

COMPENG 4DS4 Lab 0 Report

Aaron Pinto pintoa9 Raed Hassan hassam41
Jingming Liu liuj171 Jeffrey Guo guoj69

January 27, 2023

Problem 1

The code listing for problem 1 is Listing 1. The MEM_LOC macro was redefined for the different sizes we required (MEM_LOC_CHAR for 1 byte, MEM_LOC_SHORT for 2 bytes, and MEM_LOC_INT for 4 bytes). The required addresses were defined in the macros LOC1-LOC4, using the new macros created for MEM_LOC. These macros are used in the function `problem1` to write the required values to the memory addresses.

The memory and expression panels for problem 1 are shown in Figure 1. The endianness was set to big, the cell size to 1 byte, and the radix to hex.

```
1  /* Definitions */
2  #define MEM_LOC_CHAR(x)          *((char*)x)
3  #define MEM_LOC_SHORT(x)        *((short*)x)
4  #define MEM_LOC_INT(x)          *((int*)x)
5  #define LOC1                    MEM_LOC_CHAR(0x20001000)
6  #define LOC2                    MEM_LOC_INT(0x20001001)
7  #define LOC3                    MEM_LOC_SHORT(0x20001005)
8  #define LOC4                    MEM_LOC_INT(0x20001007)
9
10 /* Problem 1 Function */
11 void problem1() {
12     LOC1 = 0xAC;
13     LOC2 = 0xAABBCCDD;
14     LOC3 = 0xABCD;
15     LOC4 = 0xAABBCCDD;
16 }
```

Listing 1: Problem 1

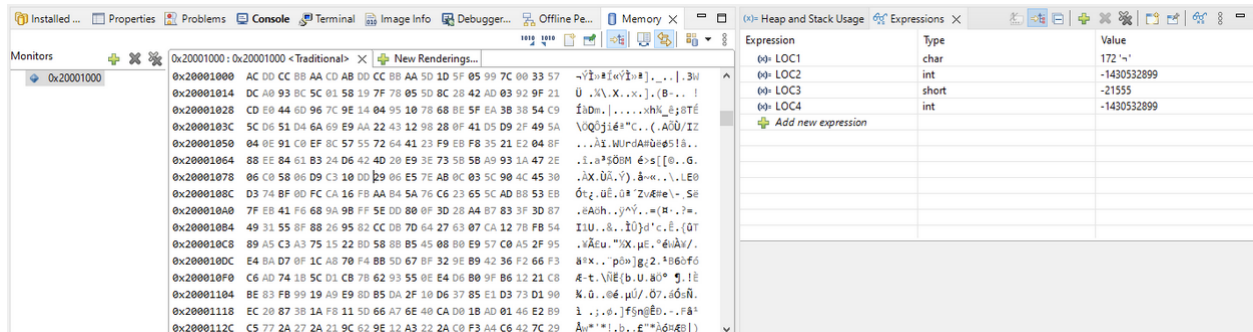


Figure 1: Memory and Expression Panels for Problem 1

Problem 3

The code listing for problem 3 is Listing 2. The `BOARD_InitPins` function was updated to initialize the appropriate GPIO ports and pins (GPIOC 8 and 9 for blue and green, GPIOD 1 for red). The `GPIO_PinInit` and `GPIO_PortToggle` functions provided in the example were used to initialize the LEDs to an OFF state and for toggling the LEDs in the while loop.

```
1  /* Definitions */
2  #define BOARD_LED_GPIO_BLUE     GPIOC
3  #define BOARD_LED_GPIO_PIN_BLUE 8
4  #define BOARD_LED_GPIO_GREEN   GPIOC
5  #define BOARD_LED_GPIO_PIN_GREEN 9
```

```

6  #define BOARD_LED_GPIO_RED      GPIOD
7  #define BOARD_LED_GPIO_PIN_RED  1
8
9  /* BOARD_InitPins function */
10 void BOARD_InitPins(void) {
11     /* Port C Clock Gate Control: Clock enabled */
12     CLOCK_EnableClock(kCLOCK_PortC);
13     /* Port D Clock Gate Control: Clock enabled */
14     CLOCK_EnableClock(kCLOCK_PortD);
15
16     /* PORTC8 is configured as PTC8 */
17     PORT_SetPinMux(PORTC, 8U, kPORT_MuxAsGpio);
18
19     /* PORTC9 is configured as PTC9 */
20     PORT_SetPinMux(PORTC, 9U, kPORT_MuxAsGpio);
21
22     /* PORTD1 is configured as PTD1 */
23     PORT_SetPinMux(PORTD, 1U, kPORT_MuxAsGpio);
24 }
25
26
27 /* Problem 3 main Function */
28 int main(void) {
29     /* Define the init structure for the output LED pin*/
30     gpio_pin_config_t led_config = {kGPIO_DigitalOutput, 0};
31
32     /* Board pin, clock, debug console init */
33     BOARD_InitBootPins();
34     BOARD_InitBootClocks();
35     BOARD_InitDebugConsole();
36
37     /* Print a note to terminal. */
38     PRINTF("\r\n GPIO Driver example\r\n");
39     PRINTF("\r\n The LED is blinking.\r\n");
40
41     /* Init output LED GPIO. */
42     GPIO_PinInit(BOARD_LED_GPIO_BLUE, BOARD_LED_GPIO_PIN_BLUE, &led_config);
43     GPIO_PinInit(BOARD_LED_GPIO_GREEN, BOARD_LED_GPIO_PIN_GREEN, &led_config);
44     GPIO_PinInit(BOARD_LED_GPIO_RED, BOARD_LED_GPIO_PIN_RED, &led_config);
45
46     while (1) {
47         delay();
48         GPIO_PortToggle(BOARD_LED_GPIO_BLUE, 1u << BOARD_LED_GPIO_PIN_BLUE);
49         delay();
50         GPIO_PortToggle(BOARD_LED_GPIO_BLUE, 1u << BOARD_LED_GPIO_PIN_BLUE);
51         delay();
52         GPIO_PortToggle(BOARD_LED_GPIO_GREEN, 1u << BOARD_LED_GPIO_PIN_GREEN);
53         delay();
54         GPIO_PortToggle(BOARD_LED_GPIO_GREEN, 1u << BOARD_LED_GPIO_PIN_GREEN);
55         delay();
56         GPIO_PortToggle(BOARD_LED_GPIO_RED, 1u << BOARD_LED_GPIO_PIN_RED);
57         delay();
58         GPIO_PortToggle(BOARD_LED_GPIO_RED, 1u << BOARD_LED_GPIO_PIN_RED);
59     }
60 }

```

Listing 2: Problem 3

Problem 4

The code listing for problem 4 is Listing 3. A structure `GPIO_Struct` was created to contain the 6 GPIO registers, which we confirmed each contained 32 bits from the datasheet. The macros `GPIOA-GPIOE` were defined using the address of the first GPIO register for the GPIO

port as listed in the datasheet.

Three helper functions were created for the solution, `initPin`, `togglePin`, and `writePin`. `initPin` calls `writePin` to ensure the pin is set such that the LED is OFF, and also sets the Port Data Direction Register (PDDR) to 1, configuring the pin as an output. `togglePin` simply performs an XOR between the Port Data Output Register (PDOR) and a mask (which points to the appropriate pin for the register), toggling the data output value in the register pin. `writePin` updates the data output value of a pin to a desired value (0 or 1) by writing the value to the PDOR pin. These helper functions are used similarly to the provided GPIO example functions in problem 3.

```
1  /* GPIO Register Structure */
2  typedef struct {
3      uint32_t GPIO_PDOR;
4      uint32_t GPIO_PSOR;
5      uint32_t GPIO_PCOR;
6      uint32_t GPIO_PTOR;
7      uint32_t GPIO_PDIR;
8      uint32_t GPIO_PDDR;
9  } GPIO_Struct;
10
11 /* Definitions */
12 #define GPIOA ((GPIO_Struct*) 0x400FF000)
13 #define GPIOB ((GPIO_Struct*) 0x400FF040)
14 #define GPIOC ((GPIO_Struct*) 0x400FF080)
15 #define GPIOD ((GPIO_Struct*) 0x400FF0C0)
16 #define GPIOE ((GPIO_Struct*) 0x400FF100)
17
18 /* Problem 4 Helper Functions */
19 void initPin(GPIO_Struct *port, uint32_t pin) {
20     writePin(port, pin, 1);
21     port->GPIO_PDDR |= (1UL << pin);
22 }
23
24 void togglePin(GPIO_Struct *port, uint32_t mask) {
25     port->GPIO_PDOR ^= (mask);
26 }
27
28 void writePin(GPIO_Struct *port, uint32_t pin, uint8_t output) {
29     if (output == 0U) {
30         port->GPIO_PDOR |= (1UL << pin);
31     } else {
32         port->GPIO_PDOR &= ~(1UL << pin);
33     }
34 }
35
36 /* Problem 4 main Function */
37 int main(void) {
38     /* Board pin, clock, debug console init */
39     BOARD_InitBootPins();
40     BOARD_InitBootClocks();
41     BOARD_InitDebugConsole();
42
43     /* Print a note to terminal. */
44     PRINTF("\r\nGPIO Driver example\r\n");
45     PRINTF("\r\nThe LED is blinking.\r\n");
46
47     /* Init output LED GPIO. */
48     initPin(BOARD_LED_GPIO_BLUE, BOARD_LED_GPIO_PIN_BLUE);
49     initPin(BOARD_LED_GPIO_GREEN, BOARD_LED_GPIO_PIN_GREEN);
50     initPin(BOARD_LED_GPIO_RED, BOARD_LED_GPIO_PIN_RED);
51
52     while (1) {
53         delay();
```

```

54     togglePin(BOARD_LED_GPIO_BLUE, 1u << BOARD_LED_GPIO_PIN_BLUE);
55     delay();
56     togglePin(BOARD_LED_GPIO_BLUE, 1u << BOARD_LED_GPIO_PIN_BLUE);
57     delay();
58     togglePin(BOARD_LED_GPIO_GREEN, 1u << BOARD_LED_GPIO_PIN_GREEN);
59     delay();
60     togglePin(BOARD_LED_GPIO_GREEN, 1u << BOARD_LED_GPIO_PIN_GREEN);
61     delay();
62     togglePin(BOARD_LED_GPIO_RED, 1u << BOARD_LED_GPIO_PIN_RED);
63     delay();
64     togglePin(BOARD_LED_GPIO_RED, 1u << BOARD_LED_GPIO_PIN_RED);
65 }
66 }

```

Listing 3: Problem 4

Problem 5

The code listing for problem 5 is split between Listing 4 for the FTM/PWM setup, and Listing 5 for the main function.

In the FTM setup, we update the `pwm_setup` function provided in the lab to include the FTM setup for `UI_LED_GREEN` and `UI_LED_BLUE`. The appropriate alternative functions to output PWM signals for these GPIO pins were documented in the datasheet.

```

1  void pwm_setup() {
2      ftm_config_t ftmInfo;
3      ftm_chnl_pwm_signal_param_t ftmParam;
4
5      /* UI_LED_RED PWM Setup */
6      ftmParam.chnlNumber = kFTM_Chnl_1;
7      ftmParam.level = kFTM_HighTrue;
8      ftmParam.dutyCyclePercent = 0;
9      ftmParam.firstEdgeDelayPercent = 0U;
10     ftmParam.enableComplementary = false;
11     ftmParam.enableDeadtime = false;
12
13     FTM_GetDefaultConfig(&ftmInfo);
14
15     FTM_Init(FTM3, &ftmInfo);
16     FTM_SetupPwm(FTM3, &ftmParam, 1U, kFTM_EdgeAlignedPwm, 5000U, CLOCK_GetFreq(
17         kCLOCK_BusClk));
18     FTM_StartTimer(FTM3, kFTM_SystemClock);
19
20     /* UI_LED_BLUE PWM Setup */
21     ftmParam.chnlNumber = kFTM_Chnl_4;
22     ftmParam.level = kFTM_HighTrue;
23     ftmParam.dutyCyclePercent = 0;
24     ftmParam.firstEdgeDelayPercent = 0U;
25     ftmParam.enableComplementary = false;
26     ftmParam.enableDeadtime = false;
27
28     FTM_GetDefaultConfig(&ftmInfo);
29
30     FTM_Init(FTM3, &ftmInfo);
31     FTM_SetupPwm(FTM3, &ftmParam, 1U, kFTM_EdgeAlignedPwm, 5000U, CLOCK_GetFreq(
32         kCLOCK_BusClk));
33     FTM_StartTimer(FTM3, kFTM_SystemClock);
34
35     /* UI_LED_GREEN PWM Setup */
36     ftmParam.chnlNumber = kFTM_Chnl_5;
37     ftmParam.level = kFTM_HighTrue;
38     ftmParam.dutyCyclePercent = 0;

```

```

37     ftmParam.firstEdgeDelayPercent = 0U;
38     ftmParam.enableComplementary = false;
39     ftmParam.enableDeadtime = false;
40
41     FTM_GetDefaultConfig(&ftmInfo);
42
43     FTM_Init(FTM3, &ftmInfo);
44     FTM_SetupPwm(FTM3, &ftmParam, 1U, kFTM_EdgeAlignedPwm, 5000U, CLOCK_GetFreq(
45         kCLOCK_BusClk));
46     FTM_StartTimer(FTM3, kFTM_SystemClock);
47 }

```

Listing 4: Problem 5 FTM/PWM Setup

The `main` function provided in the lab was modified to include the PWM functions to update `UI_LED_GREEN` and `UI_LED_BLUE`. The `scanf` was modified to read in 6 characters and store each two character inputs as two hexadecimal characters in variables corresponding to the duty cycle for the red, green, and blue LEDs. The percent value of these inputs (out of `0xFF`) are calculated as the `FTM_UpdatePwmDutycycle` function requires the duty cycle to be an `uint8_t` between 0 and 100 representing the duty cycle percentage.

```

1  int main(void) {
2      unsigned int duty_cycle_red = 0;
3      unsigned int duty_cycle_green = 0;
4      unsigned int duty_cycle_blue = 0;
5
6      /* Init board hardware. */
7      BOARD_InitBootPins();
8      BOARD_InitBootClocks();
9      BOARD_InitDebugConsole();
10
11     pwm_setup();
12
13     scanf("%2x%2x%2x", &duty_cycle_red, &duty_cycle_green, &duty_cycle_blue);
14
15     printf("red = %x\n", duty_cycle_red);
16     printf("green = %x\n", duty_cycle_green);
17     printf("blue = %x\n", duty_cycle_blue);
18
19     float red = (duty_cycle_red / 255.0) * 100;
20     float green = (duty_cycle_green / 255.0) * 100;
21     float blue = (duty_cycle_blue / 255.0) * 100;
22
23     FTM_UpdatePwmDutycycle(FTM3, kFTM_Chnl_1, kFTM_EdgeAlignedPwm, (uint8_t)red);
24     FTM_SetSoftwareTrigger(FTM3, true);
25
26     FTM_UpdatePwmDutycycle(FTM3, kFTM_Chnl_5, kFTM_EdgeAlignedPwm, (uint8_t)green);
27     FTM_SetSoftwareTrigger(FTM3, true);
28
29     FTM_UpdatePwmDutycycle(FTM3, kFTM_Chnl_4, kFTM_EdgeAlignedPwm, (uint8_t)blue);
30     FTM_SetSoftwareTrigger(FTM3, true);
31
32     while (1) {
33     }
34 }

```

Listing 5: Problem 5 main