PROJECT 6 PDF

Readme For project 6 Team Members :

1)Suraj Thite (suraj.thite@colorado.edu)

2)Atharv Desai (atharv.desai@colorado.edu)

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

The below enumerated files are contained in the repository

1) PES_Project6

i) Project Settings

ii) Build Targets

iii) Binaries

iv) Includes

v) CMSIS

vi) Board

vii) Drivers

viii) Freertos

viii) Source

a) main.c & main.h

b) logger.c & logger.h

c) RGBled.c & RGBled.h

d) adc_dac.c & adc_dac.h

e) wave.c & wave.h

f) circularbuff.c & circularbuff.h

g) timestamp.c & timestamp.h

h) dma.c & dma.h

i) task1.c & btasks.h

ix) Startup

x) Utilities

xi) Debug

xii) PESProject6 PE Debug.launch

2) Readme File (Readme Markdown File)

3) Sine wave on OScillloscope Screenshot

INSTALLATION & EXECUTION NOTES

The code is tested on the enviornment below:

* MCUXpresso IDE which is an easy to use Eclipse-based development environment for NXP® MCUs based on Arm® Cortex®-M cores.

* During this project, this IDE was used to code,execute circular buffer functions and interface them with adc and dac pins i.e PTE 20 and PTE 30 of FRDM board on FRDM-KL25z and observe output on the oscilloscope.

* Also, we were able to check and verify the workability of the DMA by comparing the ADC and DSP buffer values in the project.

* Using DSO-X-2022A, the PTE30 pin data in the form of sine waveform was observed on the DSO for 5sec peak to peak and amplitude of 2V.

* The FreeRtos Functions such as VtaskDelay and many others were read from the reference manual on the FreeRtos Website.

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

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

* 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.

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

* Set #define to 1 (ECHO) or 0 (APPLICATION) respectively.

* Set modes to Test, Debug or Status mode by setting the vale for variable 'a' in logger.c file accordingly.

DIFFICULTIES & OBSERVATIONS:

* While capturing the Sine wave output on Oscilloscope, we faced issues in capturing the time difference between two peaks using for loop. Therefore, we learned to use the vtaskdelay function and pass appropriate macro value as an argument to that function to get .

* While designing the circular buffer functions, we were not able to reallocate a new buffer while retaining the old data and adding new data further.

* While implementing the tasks in FreeRtos, we studied the working of schedular while giving priorities to the functions such as DaC, ADC etc and learnt about the operating system concepts such as mutex and semaphores.

* While implementing logger earlier, the enum values were being passed as arguments in integer format only. But on accessing them in other .c files using extern keyword, the issue was resolved. Also, integrated the timestamp with the logger and segregated the count value between hours, minutes and seconds to get the timestamp in appropriate format.

FINAL CODE:

```
/*
 * main.h
 *
 *  Created on: Dec 5, 2019
 *      Author: SURAJ THITE
 */

#ifndef MAIN_H_
#define MAIN_H_


#define APPLICATION 0


#endif /* MAIN_H_ */


/*
 * Copyright (c) 2015, Freescale Semiconductor, Inc.
 * Copyright 2016-2017 NXP
 *
 * Redistribution and use in source and binary forms, with or without modification,
 * are permitted provided that the following conditions are met:
 *
 * o Redistributions of source code must retain the above copyright notice, this list
 *   of conditions and the following disclaimer.
 *
 * o Redistributions in binary form must reproduce the above copyright notice, this
 *   list of conditions and the following disclaimer in the documentation and/or
 *   other materials provided with the distribution.
 *
 * o Neither the name of the copyright holder nor the names of its
 *   contributors may be used to endorse or promote products derived from this
 *   software without specific prior written permission.
 *
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
 * WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
 * DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
 * ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
 * (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
 * ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
 * (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
 * SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
 */

/* FreeRTOS kernel includes. */
#include "FreeRTOS.h"
#include "task.h"
```

```c
#include "queue.h"
#include "timers.h"
#include "dma.h"
#include "circularbuff.h"
#include "main.h"
#include "logger.h"
/* Freescale includes. */

#include "fsl_device_registers.h"
#include "fsl_debug_console.h"
#include "board.h"

#include "pin_mux.h"
#include "wave.h"

#include "adc_dac.h"

#include "tasks.h"

#include "semphr.h"
#include "RGBled.h"
#include "time_stamp.h"
//cbuff *adc_buffer;

/*******************************************************************************
 * Definitions
 ******************************************************************************/
/* Task priorities. */
//#define hello_task_PRIORITY (configMAX_PRIORITIES - 1)

TimerHandle_t DAC_Timer_Handle = NULL;
extern SemaphoreHandle_t led_mutex;
extern uint64_t current_time;
extern modes a;
/*******************************************************************************
 * Prototypes
 ******************************************************************************/
QueueHandle_t ADC_BUFF;
void DAC_write_task(TimerHandle_t xTimer);
/*******************************************************************************
 * Code
 ******************************************************************************/
/*!
 * @brief Application entry point.
 */


int main(void)
{

	BOARD_InitBootPins();
	BOARD_InitBootClocks();
	BOARD_InitBootPeripherals();
	BOARD_InitDebugConsole();
```

```
	initialize_dac();

	initialize_adc();

	dma_init();

	RGB_init();

	RGB_OFF();

	sine_lookup_generate();
#if APPLICATION ==0
	//	initialize_dac();

	//DAC_Timer_Handle = xTimerCreate("DAC_Write_Timer", (1000 /
portTICK_PERIOD_MS), pdTRUE, 0, DAC_write_task  );
	DAC_Timer_Handle = xTimerCreate("DAC_Write_Timer", 100, pdTRUE, 0,
DAC_write_task  );
	xTimerStart(DAC_Timer_Handle, 0);
	vTaskStartScheduler();

	while(1)
	{

	}

#else
	ADC_BUFF = xQueueCreate(64,sizeof(uint16_t));

	PRINTF("\n \r xQueue Create Initialized Initialized");

	led_mutex = xSemaphoreCreateMutex();

	//xTaskCreate(TimerUpdate,( portCHAR *)"UpdateTimervalue",
configMINIMAL_STACK_SIZE, NULL, 2, NULL);

	xTaskCreate(dac_task,( portCHAR *)"dactask", configMINIMAL_STACK_SIZE, NULL,
1, NULL);



	xTaskCreate(adc_task,( portCHAR *)"readadc", configMINIMAL_STACK_SIZE, NULL,
0, NULL);

	vTaskStartScheduler();


	while (1)
	{

	}
#endif
}
```

```
/*****************************************************************************
******************
 * Function Name:void DAC_write_task(TimerHandle_t xTimer)
 * Description :This is a callback task function to write generated sine wave values
into the DAC pin
 * @input:  TimerHandle_t xTimer
 * @Return : void

*****************************************************************************
*****************/
void DAC_write_task(TimerHandle_t xTimer)
{
	current_time++;
	uint16_t val = get_next_val();
//	PRINTF(" \n \r Writing %d to the DAC.", val);
	led_switch(0);
	dac_write(val);
	RGB_OFF();
if(a==Debug)
	Log_String(a, Main, "Value Written to DAC");
}


/*
 * adc_dac.h
 *
 *  Created on: Dec 1, 2019
 *      Author: SURAJ THITE
 */

#ifndef ADC_DAC_H_
#define ADC_DAC_H_

#include "board.h"
#include "fsl_dac.h"
#include "fsl_adc16.h"

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

void initialize_dac();

void initialize_adc();

void dac_write(uint16_t val);

uint32_t adc_read();
#endif /* ADC_DAC_H_ */


/*
 * adc_dac.c
 *
 *  Created on: Dec 1, 2019
 *      Author: SURAJ THITE
```

```
 */

#include "adc_dac.h"

#include "fsl_debug_console.h"

static adc16_config_t adc_config_struct;      //Structure to store the ADC
configuration

static adc16_channel_config_t adc_channel_config_struct;   //Structure to store
channel configuration of the ADC

static dac_config_t dac_config;         //Structure to store the DAC configuration of
the DAC

/*************************************************************************************
******************
 * Function Name:void initialize_dac()
 * Description :This function Initializes and enables the DAC on default
configuration.
 * @input:  pointer to uint8_t
 * @Return : pointer

*************************************************************************************
******************/
//Reference from SDK example to initialize the DAC
void initialize_dac()
{
	DAC_GetDefaultConfig(&dac_config);     //Get Default configuration of DAC

	DAC_Init(DAC0, &dac_config);     //INitialize the DAC with default
configuration

	DAC_Enable(DAC0,1); //ENable the DAC peripheral

	DAC_SetBufferReadPointer(DAC0, 0U);     //Set buffer read pointer to starting
location

	PRINTF("*********DAC INITIALIZED**********");
}

/*************************************************************************************
******************
 * Function Name:void initialize_adc()
 * Description :This function Initializes the ADC to default configuration. The
AutoCalibration Feature is
 * set on while interrupts is disabled upon conversion of the ADC value. Channel no 0
is selected for input.
 * @input:  pointer to uint8_t
 * @Return : pointer

*************************************************************************************
******************/
void initialize_adc()
{
```

```
	ADC16_GetDefaultConfig(&adc_config_struct);   //Get default configuration of
the ADC

	ADC16_Init(ADC0, &adc_config_struct);  //Initialize the ADC with Default
configuration

	ADC16_EnableHardwareTrigger(ADC0, false);     //Enable hardware trigger for
ADC

	 ADC16_DoAutoCalibration(ADC0);  //Enable Auto Calibration for the ADC0

	adc_channel_config_struct.channelNumber = 0;  //Select channel number for ADC


	adc_channel_config_struct.enableInterruptOnConversionCompleted = false;
	//Disable ADC interrupts
	PRINTF("\n \r *********ADC INITIALIZED**********");
}


/*************************************************************************************
******************
 * Function Name:void dac_write(uint16_t val)
 * Description :This function Writes the values to the DAC buffer
 * @input:  uint16_t value to be written
 * @Return : void

*************************************************************************************
******************/
void dac_write(uint16_t val)
{
	DAC_SetBufferValue(DAC0,0,val);  //Set the DAC buffer value to the input value
recieved
//	PRINTF("\n \r*********VALUE WRITTEN TO DAC**********");
}

/*************************************************************************************
******************
 * Function Name:uint32_t adc_read()
 * Description :This function Reads the ADC value from the buffer and returns the
value to the calling function
 * @input:  void
 * @Return :uint32_t value from ADC

*************************************************************************************
******************/
uint32_t adc_read()
{
	ADC16_SetChannelConfig(ADC0, 0, &adc_channel_config_struct);	//Set channel
to 0 and initialize with channel struct config
	while ((ADC16_GetChannelStatusFlags(ADC0,0) &
kADC16_ChannelConversionDoneFlag)==0)  //Wait for channel convertion flag to set
	{
```

```
	}
	return ADC16_GetChannelConversionValue(ADC0,0);	//Return the ADC value to
the called function

}


/*
 * circularbuff.h
 *
 *  Created on: Nov 9, 2019
 *      Author: SURAJ THITE
 */

#ifndef CIRCULARBUFF_H_
#define CIRCULARBUFF_H_

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

typedef struct
{
	uint32_t *cbuffptr;
	uint32_t *newcbuffptr;
	uint32_t *head;
	uint32_t *tail;
    uint16_t size;
    uint8_t count;
}cbuff;


typedef enum
{
    cbuff_init_success,
    cbuff_init_fail,
    cbuff_empty,
	wrap_add,
	wrap_remove,
    cbuff_not_empty,
    cbuff_full,
    cbuff_not_full,
    null_ptr,
    buffer_NA,
    cbuff_success,
	buffer_init_failed,
	buffer_init_success,
	ptr_valid,
	ptr_invalid,
	destroy_failed,
	destroy_pass
}cbuff_status;

cbuff_status cbuff_init(cbuff *ptr, uint16_t length);
cbuff_status cbuff_add(cbuff *ptr, uint8_t val);
```

```
cbuff_status cbuff_delete(cbuff *ptr, uint8_t *val);
cbuff_status cbuff_isempty(cbuff *ptr);
cbuff_status cbuff_check_full(cbuff *ptr);
cbuff_status cbuff_resize(cbuff *ptr,uint8_t length);
void cbuff_print(cbuff* ptr);
cbuff_status verify_ptr(uint32_t *ptr1,cbuff *ptr);
cbuff_status verify_init(cbuff* ptr);
cbuff_status cbuff_destroy(cbuff* ptr);
void cbuff_reset(cbuff* ptr);

#endif /* CIRCULARBUFF_H_ */


/*
 * circularbuff.c
 *
 *  Created on: Nov 17, 2019
 *      Author: SURAJ THITE & Atharv Desai
 */
#include "circularbuff.h"
#include "fsl_debug_console.h"
#include "logger.h"
extern modes a;
/**************************************************************************************
******************
 * Function Name:cbuff_status cbuff_init(cbuff *ptr, uint16_t length)
 * Description :This Function Initializes the circular buffer
 * @input: pointer to circular buffer and length of the circular buffer
 * @Return : error status messages

**************************************************************************************
*****************/

cbuff_status cbuff_init(cbuff *ptr, uint16_t length)
{

	if(ptr==NULL || length <=0)
	{
		return null_ptr;
	}
	else
	{
		if((ptr->cbuffptr)==NULL)
		{
			ptr->head=NULL;     //Initialize  head pointer to NULL
			ptr->tail=NULL;     //Initialize tail pointer to NULL
			ptr->count=0;//Initialize count to zero
			ptr->size=0; //Initialize size to zero
			return cbuff_init_fail;   //Return Fail status message
		}
		else
		{
			ptr->cbuffptr= (uint32_t*)malloc(sizeof(uint32_t)*length);
	//Allocate memory for the circular buffer
			ptr->size= length;  //Set size of the buffer
```

```
                ptr->count=0;//Initialize count to zero
                ptr->head=ptr->cbuffptr;  //Set head to base address
                ptr->tail=ptr->cbuffptr;  //Set tail to base address
                return  cbuff_init_success;      //Return status
            }
        }
}

/**************************************************************************************
******************
 * Function Name:cbuff_status cbuff_check_full(cbuff *ptr)
 * Description :This Function checks whether circular buffer is full or not
 * @input: pointer to circular buffer
 * @Return : error status messages

**************************************************************************************
*****************/

cbuff_status cbuff_check_full(cbuff *ptr)
{
        if(ptr==NULL)
        {
                return null_ptr;    //Return Status
        }
        else if ((ptr->cbuffptr)==NULL)
        {
                return buffer_NA;   //Return Status
        }
        else if ((ptr->count)==(ptr->size))
        {

                return cbuff_full;  //Return Status
        }
        else
        {
                return cbuff_not_full; //Return Status
        }
}

/**************************************************************************************
******************
 * Function Name:cbuff_status cbuff_isempty(cbuff *ptr)
 * Description :This Function checks whether circular buffer is empty or not
 * @input: pointer to circular buffer
 * @Return : error status messages

**************************************************************************************
*****************/
cbuff_status cbuff_isempty(cbuff *ptr)
{
        if(ptr==NULL)
        {

                return null_ptr;          //Return Status
        }
```

```
	else if ((ptr->cbuffptr)==NULL)
	{
		return buffer_NA;   //Return Status
	}
	else if ((ptr->count)==0)
	{

		return cbuff_empty;	//Return Status
	}
}

/**************************************************************************************
*******************
 * Function Name:cbuff_status cbuff_add(cbuff *ptr, uint8_t val)
 * Description :This Function adds the value to the address pointed by the head of
the circular buffer
 * @input: pointer to circular buffer and value to be added
 * @Return : error status messages

**************************************************************************************
******************/

cbuff_status cbuff_add(cbuff *ptr, uint8_t val)
{
	if(ptr==NULL)
	{
		return null_ptr;    //Return Status
	}
	else if ((ptr->cbuffptr)==NULL)
	{
		return buffer_NA;   //Return Status
	}
	else if (cbuff_check_full(ptr)==cbuff_full)
	{
		return cbuff_full;	//Return Status
	}
	else if(ptr->head==((ptr->cbuffptr)+((ptr->size)-1)))	//handle Wrap add
Condition
	{
		*(ptr->head)= val;  //Store the value at the address pointed by the head

		printf("WRAP ADD ::New Item Inserted at position %x location :: %d \r
\n",ptr->head,*ptr->head);
		ptr->head=ptr->cbuffptr;  //Initialize the head to base address of the
cbuff pointer
		ptr->count++;//Increment count
		return wrap_add;    //Return Status
	}
	else
	{
		*(ptr->head)=val;//Store the value at the address pointed by the head
		printf("New Item Inserted at position %d location :: %d  \r \n",ptr-
>head,*ptr->head);
		ptr->head++; //Increment the head pointer
		ptr->count++;//Increment the count
```

```
		return cbuff_success;		//Return Status
	}
}

/**************************************************************************************
******************
 * Function Name:cbuff_status cbuff_delete(cbuff *ptr, uint8_t *val)
 * Description :This Function deletes  the value to the address pointed by the tail
of the circular buffer
 * @input: pointer to circular buffer and pointer to a location where the removed
value to be stored
 * @Return : error status messages

**************************************************************************************
******************/
cbuff_status cbuff_delete(cbuff *ptr, uint8_t *val)
{
	if(ptr==NULL)
	{

		return null_ptr;	//Return Status
	}
	else if ((ptr->cbuffptr)==NULL)
	{

		return buffer_NA;	//Return Status
	}
	else if (cbuff_isempty(ptr)==cbuff_empty)
	{
		return cbuff_empty; //Return Status
	}
	else if(ptr->tail==((ptr->cbuffptr)+((ptr->size)-1)))		//Handle Wrap
remove condition
	{
		*(val)=*(ptr->tail);		//Store the value pointed by the tail to a
memory address
		ptr->tail=ptr->cbuffptr;	//point tail to the base address of the
circular buffer
		ptr->count--;//Decrement count
		if(a==0 || a==1)
			//Log_String(a,cbuffdelete,"Wrap- Deleted");  //T

		return wrap_remove; //Return Status
	}
	else
	{
		*(val)=*(ptr->tail);		//Store the value pointed by the tail to a
memory address
		ptr->tail++; //Increment the tail address
		ptr->count--;//Decrement count
		if(a==0 || a==1)
			//Log_String(a,cbuffdelete,"Deleted");  //T

		return cbuff_success;		//Return status
	}
```

```
}
/*****************************************************************************
******************
 * Function Name:cbuff_status verify_init(cbuff* ptr)
 * Description :This Function verifies whether a circular buffer is initialized or
not by checking the pointer
 * @input: pointer to circular buffer.
 * @Return : error status messages

*****************************************************************************
******************/
cbuff_status verify_init(cbuff* ptr)
{
	if(ptr->cbuffptr==NULL)
	{
		return buffer_init_failed;//Return status
	}
	else
	{
		return buffer_init_success;	//Return Status
	}
}

/*****************************************************************************
******************
 * Function Name:cbuff_status verify_ptr(cbuff *ptr)
 * Description :This Function verifies whether a pointer is within the range of
circular buffer
 * @input: pointer to circular buffer.
 * @Return : error status messages

*****************************************************************************
******************/

cbuff_status verify_ptr(uint32_t *ptr1,cbuff *ptr)
{
	if(ptr1 >= ptr->cbuffptr && ptr1 <= ptr->head )	//Check whether passed
pointer is in the range of circular buffer
	{

		return ptr_valid;  //return status
	}
	else
	{
		return ptr_invalid;	//Return Status
	}

}

/*****************************************************************************
******************
 * Function Name:cbuff_status cbuff_resize(cbuff *ptr,uint8_t length)
 * Description :This Function resizes the circular buffer to the size of length
passed as parameter implementing realloc function
```

```
 * @input: pointer to circular buffer and new length.
 * @Return : error status messages

*********************************************************************************
*****************/

cbuff_status cbuff_resize(cbuff *ptr,uint8_t length)
{
	if(ptr==NULL)
	{
		return null_ptr;	//Return Status
	}
	else if ((ptr->cbuffptr)==NULL)
	{
		return buffer_NA;	//Return Status
	}
	else
	{
		if(a==0 || a==1)
			//Log_String(a,cbuffresize,"*** EXTRA CREDIT: BUFFER
RESIZED***");//T
		ptr->newcbuffptr=(uint32_t *)realloc(ptr-
>cbuffptr,sizeof(uint32_t)*length);	//Reallocate the memory
		ptr->cbuffptr=ptr->newcbuffptr;  //Set pointer value to new memory
location pointed by newcbuff pointer
		//ptr->head =ptr->newcbuffptr;
		ptr->size=length;	//Set size to length passed
		ptr->count =0;			//Rest count to zero
		return cbuff_success;		//Return Status
	}
}

/*********************************************************************************
******************
 * Function Name:cbuff_status cbuff_resize(cbuff *ptr,uint8_t length)
 * Description :This Function prints the elements in the circular buffer along with
its location
 * @input: pointer to circular buffer.
 * @Return : void

*********************************************************************************
*****************/

void cbuff_print(cbuff* ptr)
{
	uint32_t *temp = ptr->tail;	//Temporary pointer to store address of Out
location(Tail) of the circular buffer
	for (int i=0;i<ptr->count;i++)
	{
		printf(" \r \n value at position %x location :: %d ",temp,*temp);
	//Print the elements of circular buffer
		temp++;		//Increment pointer
	}
}
```

```
/*****************************************************************************
******************
 * Function Name:cbuff_status cbuff_destroy(cbuff* ptr)
 * Description :This Function destroys the memory allocated for circular buffer
 * @input: pointer to circular buffer and new length.
 * @Return : error status messages

*****************************************************************************
*****************/


cbuff_status cbuff_destroy(cbuff* ptr)
{
	if(ptr->cbuffptr==NULL)
	{
		return destroy_failed;	//Return Status
	}
	else
	{
		free(ptr->cbuffptr);	//Free memory allocated to cbuff pointer

		return destroy_pass;	//Return Status
	}
}


void cbuff_reset(cbuff* ptr)
{
	ptr->count=0;//Initialize count to zero
	ptr->head=ptr->cbuffptr;//Set head to base address
	ptr->tail=ptr->cbuffptr;//Set tail to base address
}


/*
 * dma.h
 *
 *  Created on: Dec 1, 2019
 *      Author: SURAJ THITE
 */

#ifndef DMA_H_
#define DMA_H_

#include "fsl_dma.h"
#include "fsl_dmamux.h"
#include "tasks.h"
#include "circularbuff.h"
#include "fsl_debug_console.h"

//void dma_transfer(uint8_t *srcAddr,uint32_t *destAddr , uint8_t no_of_words);
void dma_transfer(uint8_t srcAddr,uint32_t *destAddr , uint8_t no_of_words);
//void DMA_Callback(dma_handle_t *handle, void *param);
void dma_init();
```

```
#define BUFF_LENGTH 4
#define DMA_CHANNEL 0
#define DMA_SOURCE 63

#endif /* DMA_H_ */


/*
 * dma.c
 *
 *  Created on: Dec 1, 2019
 *      Author: SURAJ THITE
 */

#include "dma.h"


dma_transfer_config_t transferConfig;  //Structure to store the DMA configuration


//bool g_Transfer_Done = false;

dma_handle_t g_DMA_Handle;//Handle for DMA transfer

/*************************************************************************************
******************
 * Function Name:void dma_init()
 * Description :This function initializes the DMA on Zero Channel and DMAMUX0 for
shot transfer mode
 * @input:  pointer to uint8_t
 * @Return : pointer
 * Reference: DMA Transfer example from SDK

*************************************************************************************
******************/
void dma_init()
{
	//Configure MUX for DMA
	DMAMUX_Init(DMAMUX0);	//Initialize the DMA Multiplexer
	DMAMUX_SetSource(DMAMUX0, DMA_CHANNEL, DMA_SOURCE); //Set source for and
channel for DMA transfer
	DMAMUX_EnableChannel(DMAMUX0, DMA_CHANNEL); //Enable DMA

	//COnfigure DMA for Shot transfer
	DMA_Init(DMA0);	//Initialize the DMA peripheral

	PRINTF("\n \r ***********DMA Initialized********** \n \r");

}


/* User callback function for DMA transfer. */
//void DMA_Callback(dma_handle_t *handle, void *param)
//{
//	g_Transfer_Done = true;
```

```
//}
/*************************************************************************************
******************
 * Function Name:void dma_transfer(uint8_t srcAddr,uint32_t *destAddr , uint8_t
no_of_words)
 * Description :This function transfers the data from source to destination passed as
arguments.
 * @input:  source address,destination address ,words to transfer.
 * @Return : pointer

*************************************************************************************
******************/
void dma_transfer(uint8_t srcAddr,uint32_t *destAddr , uint8_t no_of_words)
{
	DMA_CreateHandle(&g_DMA_Handle, DMA0, DMA_CHANNEL); //Create the DMA handle
associated with DMA0 and DMA_channel
	DMA_SetCallback(&g_DMA_Handle, DMA_Callback, NULL); //Set callback function
for DMA complete.
	DMA_PrepareTransfer(&transferConfig, srcAddr, 2 , destAddr, 2 , 2*no_of_words,
kDMA_MemoryToMemory);	//Prepare transfer
	DMA_SubmitTransfer(&g_DMA_Handle, &transferConfig, kDMA_EnableInterrupt);
	//Submit transfer configuration to the DMA handle
	DMA_StartTransfer(&g_DMA_Handle);	//Start DMA Transfer
//	while (g_Transfer_Done != true)
//	{
//
//	}
	PRINTF("\n \r ************DMA TRANSFER SUCCESSFUL**************");
}

#ifndef LOGGER_H_
#define LOGGER_H_

///////////////ENUM for test, debug and status//////////////////
typedef enum
{
	Test,
	Debug,
	Normal
}modes;


/////////////////////////////////////////////////////////

typedef enum
{
	dactask,
	adctask,
	startdsp,
	dmacallback,
	Main

}fnnames;

void Log_enable();
```

```
void Log_disable();
uint8_t Log_status();
void Log_data (uint32_t *, uint32_t );
void Log_String(uint8_t ,uint8_t,char *str);
void Log_integer(uint8_t ,int16_t);
uint8_t Log_level();

#endif


/*
 * logger.c
 *
 *  Created on: Nov 17, 2019
 *      Author: SURAJ THITE & Atharv Desai
 */
/*
 * logger.c
 *
 *  Created on: Nov 17, 2019
 *      Author: SURAJ THITE & Atharv Desai
 */
#include <stdio.h>
#include <stdlib.h>
#include <stdint.h>
#include "logger.h"

#include "circularbuff.h"
#include "time_stamp.h"
#include "fsl_debug_console.h"


modes a = Debug;   // setting mode

fnnames fn_name;

////////TABLE for modes and current states//////////////////////////////

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(modes current_mode,int16_t intval)
{
      if(current_mode != 1)      // since no integers to print in normal status mode
      //printf("%d ",intval); // Print the  data
      PRINTF("%d",intval);
```

```
}


/*****************************************************************************
******************
 * 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(uint8_t current_mode,fnnames mycurrent_function, char *str)
{
	time_stamp_print();	//Print time stamp
	if (current_mode ==0)
	{
	   PRINTF("\t Test Mode: ");
	}
	 if (current_mode ==1)
	{
		PRINTF("\t Debug Mode: ");
	}
	 if (current_mode ==2)
	{
		PRINTF("\t Normal Mode: ");
	}


	 PRINTF(" %s ",str);


	if (mycurrent_function==dactask)
	{
		PRINTF("\t Function: DACtask \n");
	}
	else if (mycurrent_function==adctask)
	{
		PRINTF("\t Function: ADCtask \n");
	}
	else if (mycurrent_function==startdsp)
	{
		PRINTF("\t Function: Start Dsp \n");
	}
	else if (mycurrent_function==dmacallback)
	{
		PRINTF("\t Function: DMA Callback \n");
	}
	else if (mycurrent_function==Main)
	{
		PRINTF("\t Function: Main \n");
	}
```

```
}

///////////////Log level//////////////////////////
uint8_t Log_level()
{

      return a;
}



/*
 * 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
void delay(int time_ms); // Delay
#endif /* RGBLED_H_ */


/*
 * 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 "main.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);
#if APPLICATION ==1
            delay(100);
#else
            delay(10);
#endif
            }
            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(100);
```

```
	}
			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(100);
	}
			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: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 */
	}
}

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

```
                if(iterations > 4)
                {
                  vTaskEndScheduler();      //If 4 iterations are done then , End
the task scheduler
                }
                vTaskSuspend( NULL ); //Suspend task and service next task in the
Queue
            }

}

/*************************************************************************************
******************
 * Function Name:void DMA_Callback(dma_handle_t *handle, void *param)
 * Description :This is a callback function to indicate the successful transfer and
set the flag respectively.
 * @input:  dma handle pointer , void pointer
 * @Return : void

*************************************************************************************
******************/
void DMA_Callback(dma_handle_t *handle, void *param)
{
      g_Transfer_Done = true;    //Set a flag to indicate thw DMA transfer is
complete.
}

/*
 * time_stamp.h
 *
 *  Created on: Nov 17, 2019
 *      Author: SURAJ THITE
 */

#ifndef TIME_STAMP_H_
#define TIME_STAMP_H_

#include "circularbuff.h"

void Init_SysTick(void);
void SysTick_Handler();
uint64_t get_current_time();
uint64_t time_passed(uint64_t since);
void time_stamp_print();

#endif /* TIME_STAMP_H_ */


/*
 * timer_stamp.c
 *
 *  Created on: Nov 17, 2019
 *      Author: SURAJ THITE ,Atrharv Desai
 */
```

```c
#include "time_stamp.h"
#include "fsl_debug_console.h"
#include "MKL25Z4.h"
uint64_t current_time = 0;
static const uint64_t time_max = ~0;
/*!
 * Function Name:Init_SysTick(void)
 * Description :This function Initializes the SysTick Timer for 0.1 second interrupt.
 * @input: void
 * @Return : Void
 */
void Init_SysTick(void)
{
	SysTick->LOAD = (48000000L/100); //Initialize Load value
	NVIC_SetPriority(SysTick_IRQn,3);		//Enable NVIC Interrupt with priority 3
	NVIC_EnableIRQ(SysTick_IRQn);	//Enable NVIC IRQ
	SysTick->VAL=0;	//Set VAL =0
	SysTick->CTRL = SysTick_CTRL_TICKINT_Msk | SysTick_CTRL_ENABLE_Msk;
	//Enable interrupt
}

/*!
 * Function Name:void SysTick_Handler()
 * Description :This function is the IRQ Handler which increments global varibale value for current time in tenths of seconds
 * @input: void
 * @Return : void
 */
//Event handler for SystickTimer for 15 seconds delay
//void SysTick_Handler()
//{
//	current_time++;	//Increment the gloabl variable
//}

/*!
 * Function Name:uint64_t get_current_time()
 * Description :This function returns current Systick Counter ValueE
 * @input: void
 * @Return : void
 */

uint64_t get_current_time()
{
	return current_time;	//Return current time
```

```
}

/*****************************************************************************
******************
 * Function Name:uint64_t time_passed(uint64_t since)
 * Description :This function returns time elapsed since the bootup
 * Reference from "Making Embedded Systems: Design Patterns for Great Software
;Elecia White Book"
 * @input: void
 * @Return : void

*****************************************************************************
*****************/

uint64_t time_passed(uint64_t since)
{
	uint64_t now = current_time;

	if(now >= since)
	{
		return now - since;
	}

	return (now + (time_max-since));
}

/*****************************************************************************
******************
 * Function Name:void time_stamp_print()
 * Description :This function prints the time stamp on host connected to the UART0
terminal
 * @input: void
 * @Return : void

*****************************************************************************
*****************/

void time_stamp_print()
{
	static char time_buf[2048] = {0};
	for(int i = 0; i < 2048; i++) time_buf[i] = '\0';    //Initialize array with
nullCharacters

	uint64_t tenths_count = get_current_time();   //Get current time

	float current = tenths_count / 10;

	//Calculations for conversion to Hours , mins , seconds
	uint64_t sec = (uint64_t)(current)%60;
	uint64_t min = (uint64_t)(current/60)%60;
	uint64_t hrs = (uint64_t)(current/3600)%60;


	sprintf(time_buf, "\n%02d:",  hrs);     //Convert hrs to string
	PRINTF("\n \r %s",time_buf);     //Send value over UART
```

```
        sprintf(time_buf, "%02d:",  min);//Convert min to string
        PRINTF(time_buf);//Send value over UART
        sprintf(time_buf, "%02d:",  sec);//Convert sec to string
        PRINTF(time_buf);//Send value over UART
        sprintf(time_buf, ".%1d",  tenths_count%10);//Convert tenths_count to string
        PRINTF(time_buf);//Send value over UART
}


/*
 * wave.h
 *
 *  Created on: Dec 1, 2019
 *      Author: SURAJ THITE
 */

#ifndef WAVE_H_
#define WAVE_H_

#include <math.h>
#include <stdint.h>

#define NUM_OF_SAMPLES 50
#define INV_3_FACTORIAL (1/6)
#define INV_5_FACTORIAL (1/120)
#define INV_7_FACTORIAL (1/5040)
#define PI 3.14159265358979323846


void sine_lookup_generate();

uint16_t get_next_val();


#endif /* WAVE_H_ */


/*
 * wave.h
 *
 *  Created on: Dec 1, 2019
 *      Author: SURAJ THITE
 */

#ifndef WAVE_H_
#define WAVE_H_

#include <math.h>
#include <stdint.h>

#define NUM_OF_SAMPLES 50
#define INV_3_FACTORIAL (1/6)
#define INV_5_FACTORIAL (1/120)
#define INV_7_FACTORIAL (1/5040)
#define PI 3.14159265358979323846
```

```
void sine_lookup_generate();

uint16_t get_next_val();


#endif /* WAVE_H_ */
```