Prac4B_LPHTUM003

Question a

```
void initPorts()
{
	// ENABLE PUSHBUTTONS
	RCC->AHBENR |= RCC_AHBENR_GPIOAEN;
	// SET PUSHBUTTONS AND POTENTIOMETER 0 AS INPUT
	GPIOA->MODER &= ~ ( GPIO_MODER_MODER0|
					GPIO_MODER_MODER1|
					GPIO_MODER_MODER2|
					GPIO_MODER_MODER3|
					GPIO_MODER_MODER5);
	// SET PUSHBUTTONS PULL UP AND PULL DOWN RESISTORS
	GPIOA->PUPDR |=		( GPIO_PUPDR_PUPDR0_0|
					GPIO_PUPDR_PUPDR1_0|
					GPIO_PUPDR_PUPDR2_0|
					GPIO_PUPDR_PUPDR3_0|
					GPIO_PUPDR_PUPDR5_0);
	// ENABLE LED
	RCC->AHBENR |= RCC_AHBENR_GPIOBEN;
	// SET LED AS OUTPUT
	GPIOB->MODER |=		(	GPIO_MODER_MODER0_0|
					GPIO_MODER_MODER1_0|
					GPIO_MODER_MODER2_0|
					GPIO_MODER_MODER3_0|
					GPIO_MODER_MODER4_0|
					GPIO_MODER_MODER5_0|
					GPIO_MODER_MODER6_0|
					GPIO_MODER_MODER7_0|
					GPIO_MODER_MODER9_0);
}
void init_NVIC(void)
{
	NVIC_EnableIRQ(EXTI0_1_IRQn); //
}

void init_ADC(void)
{
	GPIOA -> MODER |= GPIO_MODER_MODER5;		// MAKE POT0 ANALOGUE INPUT
	RCC -> APB2ENR |= RCC_APB2ENR_ADCEN;		// ENABLE ADC CLOCK
	ADC1 -> CFGR1  |= ADC_CFGR1_RES_1;		// SET 8 BIT RESOLUTION
	ADC1 -> CFGR1  &= ~ADC_CFGR1_ALIGN;		// ALIGN TO THE RIGHT
	ADC1 -> CFGR1  &= ~ADC_CFGR1_CONT;		// USE SINGLE CONVERSION MODE
	ADC1 -> CR     |= ADC_CR_ADEN;			// MAKE ADEN IN CR
	while((ADC1->ISR & ADC_ISR_ADRDY) == 0);	// EXIT WHEN AD IS READY

}void init_EXTI(void)
{
	// ENABLE SYSCFG AND COMP
	RCC->APB2ENR |= RCC_APB2ENR_SYSCFGCOMPEN;
	// CONFIGURE EXTIx IN THE SYSCFG->EXTICRx
	SYSCFG->EXTICR[1] |= SYSCFG_EXTICR1_EXTI1_PA;
	// SET STATE PIN CHANGE ON RISING EDGE
	EXTI->RTSR |=  EXTI_RTSR_TR1;
	// UNMASK EXTI LINE
	EXTI->IMR |=    EXTI_IMR_MR1;
}
```

Question b

$$Output\ Value = \frac{V_{input}(2^n - 1)}{V_{span}}$$

$$Output\ Value = \frac{14(2^3 - 1)}{24}$$

$$Output\ Value = 4.083333V$$

Question c

```
void check_battery(void)
{
	int threshold = 149;
	sample_ADC();
	int ADC_value = ADC1->DR;
	if (ADC_value < threshold)
	{
		GPIOB->ODR = 0b100000000;
	}
	battery_voltage = ADC_value / 10.5;
	lcd_command(CLEAR);
	converttoBCD_voltage();
	lcd_putstring("Battery");
	lcd_command(LINE_TWO);
	lcd_putstring(decimal);
	lcd_putstring(" V");
}
```

Question d

```
void EXTI0_1_IRQHandler(void)
{
	EXTI -> PR |= EXTI_PR_PR0; // clear the interrupt pending bit
	rainfall += 1;
}
```

Question e

```
void display(void)
{
	if ((GPIOA->IDR & SW0) == 0)
	{
		lcd_command(CLEAR);
		lcd_putstring("Weather Station"); // Display string on line 1
		lcd_command(LINE_TWO); // Move cursor to line 2
		lcd_putstring("Press SW2"); // Display string on line 2
	}
	if ((GPIOA->IDR & SW1) == 0)
	{
		Delay();// DELAY FOR +/- 1 SECOND
		count = 1 + count; // INCREASE COUNT BY 1
		GPIOB->ODR = count; // WRITE COUNT VALUE ONTO LED
		lcd_command(CLEAR); // CLEAR SCREEN
		lcd_putstring("Rain bucket tip"); // WRITE ON SCREEN
	}
	if ((GPIOA->IDR & SW2) == 0)
	{
		lcd_command(CLEAR); // CLEAR SCREEN
		converttoBCD_rainfall(); // CONVERT TO BCD
		lcd_putstring("Rainfall:"); // WRITE ON SCREEN
		lcd_command(LINE_TWO); // GO TO LINE 2
		lcd_putstring(decimal); // WRITE DECIMAL NUMBER ON SCREEN
		lcd_putstring(" mm");
	}
	if ((GPIOA->IDR & SW3) == 0)
	{
		check_battery();
	}
}
```

Question f

```
//*****************************************************************

//*            EEE2050F C MAIN                  *

//*================================================================*

//* WRITTEN BY: TUMELO LEPHADI                                     *

//* DATE CREATED: 09/06/2017                      *

//* MODIFIED: 09/06/2017                          *

//*================================================================*
```

```
//* PROGRAMMED IN: Eclipse Luna Service Release 1 (4.4.1)          *
//* DEV. BOARD:    UCT STM32 Development Board                     *
//* TARGET:                          STMicroelectronics STM32F051C6                *
//*================================================================*
//* DESCRIPTION:                                    *
//* PROGRAM THAT MEASURES RAINFALL IN MILIMETRES                   *
//******************************************************************
// INCLUDE FILES
//================================================================
#include "lcd_stm32f0.h"
#include "stm32f0xx.h"
#include "stm32f0xx_adc.h"
#include "stm32f0xx_exti.h"
//================================================================
// SYMBOLIC CONSTANTS
//================================================================
#define SW0             GPIO_IDR_0
#define SW1             GPIO_IDR_1
#define SW2             GPIO_IDR_2
#define SW3             GPIO_IDR_3
#define DELAY1 1092
#define DELAY2 1092
//================================================================
// GLOBAL VARIABLES
//================================================================
int count;
char rainfall, thousands, hundreds, tens, units, remainder, battery_voltage, one_decimal_place, two_decimal_places,
three_decimal_places;
char decimal_rain[10], decimal_volts[10];
//================================================================
// FUNCTION DECLARATIONS
//================================================================
void initPorts(void);
void converttoBCD_rainfall(void);
void converttoBCD_voltage(void);
void Delay(void);
void init_NVIC(void);
void init_EXTI(void);
```

```
void EXTI0_1_IRQHandler(void);
void init_ADC(void);
void check_battery(void);
void sample_ADC(void);
void display(void);
//void init_ADC(void);
//===========================================================================
// MAIN FUNCTION
//===========================================================================
void main (void)
{
	init_LCD();                                                    // Initialise lcd
	initPorts();                                                   // INITIALISE PORTS
	init_EXTI();                                                   // INITIALISE EXTI
	init_ADC();                                                    // INITIALISE ADC
	lcd_putstring("EEE2046F PRAC4B");          // Display string on line 1
	lcd_command(LINE_TWO);                                         // Move cursor to line 2
	lcd_putstring("**LPHTUM003**");                // Display string on line 2
	for(;;)                                                        // Loop forever
	{
		display();
	}
}
// End of main
//===========================================================================
// FUNCTION DEFINITIONS
//===========================================================================
void initPorts()
{
	// ENABLE PUSHBUTTONS
	RCC->AHBENR |= RCC_AHBENR_GPIOAEN;
	// SET PUSHBUTTONS AND POTENTIOMETER 0 AS INPUT
	GPIOA->MODER &= ~ ( GPIO_MODER_MODER0|
                                                GPIO_MODER_MODER1|
                                                GPIO_MODER_MODER2|
                                                GPIO_MODER_MODER3|
                                                GPIO_MODER_MODER5);
	// SET PUSHBUTTONS PULL UP AND PULL DOWN RESISTORS
```

```
	GPIOA->PUPDR |=	( GPIO_PUPDR_PUPDR0_0|
							GPIO_PUPDR_PUPDR1_0|
							GPIO_PUPDR_PUPDR2_0|
							GPIO_PUPDR_PUPDR3_0|
							GPIO_PUPDR_PUPDR5_0);
	// ENABLE LED
	RCC->AHBENR |= RCC_AHBENR_GPIOBEN;
	// SET LED AS OUTPUT
	GPIOB->MODER |=	(		GPIO_MODER_MODER0_0|
							GPIO_MODER_MODER1_0|
							GPIO_MODER_MODER2_0|
							GPIO_MODER_MODER3_0|
							GPIO_MODER_MODER4_0|
							GPIO_MODER_MODER5_0|
							GPIO_MODER_MODER6_0|
							GPIO_MODER_MODER7_0|
							GPIO_MODER_MODER9_0);
}
void converttoBCD_rainfall(void)
{
	double rain = rainfall;
	rain = rain * 0.2;
							// 0.2 mm PER BUCKET
	thousands = (rain / 1000) ;
					// FIND THOUSANDS
	hundreds = ((rain - thousands * 1000) / 100);
		// FIND HUNDREDS
	tens = (rain - thousands * 1000 - hundreds * 100) / 10;                                                    // FIND TENS
	units = (rain - thousands * 1000 - hundreds * 100 - tens * 10) / 1;                        // FIND UNITS
	remainder = 10 * (rain - thousands * 1000 - hundreds * 100 - tens * 10 - units);// FIND DECIMAL NUMBERS
	decimal_rain[0] = thousands + 48;
					// INSERT ASCII NUMBERS INTO ARRAY
	decimal_rain[1] = hundreds + 48;
	decimal_rain[2] = tens + 48;
	decimal_rain[3] = units + 48;
	decimal_rain[4] = 46;
							// INSERT ASCII VALUE FOR DECIMAL POINT
	decimal_rain[5] = remainder + 48;
}
void converttoBCD_voltage(void)
```

```
{
	double volts = battery_voltage;
	tens = volts / 10;
					// FIND TENS
	units = (volts - tens * 10) / 1;
	// FIND UNITS
	one_decimal_place = ( 1000 * ( volts - tens * 10 - units ) ) / 100;
							// FIND FIRST DECIMAL NUMBER
	two_decimal_places = ( 1000 * (volts - tens * 10 - units - one_decimal_place / 10 ) ) / 10;
					// FIND SECOND DECIMAL NUMBER
	three_decimal_places = ( 1000 * (volts - tens * 10 - units - one_decimal_place / 10 - two_decimal_places / 100 ) ) / 1; // FIND
THIRD DECIMAL NUMBER
	decimal_volts[0] = tens + 48;
				// INSERT ASCII NUMBERS INTO ARRAY
	decimal_volts[1] = units + 48;
	decimal_volts[2] = 46;
					// INSERT ASCII VALUE FOR DECIMAL POINT
	decimal_volts[3] = one_decimal_place + 48;
	decimal_volts[4] = two_decimal_places + 48;
	decimal_volts[5] = three_decimal_places + 48;
}
void Delay(void)
{
	// INITIALIZE I AND J
	int i, j;
	for (i = 1; i < DELAY1; i++)
		for (j = 1; j < DELAY2; j++) // DELAY FOR 1 SECOND
		{};
}
void init_NVIC(void)
{
	NVIC_EnableIRQ(EXTI0_1_IRQn);//CLEAR INTERRUPT PENDING BIT
}
void init_EXTI(void)
{
	// ENABLE SYSCFG AND COMP
	RCC->APB2ENR |= RCC_APB2ENR_SYSCFGCOMPEN;
	// CONFIGURE EXTIx IN THE SYSCFG->EXTICRx
```

```
	SYSCFG->EXTICR[0] |= SYSCFG_EXTICR1_EXTI0;
	// SET STATE PIN CHANGE ON RISING EDGE
	EXTI->RTSR |=		EXTI_RTSR_TR0;
	// UNMASK EXTI LINE
	EXTI->IMR |=	EXTI_IMR_MR0;
}
// ISR
void EXTI0_1_IRQHandler(void)
{
	EXTI -> PR |= EXTI_PR_PR0; // clear the interrupt pending bit
	rainfall += 1;
}
void init_ADC(void)
{
	GPIOA -> MODER	|= GPIO_MODER_MODER5;		// MAKE POT0 ANALOGUE INPUT
	RCC -> APB2ENR	|= RCC_APB2ENR_ADCEN;		// ENABLE ADC CLOCK
	ADC1 -> CFGR1	|= ADC_CFGR1_RES_1;		// SET 8 BIT RESOLUTION
	ADC1 -> CFGR1	&= ~ADC_CFGR1_ALIGN;		// ALIGN TO THE RIGHT
	ADC1 -> CFGR1	&= ~ADC_CFGR1_CONT;		// USE SINGLE CONVERSION MODE
	ADC1 -> CR |=		ADC_CR_ADEN;			// MAKE ADEN IN CR
	while((ADC1->ISR & ADC_ISR_ADRDY) == 0); // EXIT WHEN AD IS READY

}
void sample_ADC(void)
{
	ADC1->CHSELR		|=		ADC_CHSELR_CHSEL5;//CONNECT TO POTENTIOMETER AT PA5
	// SET ADSTART IN CR
	ADC1->CR		|=		ADC_CR_ADSTART;
	while(ADC_ISR_EOC==0); //WAIT UNTIL EOC IS INTERRUPTED
}
void check_battery(void)
{
	int threshold = 149;
	sample_ADC();
	int ADC_value = ADC1->DR;
	if (ADC_value < threshold)
	{
		GPIOB->ODR = 0b100000000;
```

```
		}
		battery_voltage = ADC_value / 10.5;
		lcd_command(CLEAR);
		converttoBCD_voltage();
		lcd_putstring("Battery");
		lcd_command(LINE_TWO);
		lcd_putstring(decimal_volts);
		lcd_putstring(" V");
}
void display(void)
{
		if ((GPIOA->IDR & SW0) == 0)
		{
			lcd_command(CLEAR);
			lcd_putstring("Weather Station"); // Display string on line 1
			lcd_command(LINE_TWO); // Move cursor to line 2
			lcd_putstring("Press SW2"); // Display string on line 2
		}
		if ((GPIOA->IDR & SW1) == 0)
		{
			Delay();// DELAY FOR +/- 1 SECOND
			rainfall = 1 + rainfall; // INCREASE COUNT BY 1
			GPIOB->ODR = rainfall; // WRITE COUNT VALUE ONTO LED
			lcd_command(CLEAR); // CLEAR SCREEN
			lcd_putstring("Rain bucket tip"); // WRITE ON SCREEN
		}
		if ((GPIOA->IDR & SW2) == 0)
		{
			lcd_command(CLEAR); // CLEAR SCREEN
			converttoBCD_rainfall(); // CONVERT TO BCD
			lcd_putstring("Rainfall:"); // WRITE ON SCREEN
			lcd_command(LINE_TWO); // GO TO LINE 2
			lcd_putstring(decimal_rain); // WRITE DECIMAL NUMBER ON SCREEN
			lcd_putstring(" mm");
		}
		if ((GPIOA->IDR & SW3) == 0)
		{
			check_battery();
```

```
            }
}
//*********************************************************************
// END OF PROGRAM
//*********************************************************************
```