```c
/* ************************************************************************** */
/** Descriptive File Name

  @Company
    Company Name

  @File Name
    filename.c

  @Summary
    Brief description of the file.

  @Description
    Describe the purpose of this file.
 */
/* ************************************************************************** */

/* ************************************************************************** */
/* ************************************************************************** */
/* Section: Included Files                                                    */
/* ************************************************************************** */
/* ************************************************************************** */
#include "Mc32_PressAdc.h"
#include "app.h"
#include "peripheral/adc/plib_adc.h"
/* This section lists the other files that are included in this file.
 */

/* TODO:  Include other files here if needed. */


/* ************************************************************************** */
/* ************************************************************************** */
/* Section: File Scope or Global Data                                         */
/* ************************************************************************** */
/* ************************************************************************** */

/*  A brief description of a section can be given directly below the section
    banner.
 */

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



/* ************************************************************************** */
/* ************************************************************************** */
// Section: Local Functions                                                   */
/* ************************************************************************** */
/* ************************************************************************** */

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


/* ************************************************************************** */
/* ************************************************************************** */
// Section: Interface Functions                                               */
/* ************************************************************************** */
/* ************************************************************************** */

/*  A brief description of a section can be given directly below the section
    banner.
 */

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

void Press_InitADC (void){
    //Configuration de l'adresse choisi ADC
    PLIB_ADC_InputScanMaskAdd(ADC_ID_1, ADC_AN_SCAN_ADDRES);
    // Configure l'ADC en mode alterné
    PLIB_ADC_ResultFormatSelect(ADC_ID_1, ADC_RESULT_FORMAT_INTEGER_16BIT);
    //Choisir ce mode -> Buffer alterné
    PLIB_ADC_ResultBufferModeSelect(ADC_ID_1, ADC_BUFFER_MODE_TWO_8WORD_BUFFERS);
    //mode multiplexage
    PLIB_ADC_SamplingModeSelect(ADC_ID_1, ADC_SAMPLING_MODE_MUXA);

    //la lecture des ADC est cadensée par le timer interne
    PLIB_ADC_ConversionTriggerSourceSelect(ADC_ID_1, ADC_CONVERSION_TRIGGER_INTERNAL_COUNT);
    //Tension de réference de l'ADC alimentation 3V3
    PLIB_ADC_VoltageReferenceSelect(ADC_ID_1, ADC_REFERENCE_VDD_TO_AVSS);
    PLIB_ADC_SampleAcquisitionTimeSet(ADC_ID_1, 0x1F);
    PLIB_ADC_ConversionClockSet(ADC_ID_1, SYS_CLK_FREQ, 32);

    //ADC fait 3 mesures par interruption (car 3 canaux utilisés) -> adapter en fct des ADC utilisés
    PLIB_ADC_SamplesPerInterruptSelect(ADC_ID_1, ADC_1SAMPLE_PER_INTERRUPT);
    //active le scan en mode multiplexage des entrées AN
    PLIB_ADC_MuxAInputScanEnable(ADC_ID_1);

    // Enable the ADC module
    PLIB_ADC_Enable(ADC_ID_1);

}

S_ADCResults Press_ReadAllADC( void ) {
    //structure de valeurs brutes des ADCs
    volatile S_ADCResults rawResult;
    // Traitement buffer
    ADC_RESULT_BUF_STATUS BufStatus;
    //stop sample/convert
    PLIB_ADC_SampleAutoStartDisable(ADC_ID_1);
    // traitement avec buffer alterné
    BufStatus = PLIB_ADC_ResultBufferStatusGet(ADC_ID_1);
    //Buffer 8 bits -> 0 à 7 -> expliqué après
    if (BufStatus == ADC_FILLING_BUF_0TO7) {
        rawResult.AN3 = PLIB_ADC_ResultGetByIndex(ADC_ID_1, 0);
    }
    else //Buffer 8 bits -> 8 à 15
    {
        rawResult.AN3 = PLIB_ADC_ResultGetByIndex(ADC_ID_1, 8);
    }
    // Retablit Auto start sampling
    PLIB_ADC_SampleAutoStartEnable(ADC_ID_1);

    //retourner valeurs lue
    return rawResult;
}

float Press_RawToVoltage(float raw){
    float voltage = 0;
    /* Raw ADC to voltage */
    voltage = raw * ADC_RES;
    /* Voltage before op-amp */
    voltage = voltage / OPAMP_GAIN;
    return voltage;
}

float Press_voltageToPressure(float voltage) {
    float pressure = 0;
        /* Convet voltage to pressure in bar */
    pressure = ((voltage - V_MIN)*P_RANGE)/V_MAX;

    return pressure;
}

float Press_readPressure( void ) {
    //structure de valeurs brutes des ADCs
    volatile S_ADCResults rawResult;
    /* Voltage variable */
    float voltage = 0;
    /* Pressure variable */
    float pressure = 0;
    /* Read ADC */
    rawResult = Press_ReadAllADC();
    /* Convert raw data to voltage */
    voltage = Press_RawToVoltage(rawResult.AN3);
    /* Get the pressure from the voltage */
    pressure = Press_voltageToPressure(voltage);

    return pressure;
}


/* *****************************************************************************
 End of File
 */

```