```c
/*
 * Copyright 2016-2018 NXP Semiconductor, Inc.
 * All rights reserved.
 *
 * 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 NXP Semiconductor, Inc. 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.
 */

/**
 * @file    DMA_Project.c
 * @brief   Application entry point.
 */
#include <stdio.h>
#include "board.h"
#include "peripherals.h"
#include "pin_mux.h"
#include "clock_config.h"
#include "MKL25Z4.h"
#include "fsl_debug_console.h"

/* APPLICATION INCLUDES */
#include "adc_driver.h"
#include "dma_driver.h"
#include "peak_detect.h"


/* DEFINES AND TYPEDEFS */
#define PRINT_PRETTY_LINES    1
#define PRINT_TEXT_OUT        1
#if PRINT_PRETTY_LINES && PRINT_TEXT_OUT
#warning Printing Lines and Text is very slow and may break the program
#endif

#define BUFF_DOUBLE_SIZE    128
#define BUFF_ITEM_BYTES        2
#define BUFF_DOUBLE_BYTES    (BUFF_DOUBLE_SIZE*BUFF_ITEM_BYTES)
#define BUFF_HALF_SIZE        (BUFF_DOUBLE_SIZE/2)
#define BUFF_HALF_BYTES        (BUFF_HALF_SIZE*BUFF_ITEM_BYTES)
#define RAND_GPIO_BASE        GPIOE
#define RAND_GPIO_PORT        PORTE
#define RAND_GPIO_PIN        5
#define RAND_GPIO_SETUP        {kGPIO_DigitalOutput, 0}
#define RAND_PORT_SETUP        {.driveStrength = kPORT_HighDriveStrength, .mux = kPORT_MuxAsGpio, .pullSelect = kPORT_PullDown}
#define RAND_GPIO_CLOCK        kCLOCK_PortE

/* GLOBALS */
volatile int16_t buffer[BUFF_DOUBLE_SIZE];
volatile bool active_DMA_buffer = 0;
volatile void* const buffer_ptr_lut[] = {&buffer[0], &buffer[BUFF_HALF_SIZE]};


/*
 * @brief   Application entry point.
 */
int main(void)
{
      /* Init board hardware. */
```

```c
    BOARD_InitBootPins();
    BOARD_InitBootClocks();
    BOARD_InitBootPeripherals();
      /* Init FSL debug console. */
    BOARD_InitDebugConsole();

    PRINTF("START\n");

    // SETUP RANDOM GPIO
    CLOCK_EnableClock(RAND_GPIO_CLOCK);
    port_pin_config_t port_fig = RAND_PORT_SETUP;
    PORT_SetPinConfig(RAND_GPIO_PORT, RAND_GPIO_PIN, &port_fig);
    gpio_pin_config_t pin_fig = RAND_GPIO_SETUP;
    GPIO_PinInit(RAND_GPIO_BASE, RAND_GPIO_PIN, &pin_fig);

    // SETUP DMAMUX
    dma_mux_config dma_mux_fig_chan0 = DMA_MUX_CONFIG_DEFAULT;
    dma_error dma_mux_0_err = dma_mux_init(&dma_mux_fig_chan0);

    // SETUP DMA
    dma_init_config dma_fig_chan0 = DMA_INIT_CONFIG_DEFAULT;
    dma_fig_chan0.dma = DMA0;
    dma_fig_chan0.src_addr = &(ADC0->R[ADC_MUX_A]);
    dma_fig_chan0.dest_addr = &buffer[0];
    dma_fig_chan0.byte_count = BUFF_HALF_BYTES;
    dma_fig_chan0.src_size = DMA_SIZE_16;
    dma_fig_chan0.dest_size = DMA_SIZE_16;
    dma_fig_chan0.interrupt = true;
    dma_fig_chan0.peripheral_en = true;
    dma_fig_chan0.steal_cycles = true;
    dma_fig_chan0.dest_inc = true;
    dma_fig_chan0.auto_disable_req = true;

    dma_error dma_0_err = dma_init(&dma_fig_chan0);


    // SETUP ADC
    adc_init_config adc_fig = ADC_INIT_CONFIG_DEFAULT;
    adc_fig.channel = ADC_CHAN_DAD0;
    adc_fig.bits = ADC_BITS_16BIT_DIFF;
    adc_fig.continuous = ADC_CONTINUOUS_CONTINUOUS;
    adc_fig.avg_samps = ADC_SAMP_AVG_4;
    adc_fig.sample_cycle_add = ADC_SMP_CYCLE_ADD_HS_22;
    adc_fig.port = PORTE;
    adc_fig.pin_1 = 20;
    adc_fig.pin_2 = 21;
    adc_fig.dma_mode = ADC_DMA_ENABLED;

    adc_error adc_err = adc_init(&adc_fig);

    if(     (dma_0_err != DMA_ERROR_SUCCESS)     |
        (adc_err != ADC_ERROR_SUCCESS)          |
        (dma_mux_0_err != DMA_ERROR_SUCCESS))
    {
        __asm__("BKPT");
    }

    // Enable DMA Mux
    dma_mux_channel_enable(dma_mux_fig_chan0.dma_mux, dma_mux_fig_chan0.channel, true);

    bool last_active_DMA_buffer = active_DMA_buffer;
    uint16_t output_adc_counts = 0;
    uint16_t output_dbfs = 0;

    while(1)
    {
        if(active_DMA_buffer != last_active_DMA_buffer)
        {
            output_adc_counts = peak_output(buffer_ptr_lut[last_active_DMA_buffer], BUFF_HALF_SIZE, 1);
            output_dbfs = dbfs_output(output_adc_counts);

            #if PRINT_TEXT_OUT
            uint16_t out_whole = output_dbfs/100;
            uint16_t out_decimal = output_dbfs - (out_whole * 100);
                printf("ADC:%d - dBFS:-%d.%d\n", output_adc_counts, out_whole, out_decimal);
            #endif
            #if PRINT_PRETTY_LINES
                pretty_print(output_dbfs, 8);
            #endif
        }
    }

    return 0 ;
}

void DMA0_IRQHandler()
{
    uint32_t primask = DisableGlobalIRQ();                        // Disable Interrupts
    GPIO_SetPinsOutput(RAND_GPIO_BASE, 1 << RAND_GPIO_PIN);        // Turn on Pin

    DMA0->DMA[DMA_CHANNEL_0].DSR_BCR |= DMA_DSR_BCR_DONE(true);    // Clear Interrupt on the channel that finished

    active_DMA_buffer = !active_DMA_buffer;                        // Swap Buffers

    volatile void* buff_ptr = buffer_ptr_lut[active_DMA_buffer];// Look up the Buffer Ptr (mainly for readability)

    dma_transfer_restart(DMA0, DMA_CHANNEL_0, buff_ptr, BUFF_HALF_BYTES);    // Enable DMA

    GPIO_ClearPinsOutput(RAND_GPIO_BASE, 1 << RAND_GPIO_PIN);        // Turn off Pin
    EnableGlobalIRQ(primask);                                    // Enable Interrupts
}
```