *
 *  Created on: Oct 20, 2019
 *      Author: SURAJ THITE ,Atharv Desai
 */


#include <stdio.h>
#include <stdlib.h>
#include <stdint.h>
#include "memory.h"
#include "RGBled.h"

uint8_t flag;
////// Logger for integer //////////

/*******************************************************************************************
 * Function Name: Log_integer(uint32_t intval)
 * Description : This function prints the integer value to the serial terminal
 * @input: integer to be printed
 * @Return : void
 *******************************************************************************************/
void Log_integer(uint32_t intval)
{
	if(flag==1)	//if logging enabled
		printf("%d \n",intval); // Print the  data
}


/*******************************************************************************************
 * Function Name: Log_string(char* str)
 * Description : This function prints the string pointed by the input argument
 * @input:  pointer from which string  to be printed
 * @Return : void
 *******************************************************************************************/
////// Logger for string //////////
void Log_string(char* str)
{
	if(flag==1)			//Check
		printf("\n \r%s",str);	//Print the string to the terminal

}


/*******************************************************************************************
 * Function Name: Log_data(uint32_t * loc,uint32_t length)
 * Description : This function  is used to log the address and variable value stored at it.
 * @input:  pointer to the memory location and the length
 * @Return : void
 *******************************************************************************************/
////// Logger for data //////////
void Log_data(uint32_t * loc,uint32_t length)
{
	if(flag==1) {
		int i=0;
		printf("\n\rThe Address is %p  ",loc);	//Print the address of the memory block pointed
by the pointer
		if (length !=0)
		{
			printf(" and data is");
		}
		for(i=0;i<length;i++)
		{
			printf(" %x",*(loc+i));	//Print the data stored at the memory location
		}

	}
}
```

```
////// Logger status, enable and Disable ///////////

/*********************************************************************************************
 * Function Name:void Log_enable()
 * Description : This function .enables the logging for the system
 * @input:  void
 * @Return : void
 *********************************************************************************************/
////// Logger for data ///////////
void Log_enable()
{
	flag =1;
}

/*********************************************************************************************
 * Function Name: Log_disable()
 * Description : This function .disables the logging for the system
 * @input:  void
 * @Return : void
 *********************************************************************************************/
////// Logger for data ///////////
void Log_disable()
{
	flag =0;
}

/*********************************************************************************************
 * Function Name:uint8_t Log_status()
 * Description : This function  enables whether log is enabled or disabled for the system
 * @input:  void
 * @Return : flag of uint8_t
 *********************************************************************************************/
////// Logger for data ///////////
uint8_t Log_status()
{
	return flag;
}
```

```c
/*
 * logger.h
 *
 *  Created on: Oct 20, 2019
 *      Author: SURAJ THITE ,Atharv Desai
 */



#ifndef LOGGER_H_
#define LOGGER_H_

void Log_enable();
void Log_disable();
uint8_t Log_status();
void Log_data (uint32_t *, uint32_t );
void Log_string(char* str);
void Log_integer(uint32_t);

#endif
```

```c
/*
 * main.c
 *
 *  Created on: Oct 20, 2019
 *      Author: SURAJ THITE ,Atharv Desai
 */


#include "main.h"
#include "RGBled.h"
#include "unitTest.h"


/*******************************************************************************************************
 * Function Name:main (void)
 * Description : this is the main function
 * @input:void
 * @Return : 0
 *******************************************************************************************************/
int main(void)
{
        Log_enable(); // Enable Logging
        /* Init board hardware. */
        BOARD_InitBootPins();
        BOARD_InitBootClocks();
        BOARD_InitBootPeripherals();
        /* Init FSL debug console. */
        BOARD_InitDebugConsole();
        //Code Start

        // Initialize the pointers, length and seed value
        uint32_t  *ptr1;
        uint32_t  *ptr2 ;


        //Initialize the LED and Set them to off state
        RGB_init();
        RGB_OFF();

        //Turn the Blue LED on
        led_switch(0);

        //Allocate the 16 bytes of memory and take store the returning pointer of the starting memory
location
        ptr1 =  allocate_words(length);


        //Turn the Blue LED on
        led_switch(0);

        //Initialize the enum to store the response

        //Display the log status

        uint8_t a = Log_status();
        printf(" \n \r Log Status Value");
        printf("\n \r %d",a);


        mem_status r1;

        //Write pattern with length 4 and seed value 5. Response is stored in enum declared.
        r1 = write_pattern(ptr1,length,SEED);

        //Check for response
        if (r1==SUCCESS)  Log_string(" \n \r RESPONSE : SUCCESS");
        else Log_string("\n \r RESPONSE : FAILED");
```

```
//Display the contents of memory location pointed by ptr1.
display_mem(ptr1,length);

//Turn the Blue LED on
led_switch(0);
//Initialize the pointer to store the result of verify pattern
ptr2=NULL;
// Verify the pattern with pattern generated at ptr1 location
ptr2 = verify_pattern(ptr1,length,SEED);

//Print the status
if(ptr2 != NULL)
{
        Log_string("\n \r Different value found at location ");
        Log_data(ptr2,0);
        //Change the LED color to RED
        led_switch(1);
}
else
        {
        Log_string("\n \r Passed!");

        //Change the LED Color to Green
        led_switch(2);
        }

//Turn the Blue LED on
led_switch(0);


//Initialize the enum to store the response
mem_status res1;


//Write to memory address with OFFSET 1 with value FFEE and store its response in res1
res1 = write_memory(get_address(1), 0xFFEE);

//Check for the response
if (res1==SUCCESS)  Log_string(" \n \r RESPONSE : SUCCESS");
else Log_string("\n \r RESPONSE : FAILED");


//Display the contents of the memory location
display_mem(ptr1,length);


//Turn the Blue LED on
led_switch(0);

//Initialize the pointer to store the result of verify pattern
ptr2=NULL;

// Verify the pattern with pattern generated at ptr1 location
ptr2 = verify_pattern(ptr1,length,SEED);

//Print the status
if(ptr2 != NULL)
{
        Log_string("\n \r Different value found at location ");
        Log_data(ptr2,0);
  //Change the LED color to RED
        led_switch(1);
}
else
{
        Log_string("\n \r Passed!");
        //Change the LED Color to Green
        led_switch(2);
}
```

```
//Display the COntents of Memory Location
display_mem(ptr1,length);


//Turn the Blue LED on
led_switch(0);

//Initialize the enum to store the response
mem_status r2;
//Write the pattern to the location pointed by ptr1 with SEED Value 5
r2=write_pattern(ptr1,length,SEED);

//Print the response
if (r2==SUCCESS)  Log_string(" \n \r RESPONSE : SUCCESS");
else Log_string("\n \r RESPONSE : FAILED");



//Display the contents of memory location pointed by ptr1
display_mem(ptr1,length);


//Turn the Blue LED on
led_switch(0);

//Initialize the pointer to store the result of verify pattern
ptr2=NULL;

// Verify the pattern with pattern generated at ptr1 location
ptr2 = verify_pattern(get_address(0),length,SEED);

//Print the response
if(ptr2 != NULL)
{
        Log_string("\n \r Different value found at location ");
        Log_data(ptr2,0);

        //Change the LED color to RED
        led_switch(1);
}
else
{
        Log_string("\n \r Passed!");
        //Change the LED color to GREEN
        led_switch(2);
}

//Turn the Blue LED on
led_switch(0);

//INvert the contents of the zero location with respect to first memory of the block
mem_status res2;
res2 = invert_block(get_address(0),1);

//Print the response
if (res2==SUCCESS)  Log_string(" \n \r RESPONSE : SUCCESS");
else Log_string("\n \r RESPONSE : FAILED");


//Display the contents of the memory location
display_mem(ptr1,length);


//Turn the Blue LED on
led_switch(0);

//Initialize the pointer
```

```
ptr2=NULL;

//Verify the pattern
ptr2 = verify_pattern(ptr1,length,SEED);

//Print the response
if(ptr2 != NULL)
{
        Log_string("\n \r Different value found at location ");
        Log_data(ptr2,0);
        //Change the LED color to RED
        led_switch(1);
}
else
{
Log_string("\n \r Passed!");
//Change the LED Color to Green
led_switch(2);

}


//Turn the Blue LED on
led_switch(0);

//Invert the same location which was inverted before and log its response
mem_status res3;
res3 = invert_block(get_address(0),1);

//Check for return data
if (res3==SUCCESS)  Log_string(" \n \r RESPONSE : SUCCESS");
else Log_string("\n \r RESPONSE : FAILED");


//Display the contents of the memory location pointed by ptr1
display_mem(ptr1,length);



//Turn the Blue LED on
led_switch(0);

//Initialize the pointer to null and check for return type for the address of mismatching locaiton
ptr2=NULL;

//Verify the Pattern
ptr2 = verify_pattern(ptr1,length,SEED);
if(ptr2 != NULL)
{
        Log_string("\n \r Different value found at location ");
        Log_data(ptr2,0);
        //Change the Color of the LED to red
        led_switch(1);
}
else
{
        Log_string("\n \r Passed!");
        //Change the color of the LED to green
        led_switch(2);

}

//Free the memory pointed by ptr1
free_mem(ptr1);

//Display the COntents of the memory location
display_mem(ptr1,length);
```

```c
	//retun 0 to the funcion calling main
	RGB_OFF();
	printf("\n \r Running Test Cases");
	Testsuite_RunTests();
	return 0;
}


/*********************************************************************************************************
 * Function Name:int delay(int time_ms)
 * Description : this function provides delay in milliseconds according to input parameters
 * @input:time in milliseconds
 * @Return : NULL
 *********************************************************************************************************/

void delay(int time_ms)
{
	volatile uint32_t i = 0;
	for (i = 0; i < 2400*time_ms; ++i)
	{
		__asm("NOP"); /* No operation */
	}
}
```

```c
/*
 * main.h
 *
 *  Created on: Oct 20, 2019
 *      Author: SURAJ THITE , Atharv Desai
 */

#ifndef MAIN_H_
#define MAIN_H_

//Include Files
#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 "memory.h"
#include "pattern.h"
#include "logger.h"

//Function to create the delay for LEDS to observe the State
void delay(int time_ms);

# define SEED 5
# define length 4

#endif /* MAIN_H_ */
```

```
/*
 * memory.c
 *
 *  Created on: Oct 20, 2019
 *      Author: SURAJ THITE
 */

#include <stdio.h>
#include <stdlib.h>
#include <stdint.h>
#include "memory.h"
#include "RGBled.h"
#include "pattern.h"


uint32_t  *ptr; // GLobal Pointer to store the starting address of the memory block
uint32_t *last_ptr;  // GLobal Pointer to store the ending address of the memory block
int i;

/**********************************************************************************************
 * Function Name:uint32_t * allocate_words(size_t length)
 * Description : this function allocates memory of type uint32_t of specified length(nos).
 * @input:Length
 * @Return : pointer to first location of the memory block
 **********************************************************************************************/

uint32_t * allocate_words(size_t length)
{

	i=4*length;

	ptr= malloc(i);          //Allocate memory

	last_ptr=ptr + length;   //Store the address of the last memory block


	/*check the pointer to see whether the memory block can be allocated or not*/
	if (ptr==NULL)
	{
		Log_string("\n \r Can't allocate memory\n");
		led_switch(1);
		exit(0);
	}

	Log_string("\n \r Memory Allocated and starting memory is :"); //Print the starting address of the
memory block
	Log_data(ptr, 0);
	Log_string("\n \r Memory Allocated and ending memory is :"); // Print the ending address of the
memory block
	Log_data(last_ptr, 0);
	led_switch(2);
	return ptr;
}

/**********************************************************************************************
 * Function Name:void free_mem(uint32_t *ptr1)
 * Description : this function frees  memory  block pointed by ptr1.
 * @input:pointer to memory block *ptr1
 * @Return :void
 **********************************************************************************************/
void free_mem(uint32_t *ptr1)
{
	if(ptr1 != NULL) // Check if null pointer is passed
	{
		free(ptr1);
		Log_string("\n \r Memory block cleared for the pointer");// Memory block cleared
		led_switch(2);   //Change the LED color to Green
	}
	else
```

```
        {
                Log_string("\n \r Warning: trying to clear free memory");// Handle error
                led_switch(1);  //Change the LED color to RED

        }
}

/*********************************************************************************************
 * Function Name:uint32_t* display_mem(uint32_t * loc, size_t length)
 * Description : This function displays the contents of memory location pointed by loc of specified length
 * @input:pointer to memory block *ptr1 and length
 * @Return :uint32_t type of pointer
 *********************************************************************************************/
//////////////// To display a Memory block /////////////
uint32_t* display_mem(uint32_t * loc, size_t length)
{
        Log_data(loc, length);
//      int j=0;
//      printf(" \n \r Data at address %p of ptr \n",loc);
//      for (j=0;j<length;j++)
//      {
//              printf("%X  ",*loc);    //Print he data at the memory location
//              loc++;  //Increment the pointer
//      }
        return loc;
}
/*********************************************************************************************
 * Function Name:mem_status write_memory(uint32_t * loc, uint16_t value)
 * Description : This function writes the value to the memory pointed by the passed pointer.
 * @input:pointer to memory block *ptr1 and vlaue to be stored
 * @Return :enum of type mem_status
 *********************************************************************************************/
//////////////// To write to a memory block /////////////
mem_status write_memory(uint32_t * loc, uint16_t value)
{
        mem_status a;   // Create enum to return
        if (loc <= last_ptr && loc >= ptr)      //Memory bound checking
        {
                a=SUCCESS;
                *loc = value;   //Write data to address pointed by loc
                //printf("\n \r You wrote Data:%d at Memory Location:%p",value,loc); // Print the value
and address of the location
                Log_data(loc, 1);
                led_switch(2);  //Change the LED color to Green
                return a;       //Return Success

        }
        else
        {
                a=FAILED;
                led_switch(1);  //Change the LED color to Red
                return a;       //Return Failed

        }
}


/*********************************************************************************************
 * Function Name:mem_status invert_block(uint32_t * loc, size_t length)
 * Description : This function inverts the vlaue stored the memory block at location pointed by loc
pointer
 * @input:pointer to the memory location and length
 * @Return :enum of type mem_status
 *********************************************************************************************/
//////////////// Using XOR to invert a block ////////////

mem_status invert_block(uint32_t * loc, size_t length)
{
        mem_status a;
	if (loc <= last_ptr && loc >= ptr)	//memory bound checking
	{
		a=SUCCESS;
		int i=0;
		for(i=0;i<length; i++)
		{
			*(loc +i) ^= 0xFFFFFFFF; //Invert the contents pointed by location pointer by
implementing xor logic
		}
		led_switch(2);	//Switch LED color to Green
		return a;	//Return SUCCESS
	}
	else
	{
		a=FAILED;
		led_switch(1);	//Switch LED color to RED
		return a;	//Return Failure
	}
}


/**********************************************************************************************
 * Function Name:mem_status write_pattern(uint32_t * loc, size_t length, uint8_t seed)
 * Description : This function writes the generated pattern at location pointed by the pointer
 * @input:pointer to the memory location and length and seed value
 * @Return :enum of type mem_status
 **********************************************************************************************/

///////////////// To pass the data to Pattern Generator //////////
mem_status write_pattern(uint32_t * loc, size_t length, uint8_t seed)
{
	mem_status a;
	if (loc <= last_ptr && loc >= ptr)	//Memory Bound Checking
	{
		a=SUCCESS;
		uint32_t *temp1 =loc;
		gen_pattern(temp1,length,seed); //Call the pattern generator function
		led_switch(2); //Change the Led to Green
		return a;	//Return Success
	}
	else
	{
		a=FAILED;
		led_switch(1);	//Change the LED color to RED
		return a;	//Return Failed
	}
}

/**********************************************************************************************
 * Function Name:uint32_t * verify_pattern(uint32_t * loc, size_t length, int8_t seed)
 * Description : This function verifies the generated pattern at location pointed by the pointer
 * @input:pointer to the memory location and length and seed value
 * @Return :pointer of uint32_t type which returns the address of different value
 **********************************************************************************************/
uint32_t * verify_pattern(uint32_t * loc, size_t length, int8_t seed)
{
	uint32_t  *temp1 = NULL;
	uint32_t ptr2[length];	//Define array to store the pattern
	uint32_t *p = loc;	//store location address to temp pointer
	mem_status_write_pattern_array (ptr2,length,5);  //Write pattern at different location
	int i =0;
	int flag1=0;	//Set Flag to 0
	//temp2 = ptr2;
	while (i<length)
	{
		if(*loc == *(ptr2+i))		//Check if both the values ate equal
		{
			Log_string(" \n \r Same");
			//flag1 =0;
```

```
                }
                else
                {
                        flag1 = 1;
                        Log_string("\n \r Different value");     //Check if both the value different
                        //return ptr2;
                        temp1 = loc;     //Store the location into temporary pointer
                        p=loc;

                }
                loc ++;
                //              p++ ;
                i++;
        }

        if ( flag1 ==1 ) return p;       //Return address if different values found
        if ( flag1 ==0) return NULL;     //Return NULL if all values are same.
        //      printf("\n \r Memory Freed which was used for verification");
}

/*********************************************************************************************************
 * Function Name:uint32_t * get_address(uint32_t  offset)
 * Description : This function returns the absolute address wrt input offset.
 * @input:pointer to the memory location.
 * @Return :pointer of uint32_t type which returns the absolute  address of the memory location
 *********************************************************************************************************/

uint32_t * get_address(uint32_t  offset)
{
        uint32_t *temp;
        temp = ptr + offset;     //Calculate the offset address
        return (temp);   //return the absolute address
}
```

```
/*
 * memory.h
 *
 *  Created on: Oct 20, 2019
 *      Author: SURAJ THITE
 */

#ifndef MEMORY_H_
#define MEMORY_H_

typedef enum mem_status
{
SUCCESS = 0, // no error
FAILED // failure case
} mem_status;

#include "logger.h"
// Functions for memory tests //

mem_status write_pattern(uint32_t * loc, size_t length, uint8_t seed);
mem_status invert_block(uint32_t * loc, size_t length);
mem_status write_memory(uint32_t * loc, uint16_t value);
uint32_t* display_mem(uint32_t * loc, size_t length);
void free_mem(uint32_t *ptr1);
uint32_t * allocate_words(size_t length);
uint32_t * verify_pattern(uint32_t * loc, size_t length, int8_t seed);
uint32_t * get_address(uint32_t  offset);


#endif /* MEMORY_H_ */
```

```
/*
 * pattern.c
 *
 *  Created on: Oct 21, 2019
 *      Author: SURAJ THITE
 */

#include "pattern.h"

/*****************************************************************************************************
 * Function Name:void mem_status_write_pattern_array(uint32_t * loc, size_t length, uint8_t seed)
 * Description : this function writes the generated pattern for verification purposes.
 * @input: pointer to location , Length , Seed value
 * @Return : Void
 *****************************************************************************************************/
void mem_status_write_pattern_array(uint32_t * loc, size_t length, uint8_t seed)
{
        uint32_t *temp1 =loc;
        gen_pattern(temp1,length,seed); // Generate pattern at different location for verification
purposes

}

/*****************************************************************************************************
 * Function Name:void gen_pattern(uint32_t * pattern, size_t length, int8_t seed)
 * Description : this function generates a pattern w.r.t. seed value and specified length.
 *                               this pattern algorithm  is referenced on
https://stackoverflow.com/questions/7602919/how-do-i-generate-random-numbers-without-rand-function thread
to create a random sequence on the basis of seed value.
 *                               More randomization have been attained by implementing complex logic using
seed value in the calculations. The random value set generated is different for each seed value.
 * @input:Location where the address to be stored , Length and Seed value
 * @Return : void
 *****************************************************************************************************/
void gen_pattern(uint32_t * pattern, size_t length, int8_t seed)
{

        uint32_t lfsr = 0x1111ACE1; //Create a constant value
        uint32_t bit;
        for (uint32_t i =0 ; i< length ; i++)
        {
                bit  = ((lfsr >> 0) ^ (lfsr >> 2) ^ (lfsr >> 3) ^ (lfsr >> 5) ) & 1; // Shift the buts by
0 ,2, 3,5 to the right and xor the result and and the overall value with one
                lfsr =  (lfsr >> 1) | (bit << 31); //Shit the lsfr by one bit to the right and shift the
above calculated value by 31 bits to left and or the result.
                lfsr = (uint32_t)lfsr;
                *(pattern+i)= lfsr+ (uint32_t)(seed*i*1234);     //Calculate the stored value of the
pattern in the memory location
        }
}
```

```c
/*
 * pattern.h
 *
 *  Created on: Oct 21, 2019
 *      Author: SURAJ THITE
 */

#ifndef PATTERN_H_
#define PATTERN_H_


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

//Function to generate pattern for storing and verification purpose.
void mem_status_write_pattern_array(uint32_t * loc, size_t length, uint8_t seed);
void gen_pattern(uint32_t * pattern, size_t length, int8_t seed);



#endif /* PATTERN_H_ */
```

```c
/*
 * RGBled.c
 *
 *  Created on: Sep 28, 2019
 *      Author:SURAJ THITE , ATHARV DESAI
 */



#include "board.h"
#include "fsl_debug_console.h"
#include "fsl_gpio.h"

#include "clock_config.h"
#include "pin_mux.h"

/*******************************************************************************/
/* Function name:RGB_init
 * Parameters: void
 * Return : void
 * Description: Function to initialize the GPIO RGB Led Pins . */
/*******************************************************************************/
void RGB_init()
{
                gpio_pin_config_t led_blue_config = {
                kGPIO_DigitalOutput, 1,
            };  //Config the pin  for BLUE LED to Digital Output
                GPIO_PinInit(BOARD_LED_BLUE_GPIO,BOARD_LED_BLUE_GPIO_PIN, &led_blue_config);
            gpio_pin_config_t led_red_config = {
                kGPIO_DigitalOutput, 1,
            };  //Config the pin  for RED LED to Digital Output
            GPIO_PinInit(BOARD_LED_RED_GPIO,BOARD_LED_RED_GPIO_PIN, &led_red_config);
            gpio_pin_config_t led_green_config = {
                kGPIO_DigitalOutput, 1,
            };  //Config the pin  for GREEN LED to Digital Output
            GPIO_PinInit(BOARD_LED_GREEN_GPIO,BOARD_LED_GREEN_GPIO_PIN, &led_green_config);
        //Initialize the GPIO Pins
}


/*******************************************************************************/
/* Function name:led_switch(int n )
 * Parameters: current state n
 * Return : void
 * Description: Function to initialize the GPIO RGB Led Pins . */
/*******************************************************************************/
void led_switch(int n)
 {
        GPIO_SetPinsOutput(BOARD_LED_GREEN_GPIO, 1u << BOARD_LED_GREEN_GPIO_PIN); //Clear the Pins
        GPIO_SetPinsOutput(BOARD_LED_RED_GPIO, 1u << BOARD_LED_RED_GPIO_PIN);
        GPIO_SetPinsOutput(BOARD_LED_BLUE_GPIO, 1u << BOARD_LED_BLUE_GPIO_PIN);

        switch (n)
        {
        // Switch LED BLUE ON and TURN OTHER LEDs OFF
        case 0:
                {
                GPIO_ClearPinsOutput(BOARD_LED_BLUE_GPIO, 1u << BOARD_LED_BLUE_GPIO_PIN);
                GPIO_SetPinsOutput(BOARD_LED_RED_GPIO, 1u << BOARD_LED_RED_GPIO_PIN);
                GPIO_SetPinsOutput(BOARD_LED_GREEN_GPIO, 1u << BOARD_LED_GREEN_GPIO_PIN);
                delay(1000);
                }
                break;
                // Switch LED RED ON and TURN OTHER LEDs OFF
        case 1:
        {

                        GPIO_ClearPinsOutput(BOARD_LED_RED_GPIO, 1u << BOARD_LED_RED_GPIO_PIN);
                        GPIO_SetPinsOutput(BOARD_LED_BLUE_GPIO, 1u << BOARD_LED_BLUE_GPIO_PIN);
                        GPIO_SetPinsOutput(BOARD_LED_GREEN_GPIO, 1u << BOARD_LED_GREEN_GPIO_PIN);
                        delay(1000);
        }
                        break;
                        // Switch LED GREEN ON and TURN OTHER LEDs OFF
        case 2:
        {
                        GPIO_ClearPinsOutput(BOARD_LED_GREEN_GPIO, 1u << BOARD_LED_GREEN_GPIO_PIN);
                        GPIO_SetPinsOutput(BOARD_LED_RED_GPIO, 1u << BOARD_LED_RED_GPIO_PIN);
                        GPIO_SetPinsOutput(BOARD_LED_BLUE_GPIO, 1u << BOARD_LED_BLUE_GPIO_PIN);
                        delay(1000);
        }
                        break;
        case 3:
                {
                        // Switch LED BLUE ON and TURN OTHER LEDs OFF
                GPIO_ClearPinsOutput(BOARD_LED_BLUE_GPIO, 1u << BOARD_LED_BLUE_GPIO_PIN);
                GPIO_SetPinsOutput(BOARD_LED_RED_GPIO, 1u << BOARD_LED_RED_GPIO_PIN);
                GPIO_SetPinsOutput(BOARD_LED_GREEN_GPIO, 1u << BOARD_LED_GREEN_GPIO_PIN);
                }
                break;
        }
 }


/********************************************************************************/
/* Function name:RGB_off
 * Parameters: void
 * Return : void
 * Description: Function to turn off the RGB Led Pins . */
/********************************************************************************/

void RGB_OFF()
{       // Clear all the  LEDs.
        GPIO_SetPinsOutput(BOARD_LED_BLUE_GPIO, 1u << BOARD_LED_BLUE_GPIO_PIN);
        GPIO_SetPinsOutput(BOARD_LED_GREEN_GPIO, 1u << BOARD_LED_GREEN_GPIO_PIN);
        GPIO_SetPinsOutput(BOARD_LED_RED_GPIO, 1u << BOARD_LED_RED_GPIO_PIN);
}
```

```c
/*
 * RGBled.h
 *
 *  Created on: Sep 28, 2019
 *   Author:SURAJ THITE , ATHARV DESAI
 */

#ifndef RGBLED_H_
#define RGBLED_H_
void led_switch(int n);  //Function to switch the led_state
void RGB_init();//Function to initialize the RGB Leds
void RGB_OFF(); //Function to turn off the RGB led off
#endif /* RGBLED_H_ */
```

```c
/*****************************************************************************
 *                                                                           *
 *  uCUnit - A unit testing framework for microcontrollers                   *
 *                                                                           *
 *  (C) 2007 - 2008 Sven Stefan Krauss                                       *
 *                  https://www.ucunit.org                                   *
 *                                                                           *
 *  File        : System.c                                                   *
 *  Description : System dependent functions used by uCUnit.                 *
 *                This file runs with arm-elf-run                            *
 *  Author      : Sven Stefan Krauss                                         *
 *  Contact     : www.ucunit.org                                             *
 *                                                                           *
 *****************************************************************************/

/*
 * This file is part of ucUnit.
 *
 * You can redistribute and/or modify it under the terms of the
 * Common Public License as published by IBM Corporation; either
 * version 1.0 of the License, or (at your option) any later version.
 *
 * uCUnit is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
 * Common Public License for more details.
 *
 * You should have received a copy of the Common Public License
 * along with uCUnit.
 *
 * It may also be available at the following URL:
 *       http://www.opensource.org/licenses/cpl1.0.txt
 *
 * If you cannot obtain a copy of the License, please contact the
 * author.
 */
#include <stdio.h>
#include <stdlib.h>
#include "System.h"

/* Stub: Initialize your hardware here */
void System_Init(void)
{

        printf("Init of hardware finished.\n");
}

/* Stub: Shutdown your hardware here */
void System_Shutdown(void)
{

        /* asm("\tSTOP"); */
        printf("System shutdown.\n");
        exit(0);
}

/* Stub: Recover the system */
void System_Recover(void)
{
        /* Stub: Recover the hardware */
        /* asm("\tRESET"); */
        printf("System reset.\n");
        exit(0);
}

/* Stub: Put system in a safe state */
void System_Safestate(void)
{
        /* Disable all port pins */
        /* PORTA = 0x0000; */
        /* PORTB = 0x0000; */
        /* PORTC = 0x0000; */

        /* Disable interrupts */
        /* DIE(); */

        /* Put processor into idle state */
        /* asm("\tIDLE"); */
        printf("System safe state.\n");
        exit(0);
}

/* Stub: Transmit a string to the host/debugger/simulator */
void System_WriteString(char * msg)
{
        printf(msg);
}

void System_WriteInt(int n)
{
        printf("%i", n);
}
```

```c
/*****************************************************************************
 *                                                                           *
 *  uCUnit - A unit testing framework for microcontrollers                   *
 *                                                                           *
 *  (C) 2007 - 2008 Sven Stefan Krauss                                       *
 *                  https://www.ucunit.org                                   *
 *                                                                           *
 *  File        : System.h                                                   *
 *  Description : System dependent functions used by uCUnit.                 *
 *  Author      : Sven Stefan Krauss                                         *
 *  Contact     : www.ucunit.org                                             *
 *                                                                           *
 *****************************************************************************/

/*
 * This file is part of ucUnit.
 *
 * You can redistribute and/or modify it under the terms of the
 * Common Public License as published by IBM Corporation; either
 * version 1.0 of the License, or (at your option) any later version.
 *
 * uCUnit is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
 * Common Public License for more details.
 *
 * You should have received a copy of the Common Public License
 * along with uCUnit.
 *
 * It may also be available at the following URL:
 *       http://www.opensource.org/licenses/cpl1.0.txt
 *
 * If you cannot obtain a copy of the License, please contact the
 * author.
 */
#ifndef SYSTEM_H_
#define SYSTEM_H_

/* function prototypes */
void System_Init(void);
void System_Shutdown(void);
void System_Safestate(void);
void System_Recover(void);
void System_WriteString(char * msg);
void System_WriteInt(int n);

#endif /* SYSTEM_H_ */
```

```c
/*
 * unitTest.c
 *
 *  Created on: Oct 22, 2019
 *      Author: SURAJ THITE ,ATHARV
 */

#include "uCUnit-v1.0.h"
#include "stdint.h"
#include "main.h"
uint32_t  *ptr_test_case;


//test-example-1
/*********************************************************************************************************
 * Function Name:test_memory_allocation_limits(void)
 * Description : this function tests the successful allocation of the memory
 * @input:void
 * @Return :void
 *********************************************************************************************************/

static void test_memory_allocation_limits(void)
{
        UCUNIT_TestcaseBegin("\n \r TEST1:Checking for returned pointer of memory allocation");
        uint32_t  *ptr1;
        ptr1 =  allocate_words(1);
        UCUNIT_CheckIsNull(ptr1);//fail
        UCUNIT_TestcaseEnd();//Fail
        ptr_test_case = ptr1;
}
/*********************************************************************************************************
 * Function Name:test_equal_data(void)
 * Description : this function tests for the equal data at the memory location
 * @input:void
 * @Return :void
 *********************************************************************************************************/

//test-example-2
static void test_equal_data(void)
{
        UCUNIT_TestcaseBegin("\n \r TEST2:check for equality of data in two locations ");
        uint32_t *ptr1 =ptr_test_case;
        //write_pattern(ptr1,4,5);
        UCUNIT_CheckIsEqual(*ptr1,*(ptr1+1)); //Fail
        UCUNIT_CheckIsEqual(*ptr1,*ptr1); //Pass
        UCUNIT_TestcaseEnd(); //Fail
}
/*********************************************************************************************************
 * Function Name:test_bit0_set(void)
 * Description : this function tests whether MSB is set at the memory location
 * @input:void
 * @Return :void
 *********************************************************************************************************/
//test-example-3
static void test_bit0_set(void)
{

        UCUNIT_TestcaseBegin(" \n \r TEST3:Checking whether bit 0 is set");
        uint32_t  *ptr1;
        ptr1 = ptr_test_case;
        write_memory(*ptr1, 0xFFFFFFFF);
        UCUNIT_CheckIsBitSet(*ptr1,0);//Pass
        UCUNIT_TestcaseEnd();

}

/*********************************************************************************************************
 * Function Name:test_bit32_set(void)
 * Description : this function tests whether MSB is set at the memory location
 * @input:void
 * @Return :void
 *******************************************************************************************************/
//test-example-4
static void test_bit32_set(void)
{
        UCUNIT_TestcaseBegin("\n \r TEST4:Checking whether bit 32 is set");
        uint32_t  *ptr1;
        ptr1 = ptr_test_case;
        write_memory(*ptr1, 0xFFFFFFFF);
        UCUNIT_CheckIsBitSet(*ptr1,32);//Pass
        UCUNIT_TestcaseEnd();

}

/*******************************************************************************************************
 * Function Name:void test_bit0_clear(void)
 * Description : this function tests  whether LSB is clear in the data at memory location
 * @input:void
 * @Return :void
 *******************************************************************************************************/
//test-example-5
static void test_bit0_clear(void)
{
        UCUNIT_TestcaseBegin("\n \r TEST5:Checking whether bit 0 is clear");
        uint32_t  *ptr1;
        ptr1 = ptr_test_case;
        write_memory(*ptr1, 0xFFFFFFFF);
        UCUNIT_CheckIsBitClear(*ptr1,0);//Fail
        UCUNIT_TestcaseEnd();
}

/*******************************************************************************************************
 * Function Name:test_bit32_clear(void)
 * Description : this function tests whether MSB is clear in the data at a memory location
 * @input:void
 * @Return :void
 *******************************************************************************************************/

//test-example-6
static void test_bit32_clear(void)
{
        UCUNIT_TestcaseBegin("\n \r TEST6:Checking whether bit 32 is clear");
        uint32_t  *ptr1;
        ptr1 = ptr_test_case;
        write_memory(*ptr1, 0xFFFFFFFF);
        UCUNIT_CheckIsBitClear(*ptr1,32);//Fail
        UCUNIT_TestcaseEnd();
}


/*******************************************************************************************************
 * Function Name:void check_whether_32_bit(void)
 * Description : this function tests the whether data is 32 bit data in the memory location
 * @input:void
 * @Return :void
 *******************************************************************************************************/

//test-example-7
static void check_whether_32_bit(void)
{
        UCUNIT_TestcaseBegin("\n \r TEST7:Checking whether value at location is 32 bit or not ");
        uint32_t  *ptr1;
        ptr1 = ptr_test_case;
        write_memory(*ptr1, 0xFFFFFFFF);
        UCUNIT_CheckIs32Bit(*ptr1); //Pass
        UCUNIT_TestcaseEnd();
}
```

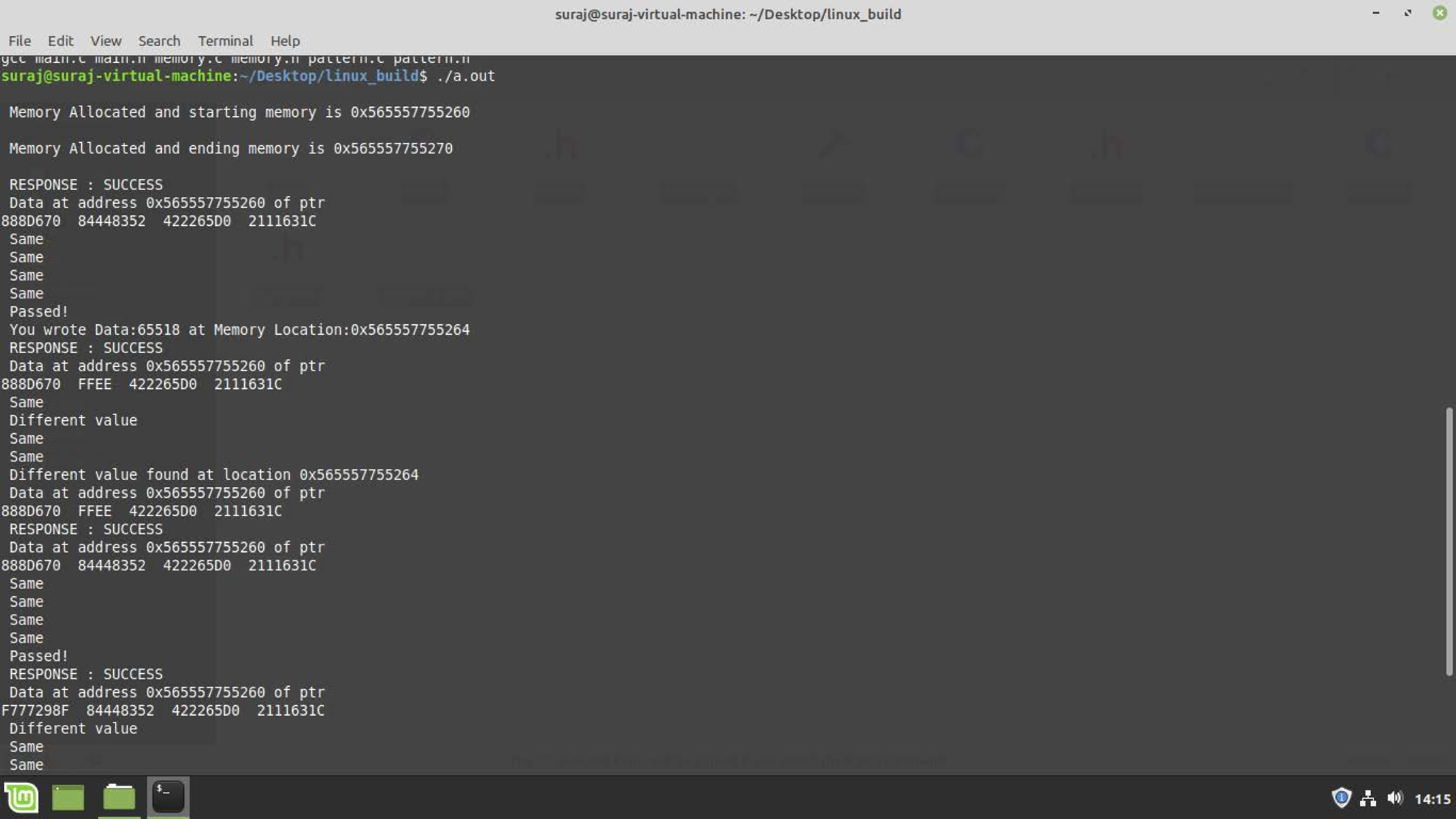
```
/*******************************************************************************************************
 * Function Name:void check_whether_16_bit(void)
 * Description : this function tests the whether data is 16 bit data in the memory location
 * @input:void
 * @Return :void
 *******************************************************************************************************/
//test-example-8
static void check_whether_16_bit(void)
{
        UCUNIT_TestcaseBegin("\n \r TEST8:Checking whether value at location is 16 bit or not ");
        uint32_t  *ptr1;
        ptr1 = ptr_test_case;
        write_memory(*ptr1, 0xFFFFFFFF);
        UCUNIT_CheckIs16Bit(*ptr1); //Fail
        UCUNIT_TestcaseEnd();

}

/*******************************************************************************************************
 * Function Name:void check_whether_8_bit(void)
 * Description : this function tests the whether data is 8 bit data in the memory location
 * @input:void
 * @Return :void
 *******************************************************************************************************/

//test-example-9
static void check_whether_8_bit(void)
{
        UCUNIT_TestcaseBegin("\n \r TEST9:Checking whether value at location is 8 bit or not ");
        uint32_t  *ptr1;
        ptr1 = ptr_test_case;
        write_memory(*ptr1, 0xFFFFFFFF);
        UCUNIT_CheckIs16Bit(*ptr1); //Fail
        UCUNIT_TestcaseEnd();

}


/*******************************************************************************************************
 * Function Name:void check_inversion(void)
 * Description : this function checks the inversion
 * @input:void
 * @Return :void
 *******************************************************************************************************/
//test-example-10
static void check_inversion(void)
{
        UCUNIT_TestcaseBegin("\n \r TEST10:Checking for inversion");
        uint32_t  *ptr1;
        ptr1 = ptr_test_case;
        *(ptr1+4) = 0;
        invert_block(ptr1+4,1);
        UCUNIT_CheckIsEqual(0xFFFFFFFF,ptr1+4);
        UCUNIT_TestcaseEnd();
}

/*******************************************************************************************************
 * Function Name:Testsuite_RunTests(void)
 * Description : this function runs all the ten tests in the test suite
 * @input:void
 * @Return :void
 *******************************************************************************************************/
void  Testsuite_RunTests(void)
{
        test_memory_allocation_limits(); //test 1
        test_equal_data();//test 2
        test_bit0_set();//test 3
        test_bit32_set();//test 4
        test_bit0_clear();//test 5
        test_bit32_clear();//test 6
        check_whether_32_bit();//test 7
        check_whether_16_bit();//test 8
        check_whether_8_bit();//test 9
        check_inversion();//test 10
}

//int main(void)
//{
//    UCUNIT_Init();
//    UCUNIT_WriteString("\n**************************************");
//    UCUNIT_WriteString("uCUnit demo application");
//    UCUNIT_WriteString(__DATE__);
//    UCUNIT_WriteString("\nTime:     ");
//    UCUNIT_WriteString(__TIME__);
//
//    UCUNIT_WriteString("\n**************************************");
////    Testsuite_RunTests();
////    UCUNIT_Shutdown();
//
//    return 0;
//}
```

```c
/*
 * unitTest.h
 *
 *  Created on: Oct 22, 2019
 *      Author: SURAJ THITE , Atharv Desai
 */

#ifndef UNITTEST_H_
#define UNITTEST_H_
#include "main.h"
#include "System.h"

//Test Suite to run tests
void Testsuite_RunTests(void);//TestSuit


//Test functions
void test_memory_allocation_limits(); //test 1
void test_equal_data();//test 2
void test_bit0_set();//test 3
void test_bit32_set();//test 4
void test_bit0_clear();//test 5
void test_bit32_clear();//test 6
void check_whether_32_bit();//test 7
void check_whether_16_bit();//test 8
void check_whether_8_bit();//test 9
void check_inversion();//test 10

#endif /* UNITTEST_H_ */
```

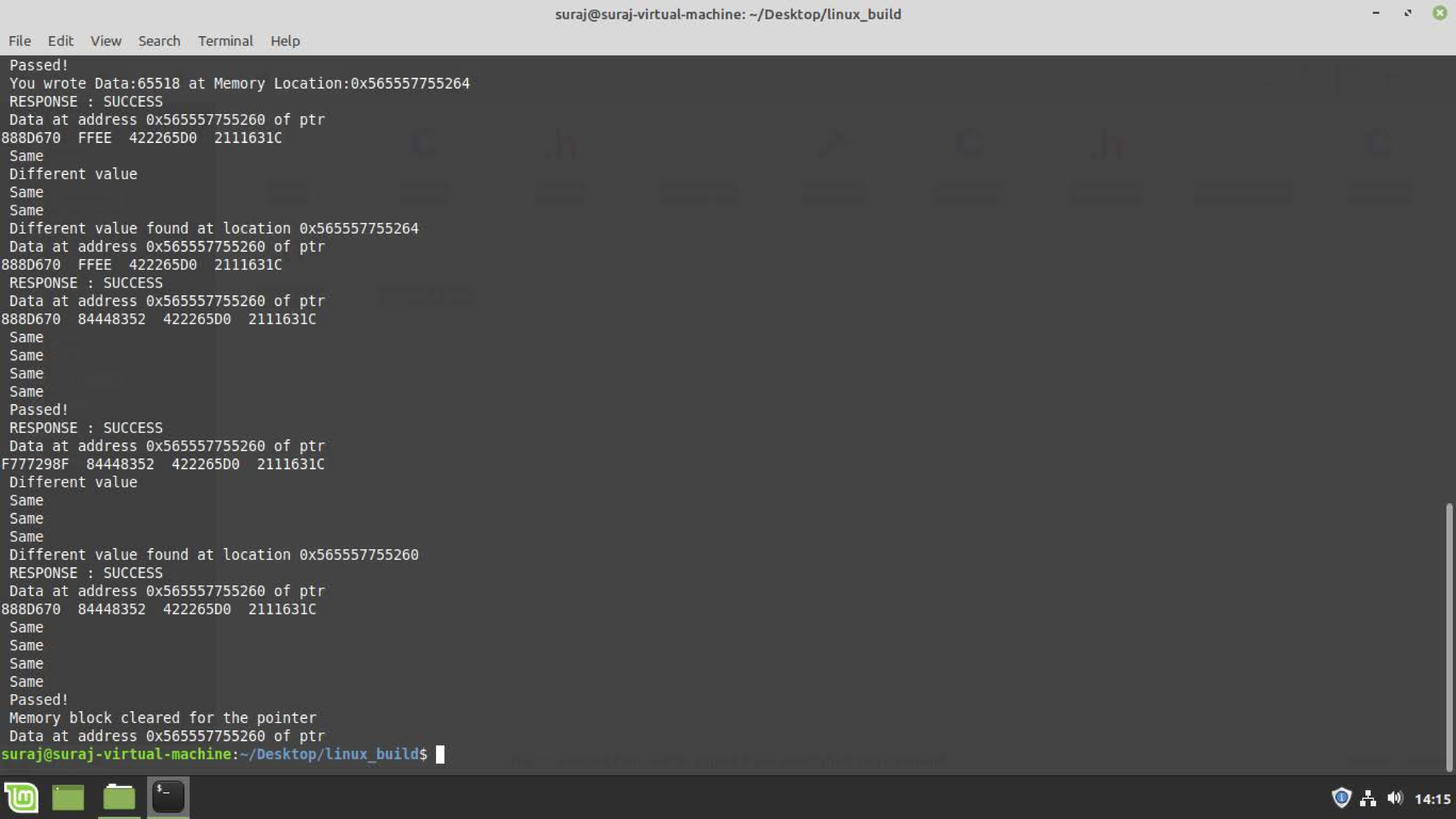
```
Memory Allocated and starting memory is :
The Address is 0x1ffff248

 Memory Allocated and ending memory is :
The Address is 0x1ffff258
 Log Status Value
 1

 RESPONSE : SUCCESS
The Address is 0x1ffff248   and data is 888d670 84448352 422265d0 2111631c

 Same

 Same

 Same

 Same

 Passed!
The Address is 0x1ffff24c   and data is ffee

 RESPONSE : SUCCESS
The Address is 0x1ffff248   and data is 888d670 ffee 422265d0 2111631c

 Same

 Different value
 Same

 Same

 Different value found at location
The Address is 0x1ffff24c
The Address is 0x1ffff248   and data is 888d670 ffee 422265d0 2111631c

 RESPONSE : SUCCESS
The Address is 0x1ffff248   and data is 888d670 84448352 422265d0 2111631c

 Same

 Same

 Same

 Same

 Passed!

 RESPONSE : SUCCESS
The Address is 0x1ffff248   and data is f777298f 84448352 422265d0 2111631c

 Different value

 Same

 Same
 Same


 Different value found at location

The Address is 0x1ffff248


 RESPONSE : SUCCESS

The Address is 0x1ffff248   and data is 888d670 84448352 422265d0 2111631c


 Same


 Same


 Same


 Same


 Passed!


 Memory block cleared for the pointer

The Address is 0x1ffff248   and data is 646e6120 32313665 73690035 73692000

 Running Test Cases

                ======================================


 TEST1:Checking for returned pointer of memory allocation

                                        ======================================



 Memory Allocated and starting memory is :
```

```
The Address is 0x1ffff248


 Memory Allocated and ending memory is :

The Address is 0x1ffff24c  ../source/unitTest.c:27: failed:IsNull(ptr1)

                                                                       ========================================

                                                                                                                       ../source/unitTest.c:28:
failed:EndTestcase()

========================================



========================================


 TEST2:check for equality of data in two locations

                                                    ========================================

                                                                                                      ../source/unitTest.c:44: failed:IsEqual(*ptr1,*(ptr1+1))

                                                                                                                                                         ../source/unitTest.c:45:
passed:IsEqual(*ptr1,*ptr1)

========================================

           ../source/unitTest.c:46: failed:EndTestcase()

                                                         ========================================


                                                                                                         ========================================


 TEST3:Checking whether bit 0 is set

                                  ========================================

                                                                          ../source/unitTest.c:62: failed:IsBitSet(*ptr1,0)


========================================
../source/unitTest.c:63: failed:EndTestcase()

========================================


          ========================================


 TEST4:Checking whether bit 32 is set
                  ========================================
                                    ../source/unitTest.c:80: failed:IsBitSet(*ptr1,32)

========================================

../source/unitTest.c:81: failed:EndTestcase()

========================================


           ========================================


 TEST5:Checking whether bit 0 is clear
                  ========================================
                                    ../source/unitTest.c:98: passed:IsBitClear(*ptr1,0)

========================================
                                                                    Testcase
passed.

========================================



========================================
 TEST6:Checking whether bit 32 is clear

                              ========================================

                                                    ../source/unitTest.c:116: passed:IsBitClear(*ptr1,32)

========================================

                                                                                              Testcase
passed.

========================================



========================================

 TEST7:Checking whether value at location is 32 bit or not

                                        ========================================

                                                              ../source/unitTest.c:135: passed:Is32Bit(*ptr1)

========================================

Testcase passed.

========================================

      ========================================

 TEST8:Checking whether value at location is 16 bit or not

                                        ========================================

                                                              ../source/unitTest.c:152: failed:Is16Bit(*ptr1)

========================================

../source/unitTest.c:153: failed:EndTestcase()
```

```
    ========================================

                                   ========================================

 TEST9:Checking whether value at location is 8 bit or not
                                              ========================================
                                                                                ../source/unitTest.c:171: failed:Is16Bit(*ptr1)

========================================

../source/unitTest.c:172: failed:EndTestcase()
   ========================================

                                  ========================================

 TEST10:Checking for inversion
                         ========================================
                                                              ../source/unitTest.c:191: failed:IsEqual(0xFFFFFFFF,ptr1+4)

========================================

../source/unitTest.c:192: failed:EndTestcase()

========================================
suraj@suraj-virtual-machine:~/Desktop/linux_build$ ./a.out

 Memory Allocated and starting memory is 0x565557755260

 Memory Allocated and ending memory is 0x565557755270

 RESPONSE : SUCCESS
 Data at address 0x565557755260 of ptr
888D670  84448352  422265D0  2111631C
 Same
 Same
 Same
 Same
 Passed!
 You wrote Data:65518 at Memory Location:0x565557755264
 RESPONSE : SUCCESS
 Data at address 0x565557755260 of ptr
888D670  FFEE  422265D0  2111631C
 Same
 Different value
 Same
 Same
 Different value found at location 0x565557755264
 Data at address 0x565557755260 of ptr
888D670  FFEE  422265D0  2111631C
 RESPONSE : SUCCESS
 Data at address 0x565557755260 of ptr
888D670  84448352  422265D0  2111631C
 Same
 Same
 Same
 Same
 Passed!
 RESPONSE : SUCCESS
 Data at address 0x565557755260 of ptr
F777298F  84448352  422265D0  2111631C
 Different value
 Same
 Same
 Same
 Different value found at location 0x565557755260
 RESPONSE : SUCCESS
 Data at address 0x565557755260 of ptr
888D670  84448352  422265D0  2111631C
 Same
 Same
 Same
 Same
 Passed!
 Memory block cleared for the pointer
 Data at address 0x565557755260 of ptr
```


suraj@suraj-virtual-machine: ~/Desktop/linux_build
File Edit View Search Terminal Help
14:15

```
gcc main.c main.h memory.c memory.h pattern.c pattern.h
suraj@suraj-virtual-machine:~/Desktop/linux_build$ ./a.out

 Memory Allocated and starting memory is 0x565557755260

 Memory Allocated and ending memory is 0x565557755270

 RESPONSE : SUCCESS
 Data at address 0x565557755260 of ptr
888D670  84448352  422265D0  2111631C
 Same
 Same
 Same
 Same
 Passed!
 You wrote Data:65518 at Memory Location:0x565557755264
 RESPONSE : SUCCESS
 Data at address 0x565557755260 of ptr
888D670  FFEE  422265D0  2111631C
 Same
 Different value
 Same
 Same
 Different value found at location 0x565557755264
 Data at address 0x565557755260 of ptr
888D670  FFEE  422265D0  2111631C
 RESPONSE : SUCCESS
 Data at address 0x565557755260 of ptr
888D670  84448352  422265D0  2111631C
 Same
 Same
 Same
 Same
 Passed!
 RESPONSE : SUCCESS
 Data at address 0x565557755260 of ptr
F777298F  84448352  422265D0  2111631C
 Different value
 Same
 Same
```


suraj@suraj-virtual-machine: ~/Desktop/linux_build
File Edit View Search Terminal Help

```
 Passed!
 You wrote Data:65518 at Memory Location:0x565557755264
 RESPONSE : SUCCESS
 Data at address 0x565557755260 of ptr
888D670  FFEE  422265D0  2111631C
 Same
 Different value
 Same
 Same
 Different value found at location 0x565557755264
 Data at address 0x565557755260 of ptr
888D670  FFEE  422265D0  2111631C
 RESPONSE : SUCCESS
 Data at address 0x565557755260 of ptr
888D670  84448352  422265D0  2111631C
 Same
 Same
 Same
 Same
 Passed!
 RESPONSE : SUCCESS
 Data at address 0x565557755260 of ptr
F777298F  84448352  422265D0  2111631C
 Different value
 Same
 Same
 Same
 Different value found at location 0x565557755260
 RESPONSE : SUCCESS
 Data at address 0x565557755260 of ptr
888D670  84448352  422265D0  2111631C
 Same
 Same
 Same
 Same
 Passed!
 Memory block cleared for the pointer
 Data at address 0x565557755260 of ptr
suraj@suraj-virtual-machine:~/Desktop/linux_build$
```