COMPENG 4DS4 Lab 0 Report

Aaron Pinto
pintoa9

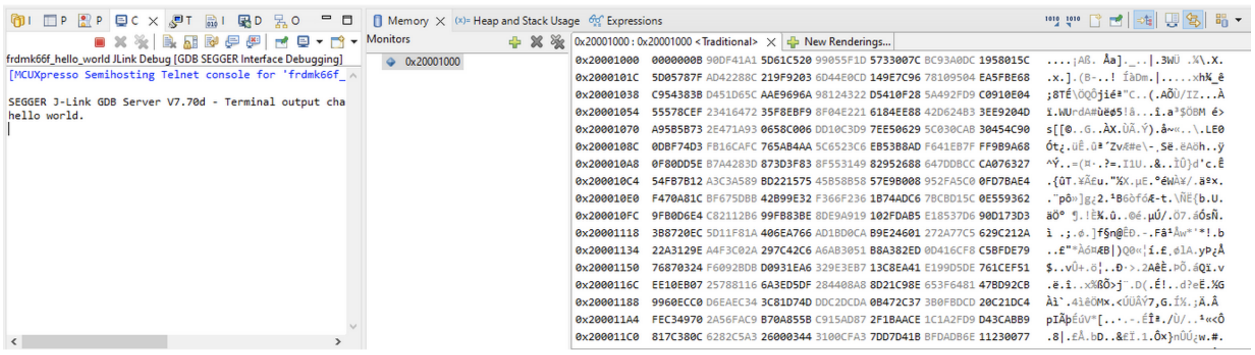
Raeed Hassan
hassam41

Jingming Liu
liuj171

Jeffrey Guo
guoj69

January 27, 2023

Experiment 2 Setup: Part A



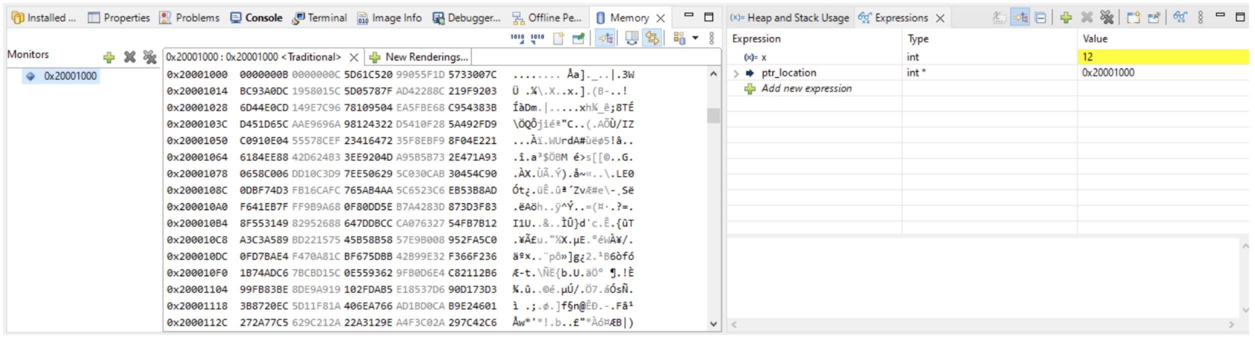
```
frdmk66f_hello_world JLink Debug [GDB SEGGER Interface Debugging]
[MCUXpresso Semihosting Telnet console for 'frdmk66f_

SEGGER J-Link GDB Server V7.70d - Terminal output cha
hello world.
```

```
0x20001000 : 0x20001000 <Traditional>
0x20001000   0000000B 90DF41A1 5D61C520 99055F1D 5733007C BC93A0DC 1958015C   ....¡Aß. Åa]._..|.3WÜ .%\.X.
0x2000101C   5D05787F AD42288C 219F9203 6D44E0CD 149E7C96 78109504 EA5FBE68   .x.].(B-..! ÍàDm.|.....xh¾_ê
0x20001038   C954383B D451D65C AAE9696A 98124322 D5410F28 5A492FD9 C0910E04   ;8TÉ\ÖQÔjiéª"C..(.AÕÙ/IZ...À
0x20001054   55578CEF 23416472 35F8EBF9 8F04E221 6184EE88 42D624B3 3EE9204D   ï.WUrdA#ùëø5!â...î.a³$ÖBM é>
0x20001070   A95B5B73 2E471A93 0658C006 DD10C3D9 7EE50629 5C030CAB 30454C90   s[[©..G..ÀX.ÙÃ.Ý).å~«..\.LE0
0x2000108C   0DBF74D3 FB16CAFC 765AB4AA 5C6523C6 EB53B8AD F641EB7F FF9B9A68   Ót¿.üÊ.ûª´ZvÆ#e\-¸Së.ëAöh..ÿ
0x200010A8   0F80DD5E B7A4283D 873D3F83 8F553149 82952688 647DDBCC CA076327   ^Ý..=(¤·.?=.I1U..&..ÌÛ}d'c.Ê
0x200010C4   54FB7B12 A3C3A589 BD221575 45B58B58 57E9B008 952FA5C0 0FD7BAE4   .{ûT.¥Ã£u."½X.µE.°éWÀ¥/.äºx.
0x200010E0   F470A81C BF675DBB 42B99E32 F366F236 1B74ADC6 7BCBD15C 0E559362   .¨pô»]g¿2.¹B6òfóÆ-t.\ÑË{b.U.
0x200010FC   9FB0D6E4 C82112B6 99FB83BE 8DE9A919 102FDAB5 E18537D6 90D173D3   äÖ° ¶.!È¾.û..©é.µÚ/.Ö7.áÓsÑ.
0x20001118   3B8720EC 5D11F81A 406EA766 AD1BD0CA B9E24601 272A77C5 629C212A   ì .;.ø.]f§n@ÊÐ.-.Fâ¹Åw*'*!.b
0x20001134   22A3129E A4F3C02A 297C42C6 A6AB3051 B8A382ED 0D416CF8 C5BFDE79   ..£"*Àó¤ÆB|)Q0«¦í.£¸ølA.yÞ¿Å
0x20001150   76870324 F6092BDB D0931EA6 329E3EB7 13C8EA41 E199D5DE 761CEF51   $..vÛ+.ö¦..Ð·>.2AêÈ.ÞÕ.áQï.v
0x2000116C   EE10EB07 25788116 6A3ED5DF 284408A8 8D21C98E 653F6481 47BD92CB   .ë.î..x%ßÕ>j¨.D(.É!..d?eË.½G
0x20001188   9960ECC0 D6EAEC34 3C81D74D DDC2DCDA 0B472C37 3B0FBDCD 20C21DC4   Àì`.4ìêÖMx.<ÚÜÂÝ7,G.Í½.;Ä.Â
0x200011A4   FEC34970 2A56FAC9 B70A855B C915AD87 2F1BAACE 1C1A2FD9 D43CABB9   pIÃþÉúV*[...·.-.ÉÎª./Ù/..¹«<Ô
0x200011C0   817C380C 6282C5A3 26000344 3100CFA3 7DD7D41B BFDADB6E 11230077   .8|.£Å.bD..&£Ï.1.Ô×}nÛÚ¿w.#.
```

Figure 1: Memory and Expression Panels for Experiment 2 Setup: Part A

Experiment 2 Setup: Part B



```
0x20001000 : 0x20001000 <Traditional>
0x20001000   0000000B 0000000C 5D61C520 99055F1D 5733007C   ........ Åa]._..|.3W
0x20001014   BC93A0DC 1958015C 5D05787F AD42288C 219F9203   Ü .%\.X..x.].(B-..!
0x20001028   6D44E0CD 149E7C96 78109504 EA5FBE68 C954383B   ÍàDm.|.....xh¾_ê;8TÉ
0x2000103C   D451D65C AAE9696A 98124322 D5410F28 5A492FD9   \ÖQÔjiéª"C..(.AÕÙ/IZ
0x20001050   C0910E04 55578CEF 23416472 35F8EBF9 8F04E221   ...Àï.WUrdA#ùëø5!â..
0x20001064   6184EE88 42D624B3 3EE9204D A95B5B73 2E471A93   .î.a³$ÖBM é>s[[©..G.
0x20001078   0658C006 DD10C3D9 7EE50629 5C030CAB 30454C90   .ÀX.ÙÃ.Ý).å~«..\.LE0
0x2000108C   0DBF74D3 FB16CAFC 765AB4AA 5C6523C6 EB53B8AD   Ót¿.üÊ.ûª´ZvÆ#e\-¸Së
0x200010A0   F641EB7F FF9B9A68 0F80DD5E B7A4283D 873D3F83   .ëAöh..ÿ^Ý..=(¤·.?=.
0x200010B4   8F553149 82952688 647DDBCC CA076327 54FB7B12   I1U..&..ÌÛ}d'c.Ê.{ûT
0x200010C8   A3C3A589 BD221575 45B58B58 57E9B008 952FA5C0   .¥Ã£u."½X.µE.°éWÀ¥/.
0x200010DC   0FD7BAE4 F470A81C BF675DBB 42B99E32 F366F236   äºx..¨pô»]g¿2.¹B6òfó
0x200010F0   1B74ADC6 7BCBD15C 0E559362 9FB0D6E4 C82112B6   Æ-t.\ÑË{b.U.äÖ° ¶.!È
0x20001104   99FB83BE 8DE9A919 102FDAB5 E18537D6 90D173D3   ¾.û..©é.µÚ/.Ö7.áÓsÑ.
0x20001118   3B8720EC 5D11F81A 406EA766 AD1BD0CA B9E24601   ì .;.ø.]f§n@ÊÐ.-.Fâ¹
0x2000112C   272A77C5 629C212A 22A3129E A4F3C02A 297C42C6   Åw*'*!.b..£"*Àó¤ÆB|)
```

Expression	Type	Value
x	int	12
ptr_location	int *	0x20001000
Add new expression		

Figure 2: Memory and Expression Panels for Experiment 2 Setup: Part B

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.

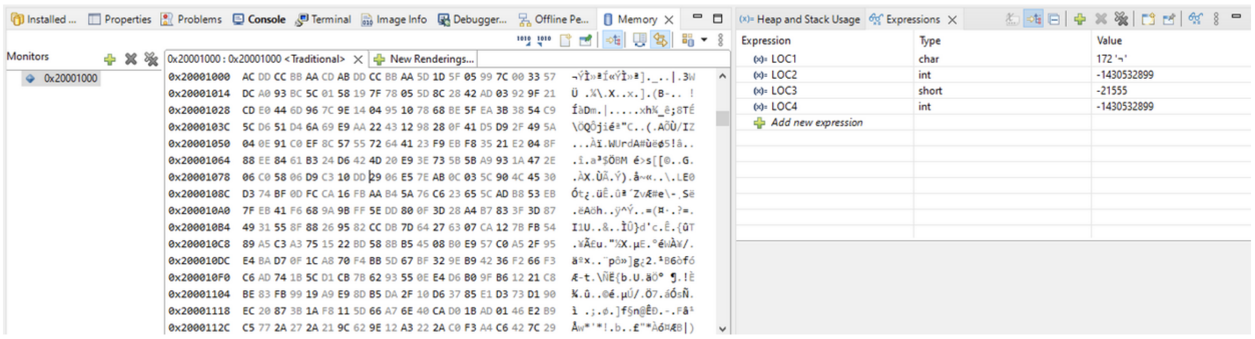
The memory and expression panels for problem 1 are shown in Figure 3. The endianness was set to big, the cell size to 1 byte, and the radix to hex.

```c
/* Definitions */
#define MEM_LOC_CHAR(x)             *((char*)x)
#define MEM_LOC_SHORT(x)            *((short*)x)
#define MEM_LOC_INT(x)              *((int*)x)
#define LOC1                        MEM_LOC_CHAR(0x20001000)
#define LOC2                        MEM_LOC_INT(0x20001001)
#define LOC3                        MEM_LOC_SHORT(0x20001005)
#define LOC4                        MEM_LOC_INT(0x20001007)

```

```c
/* Problem 1 Function */
void problem1() {
    LOC1 = 0xAC;
    LOC2 = 0xAABBCCDD;
    LOC3 = 0xABCD;
    LOC4 = 0xAABBCCDD;
}
```

Listing 1: Problem 1



```
0x20001000   AC DD CC BB AA CD AB DD CC BB AA 5D 1D 5F 05 99 7C 00 33 57   ¬ÝÌ»ªÍ«ÝÌ»ª]._..|.3W
0x20001014   DC A0 93 BC 5C 01 58 19 7F 78 05 5D 8C 28 42 AD 03 92 9F 21   Ü .¼\.X..x.].(B-.. !
0x20001028   CD E0 44 6D 96 7C 9E 14 04 95 10 78 68 BE 5F EA 3B 38 54 C9   ÍàDm.|.....xh¾_ê;8TÉ
0x2000103C   5C D6 51 D4 6A 69 E9 AA 22 43 12 98 28 0F 41 D5 D9 2F 49 5A   \ÖQÔjiéª"C..(.AÕÙ/IZ
0x20001050   04 0E 91 C0 EF 8C 57 55 72 64 41 23 F9 EB F8 35 21 E2 04 8F   ...Àï.WUrdA#ùëø5!â..
0x20001064   88 EE 84 61 B3 24 D6 42 4D 20 E9 3E 73 5B 5B A9 93 1A 47 2E   .î.a³$ÖBM é>s[[©..G.
0x20001078   06 C0 58 06 D9 C3 10 DD 29 06 E5 7E AB 0C 03 5C 90 4C 45 30   .ÀX.ÙÃ.Ý).å~«..\.LE0
0x2000108C   D3 74 BF 0D FC CA 16 FB AA B4 5A 76 C6 23 65 5C AD B8 53 EB   Ót¿.üÊ.ûª´ZvÆ#e\-¸Së
0x200010A0   7F EB 41 F6 68 9A 9B FF 5E DD 80 0F 3D 28 A4 B7 83 3F 3D 87   .ëAöh..ÿ^Ý..=(¤·.?=.
0x200010B4   49 31 55 8F 88 26 95 82 CC DB 7D 64 27 63 07 CA 12 7B FB 54   I1U..&..ÌÛ}d'c.Ê.{ûT
0x200010C8   89 A5 C3 A3 75 15 22 BD 58 8B B5 45 08 B0 E9 57 C0 A5 2F 95   .¥Ã£u."½X.µE.°éWÀ¥/.
0x200010DC   E4 BA D7 0F 1C A8 70 F4 BB 5D 67 BF 32 9E B9 42 36 F2 66 F3   äº×..¨pô»]g¿2.¹B6òfó
0x200010F0   C6 AD 74 1B 5C D1 CB 7B 62 93 55 0E E4 D6 B0 9F B6 12 21 C8   Æ-t.\ÑË{b.U.äÖ°.¶.!È
0x20001104   BE 83 FB 99 19 A9 E9 8D B5 DA 2F 10 D6 37 85 E1 D3 73 D1 90   ¾.û..©é.µÚ/.Ö7.áÓsÑ.
0x20001118   EC 20 87 3B 1A F8 11 5D 66 A7 6E 40 CA D0 1B AD 01 46 E2 B9   ì .;.ø.]f§n@ÊÐ.-.Fâ¹
0x2000112C   C5 77 2A 27 2A 21 9C 62 9E 12 A3 22 2A C0 F3 A4 C6 42 7C 29   Åw*'*!.b..£"*Àó¤ÆB|)
```

Expression	Type	Value
LOC1	char	172 '¬'
LOC2	int	-1430532899
LOC3	short	-21555
LOC4	int	-1430532899
Add new expression		

Figure 3: Memory and Expression Panels for Problem 1

Problem 2

`struct1` will take up 8 bytes. The `char x2` will use 4 bytes as it will be aligned from 1 byte to 4 bytes, and the `int x1` will also use 4 bytes. There will be 3 extra bytes between `x2` and `x1`.

x x x x x x 2 2	1 1 1 1 1 1 1 1

`struct2` will take up 8 bytes. The `short x2` will use 4 bytes as it will be aligned from 2 bytes to 4 bytes, and the `int x1` will also use 4 bytes. There will be 2 extra bytes between `x2` and `x1`.

x x x x 2 2 2 2	1 1 1 1 1 1 1 1

`struct3` will take up 8 bytes. The `int x1` will use 4 bytes, and the `short x2` will also use 4 bytes as it will be aligned from 2 bytes to 4 bytes. There will be one extra byte after `x2`.

1 1 1 1 1 1 1 1	x x x x 2 2 2 2

`struct4` will take up 12 bytes. The `inner_struct inner_struct_1` will use 8 bytes as it will be aligned from 7 bytes to 8 bytes, and the `int x1` will also use 4 bytes. There will be one extra byte between `inner_struct inner_struct_1` and `int x1`.

3 3 2 2 2 2 1 1	x x 3 3 3 3 3 3	1 1 1 1 1 1 1 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.

```c
/* Definitions */
#define BOARD_LED_GPIO_BLUE         GPIOC
#define BOARD_LED_GPIO_PIN_BLUE     8
#define BOARD_LED_GPIO_GREEN        GPIOC
#define BOARD_LED_GPIO_PIN_GREEN    9
#define BOARD_LED_GPIO_RED          GPIOD
#define BOARD_LED_GPIO_PIN_RED      1

/* BOARD_InitPins function */
void BOARD_InitPins(void) {
    /* Port C Clock Gate Control: Clock enabled */
    CLOCK_EnableClock(kCLOCK_PortC);
    /* Port D Clock Gate Control: Clock enabled */
    CLOCK_EnableClock(kCLOCK_PortD);

    /* PORTC8 is configured as PTC8 */
    PORT_SetPinMux(PORTC, 8U, kPORT_MuxAsGpio);

    /* PORTC9 is configured as PTC9 */
    PORT_SetPinMux(PORTC, 9U, kPORT_MuxAsGpio);

    /* PORTD1 is configured as PTD1 */
    PORT_SetPinMux(PORTD, 1U, kPORT_MuxAsGpio);
}


/* Problem 3 main Function */
int main(void) {
    /* Define the init structure for the output LED pin*/
    gpio_pin_config_t led_config = {kGPIO_DigitalOutput, 0};

    /* Board pin, clock, debug console init */
    BOARD_InitBootPins();
    BOARD_InitBootClocks();
    BOARD_InitDebugConsole();

    /* Print a note to terminal. */
    PRINTF("\r\n GPIO Driver example\r\n");
    PRINTF("\r\n The LED is blinking.\r\n");

    /* Init output LED GPIO. */
    GPIO_PinInit(BOARD_LED_GPIO_BLUE, BOARD_LED_GPIO_PIN_BLUE, &led_config);
    GPIO_PinInit(BOARD_LED_GPIO_GREEN, BOARD_LED_GPIO_PIN_GREEN, &led_config);
    GPIO_PinInit(BOARD_LED_GPIO_RED, BOARD_LED_GPIO_PIN_RED, &led_config);

    while (1) {
        delay();
        GPIO_PortToggle(BOARD_LED_GPIO_BLUE, 1u << BOARD_LED_GPIO_PIN_BLUE);
        delay();
        GPIO_PortToggle(BOARD_LED_GPIO_BLUE, 1u << BOARD_LED_GPIO_PIN_BLUE);
        delay();
        GPIO_PortToggle(BOARD_LED_GPIO_GREEN, 1u << BOARD_LED_GPIO_PIN_GREEN);
        delay();
        GPIO_PortToggle(BOARD_LED_GPIO_GREEN, 1u << BOARD_LED_GPIO_PIN_GREEN);
        delay();
        GPIO_PortToggle(BOARD_LED_GPIO_RED, 1u << BOARD_LED_GPIO_PIN_RED);
        delay();
        GPIO_PortToggle(BOARD_LED_GPIO_RED, 1u << BOARD_LED_GPIO_PIN_RED);
    }
}
```

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.

```c
/* GPIO Register Structure */
typedef struct {
    uint32_t GPIO_PDOR;
    uint32_t GPIO_PSOR;
    uint32_t GPIO_PCOR;
    uint32_t GPIO_PTOR;
    uint32_t GPIO_PDIR;
    uint32_t GPIO_PDDR;
} GPIO_Struct;

/* Definitions */
#define GPIOA   ((GPIO_Struct*) 0x400FF000)
#define GPIOB   ((GPIO_Struct*) 0x400FF040)
#define GPIOC   ((GPIO_Struct*) 0x400FF080)
#define GPIOD   ((GPIO_Struct*) 0x400FF0C0)
#define GPIOE   ((GPIO_Struct*) 0x400FF100)

/* Problem 4 Helper Functions */
void initPin(GPIO_Struct *port, uint32_t pin) {
    writePin(port, pin, 1);
    port->GPIO_PDDR |= (1UL << pin);
}

void togglePin(GPIO_Struct *port, uint32_t mask) {
    port->GPIO_PDOR ^= (mask);
}

void writePin(GPIO_Struct *port, uint32_t pin, uint8_t output) {
    if (output == 0U) {
        port->GPIO_PDOR |= (1UL << pin);
    } else {
        port->GPIO_PDOR &= ~(1UL << pin);
    }
}

/* Problem 4 main Function */
int main(void) {
    /* Board pin, clock, debug console init */
    BOARD_InitBootPins();
    BOARD_InitBootClocks();
    BOARD_InitDebugConsole();

    /* Print a note to terminal. */
    PRINTF("\r\nGPIO Driver example\r\n");
    PRINTF("\r\nThe LED is blinking.\r\n");

    /* Init output LED GPIO. */
    initPin(BOARD_LED_GPIO_BLUE, BOARD_LED_GPIO_PIN_BLUE);
    initPin(BOARD_LED_GPIO_GREEN, BOARD_LED_GPIO_PIN_GREEN);
    initPin(BOARD_LED_GPIO_RED, BOARD_LED_GPIO_PIN_RED);

    while (1) {
        delay();
        togglePin(BOARD_LED_GPIO_BLUE, 1u << BOARD_LED_GPIO_PIN_BLUE);
        delay();
        togglePin(BOARD_LED_GPIO_BLUE, 1u << BOARD_LED_GPIO_PIN_BLUE);
        delay();
        togglePin(BOARD_LED_GPIO_GREEN, 1u << BOARD_LED_GPIO_PIN_GREEN);
        delay();
        togglePin(BOARD_LED_GPIO_GREEN, 1u << BOARD_LED_GPIO_PIN_GREEN);
        delay();
        togglePin(BOARD_LED_GPIO_RED, 1u << BOARD_LED_GPIO_PIN_RED);
        delay();
        togglePin(BOARD_LED_GPIO_RED, 1u << BOARD_LED_GPIO_PIN_RED);
    }
}
```

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.

```c
void pwm_setup() {
    ftm_config_t ftmInfo;
    ftm_chnl_pwm_signal_param_t ftmParam;

    /* UI_LED_RED PWM Setup */
    ftmParam.chnlNumber = kFTM_Chnl_1;
    ftmParam.level = kFTM_HighTrue;
    ftmParam.dutyCyclePercent = 0;
    ftmParam.firstEdgeDelayPercent = 0U;
    ftmParam.enableComplementary = false;
    ftmParam.enableDeadtime = false;

    FTM_GetDefaultConfig(&ftmInfo);

    FTM_Init(FTM3, &ftmInfo);
    FTM_SetupPwm(FTM3, &ftmParam, 1U, kFTM_EdgeAlignedPwm, 5000U, CLOCK_GetFreq(
        kCLOCK_BusClk));
    FTM_StartTimer(FTM3, kFTM_SystemClock);

    /* UI_LED_BLUE PWM Setup */
    ftmParam.chnlNumber = kFTM_Chnl_4;
    ftmParam.level = kFTM_HighTrue;
    ftmParam.dutyCyclePercent = 0;
    ftmParam.firstEdgeDelayPercent = 0U;
    ftmParam.enableComplementary = false;
    ftmParam.enableDeadtime = false;

    FTM_GetDefaultConfig(&ftmInfo);

    FTM_Init(FTM3, &ftmInfo);
    FTM_SetupPwm(FTM3, &ftmParam, 1U, kFTM_EdgeAlignedPwm, 5000U, CLOCK_GetFreq(
        kCLOCK_BusClk));
    FTM_StartTimer(FTM3, kFTM_SystemClock);

    /* UI_LED_GREEN PWM Setup */
    ftmParam.chnlNumber = kFTM_Chnl_5;
    ftmParam.level = kFTM_HighTrue;
    ftmParam.dutyCyclePercent = 0;
    ftmParam.firstEdgeDelayPercent = 0U;
    ftmParam.enableComplementary = false;
    ftmParam.enableDeadtime = false;

    FTM_GetDefaultConfig(&ftmInfo);

    FTM_Init(FTM3, &ftmInfo);
    FTM_SetupPwm(FTM3, &ftmParam, 1U, kFTM_EdgeAlignedPwm, 5000U, CLOCK_GetFreq(
        kCLOCK_BusClk));
    FTM_StartTimer(FTM3, kFTM_SystemClock);
}
```

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.

```c
int main(void) {
    unsigned int duty_cycle_red = 0;
    unsigned int duty_cycle_green = 0;
    unsigned int duty_cycle_blue = 0;

    /* Init board hardware. */
    BOARD_InitBootPins();
    BOARD_InitBootClocks();
    BOARD_InitDebugConsole();

    pwm_setup();

    scanf("%2x%2x%2x", &duty_cycle_red, &duty_cycle_green, &duty_cycle_blue);

    printf("red = %x\n", duty_cycle_red);
    printf("green = %x\n", duty_cycle_green);
    printf("blue = %x\n", duty_cycle_blue);

    float red = (duty_cycle_red / 255.0) * 100;
    float green = (duty_cycle_green / 255.0) * 100;
    float blue = (duty_cycle_blue / 255.0) * 100;

    FTM_UpdatePwmDutycycle(FTM3, kFTM_Chnl_1, kFTM_EdgeAlignedPwm, (uint8_t)red);
    FTM_SetSoftwareTrigger(FTM3, true);

    FTM_UpdatePwmDutycycle(FTM3, kFTM_Chnl_5, kFTM_EdgeAlignedPwm, (uint8_t)green);
    FTM_SetSoftwareTrigger(FTM3, true);

    FTM_UpdatePwmDutycycle(FTM3, kFTM_Chnl_4, kFTM_EdgeAlignedPwm, (uint8_t)blue);
    FTM_SetSoftwareTrigger(FTM3, true);

    while (1) {
    }
}
```

Listing 5: Problem 5 main