```
 /*
 * MAIN Generated Driver File
 *
 * @file main.c
 *
 * @defgroup main MAIN
 *
 * @brief This is the generated driver implementation file for the MAIN driver.
 *
 * @version MAIN Driver Version 1.0.2
 *
 * @version Package Version: 3.1.2
*/

/*
© [2025] Microchip Technology Inc. and its subsidiaries.

    Subject to your compliance with these terms, you may use Microchip
    software and any derivatives exclusively with Microchip products.
    You are responsible for complying with 3rd party license terms
    applicable to your use of 3rd party software (including open source
    software) that may accompany Microchip software. SOFTWARE IS ?AS IS.?
    NO WARRANTIES, WHETHER EXPRESS, IMPLIED OR STATUTORY, APPLY TO THIS
    SOFTWARE, INCLUDING ANY IMPLIED WARRANTIES OF NON-INFRINGEMENT,
    MERCHANTABILITY, OR FITNESS FOR A PARTICULAR PURPOSE. IN NO EVENT
    WILL MICROCHIP BE LIABLE FOR ANY INDIRECT, SPECIAL, PUNITIVE,
    INCIDENTAL OR CONSEQUENTIAL LOSS, DAMAGE, COST OR EXPENSE OF ANY
    KIND WHATSOEVER RELATED TO THE SOFTWARE, HOWEVER CAUSED, EVEN IF
    MICROCHIP HAS BEEN ADVISED OF THE POSSIBILITY OR THE DAMAGES ARE
    FORESEEABLE. TO THE FULLEST EXTENT ALLOWED BY LAW, MICROCHIP?S
    TOTAL LIABILITY ON ALL CLAIMS RELATED TO THE SOFTWARE WILL NOT
    EXCEED AMOUNT OF FEES, IF ANY, YOU PAID DIRECTLY TO MICROCHIP FOR
    THIS SOFTWARE.
*/
#include "mcc_generated_files/system/system.h"
/*
    Main application
*/
adc_result_t analog_S1 = 0;
adc_result_t analog_S2 = 0;
const float offset = 0.483;

float V, P;
int signal = 0;

void blink(void)
{
    LED_SetHigh();
    __delay_ms(400);
    LED_SetLow();
}
void emitter_adc(void)
{
    ADC_ChannelSelect(ADC_CHANNEL_ANA1);
    ADC_ConversionStart();
    while (!ADC_IsConversionDone());
    analog_S1 = ADC_ConversionResultGet();
    printf("Distance: %u\r\n\n", analog_S1);

    DAC1_SetOutput(analog_S1);
}
void pressure_adc(void)
{
    ADC_ChannelSelect(ADC_CHANNEL_ANA0);
    ADC_ConversionStart();
    while (!ADC_IsConversionDone());
    analog_S2 = ADC_ConversionResultGet();
    //blink();
    V = analog_S2*5.00f/4095.00f;
    P = (V - offset)*250.00f;
    printf("Voltage: %.3f V\r\n", V);
    printf("Pressure: %.3f kPa\r\n\n", P);
}

int main(void)
{
    SYSTEM_Initialize();
    // If using interrupts in PIC18 High/Low Priority Mode you need to enable the Global High and Low Interrupts
    // If using interrupts in PIC Mid-Range Compatibility Mode you need to enable the Global Interrupts
    // Use the following macros to:

    // Enable the Global Interrupts
    //INTERRUPT_GlobalInterruptEnable();


    // Disable the Global Interrupts
    //INTERRUPT_GlobalInterruptDisable();
    while(1)
    {
        if(SW_PORT == 0)
        {
            blink();
        }

        emitter_adc();
        pressure_adc();

        if(P >= 0.8)
        {
            IO_RB0_SetHigh();
        }
        else
        {
            IO_RB0_SetLow();
        }

        if(IO_RB3_PORT == 1)
        {
            LED_SetHigh();
        }
        else
        {
            LED_SetLow();
        }
        __delay_ms(400);
    }
}
```