COMPENG 4DS4 Lab 0 Report

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

January 26, 2023

Listing 1: Problem 1

```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)
#define ARBITRARY_LOCATION          *((int*) 0x20001004)

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

Listing 2: Problem 3

```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

/* 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 3: Problem 4

```c
/* 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 4: Problem 5 FTM/PWM Setup

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

  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);

  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);
```

```c

  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 5: Problem 5 main

```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) {
  }
}
```