```
/**
  Generated Main Source File

  Company:
    Microchip Technology Inc.

  File Name:
    main.c

  Summary:
    This is the main file generated using PIC10 / PIC12 / PIC16 / PIC18 MCUs

  Description:
    This header file provides implementations for driver APIs for all modules
selected in the GUI.
    Generation Information :
        Product Revision  :  PIC10 / PIC12 / PIC16 / PIC18 MCUs - 1.78.1
        Device            :  PIC18F46K22
        Driver Version    :  2.00
*/

/*
    (c) 2018 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. It is your responsibility to
comply with third party
    license terms applicable to your use of third party software (including open
source software) that
    may accompany Microchip software.

    THIS SOFTWARE IS SUPPLIED BY MICROCHIP "AS IS". NO WARRANTIES, WHETHER
    EXPRESS, IMPLIED OR STATUTORY, APPLY TO THIS SOFTWARE, INCLUDING ANY
    IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY, AND 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 IN ANY WAY RELATED TO THIS SOFTWARE WILL NOT EXCEED THE AMOUNT
    OF FEES, IF ANY, THAT YOU HAVE PAID DIRECTLY TO MICROCHIP FOR THIS
    SOFTWARE.
*/

/*
    Autores: Pedro Ferreira N2222035
             Bernardo Santos N2222033
```

```
    Repositorio:
https://github.com/perdo1305/MiniProjeto_PedroFerreira_BernardoSantos.X
*/

#include "lib_ili9341.h"
#include "mcc_generated_files/mcc.h"
/*
                         Main application
 */

// _________________VARIÁVEIS_____________________________
char string[50] = "";  // string para escrever no LCD

volatile bool SistemaControloLigado = false;  // 0 - Desligado, 1 - Ligado
static int prevSistemaControloLigado = -1;    // Variavel para guardar o estado
anterior do sistema de controlo
volatile bool BombaLigada = false;            // 0 - Desligada, 1 - Ligada
volatile bool BuzzerLigado = false;           // 0 - Desligado, 1 - Ligado
volatile bool UpdateLCD = true;               // 0 - Nao atualizar, 1 - Atualizar

volatile uint16_t MecanismoControlo = 0;  // 1 - Potenciometro, 2 - Interface serie

volatile uint16_t nivel_referencia = 0;              // valor RAW do potenciometro
volatile uint16_t nivel_referencia_percentagem = 0;  // valor em percentagem do
potenciometro
volatile uint16_t nivel_real = 0;                    // valor RAW dos sensores
volatile uint16_t nivel_real_percentagem = 0;        // valor em percentagem dos
sensores
//_____________________________________________________
//_________________VARIÁVEIS PARA O MENU_______________
uint8_t rxData;                      // Variavel para receber o caracter da EUSART
uint8_t menu = '0';                  // Variavel para o switch case
unsigned char cnt_char = 0;          // Contador de caracteres
unsigned char s[4];                  // String para guardar o valor introduzido
unsigned char carater_recebido = 1;  // Variavel para indicar se o caracter foi
aceite
unsigned char intro_valor = 0;       // Variavel para indicar se esta a introduzir
um valor
//_____________________________________________________

// _________________PROTOTIPOS_____________________________

void CheckUSART(void);             // Funcao para verificar se ha caracteres na
EUSART
void ShowMenuInTerminal(void);     // Funcao para mostrar o menu no terminal
uint16_t CheckSensores(void);      // Funcao para verificar o estado dos sensores
de nivel de agua
void Draw_Welcome_Screen(void);    // Funcao para desenhar o ecra de boas vindas
void Draw_Interface_Screen(void);  // Funcao para desenhar o ecra da interface
```

```
//_____________INTERRUPTS_____________________

/**
 * Funcao que e chamada quando o botao RB0 e pressionado
 * Desliga a bomba de agua e o sistema de controlo
 */
void INT0_MyInterruptHandler(void) {
    BombaLigada = false;
    SistemaControloLigado = false;
}

/**
 * Funcao que e chamada quando o ADC termina a conversao
 * Guarda o valor da conversao no nivel de referencia
 * Calcula a percentagem do nivel de referencia
 */
void ADC_MyInterruptHandler(void) {
    ADC_SelectChannel(channel_AN0);
    nivel_referencia = ADC_GetConversionResult();
    nivel_referencia_percentagem = (uint16_t)((nivel_referencia * 100.0) / 1023.0);
}

/**
 * Funcao que e chamada quando o timer 0 termina a contagem (5ms)
 * se o sistema de controlo estiver ligado, inicia uma nova conversao ADC
 * e verifica o estado dos sensores
 * Calcula a percentagem do nivel de agua
 */
void TMR0_MyInterruptHandler(void) {
    if (MecanismoControlo == 1) {
        ADC_SelectChannel(channel_AN0);
        ADC_StartConversion();
    }
    nivel_real = CheckSensores();  // Verifica o estado dos sensores
    nivel_real_percentagem = (nivel_real * 100) / 10;
}

/**
 * Funcao que e chamada quando o timer 1 termina a contagem (250ms)
 * se o sistema de controlo estiver ligado, pisca o led laranja
 */
void TMR1_MyInterruptHandler(void) {
    if (SistemaControloLigado) {
        IO_RB4_LED_ORANGE_Toggle();
    }
}

/**
 * Funcao que e chamada quando o timer 2 termina a contagem (5ms)
 * se o sistema de controlo estiver ligado, calcula o erro entre o nivel de
```

```
referencia e o nivel real
 * se o erro for positivo, liga a bomba de agua
 * se o erro for negativo, desliga a bomba de agua
 */
void TMR2_MyInterruptHandler(void) {
    volatile int erro_nivel = (int)(nivel_referencia_percentagem -
nivel_real_percentagem);
    if (erro_nivel > 0) {
        BombaLigada = true;
        IO_RB7_Motor_Control_SetHigh();  // Liga a bomba de agua
    } else {
        BombaLigada = false;
        IO_RB7_Motor_Control_SetLow();  // Desliga a bomba de agua
    }
}

/**
 * Funcao que e chamada quando o timer 6 termina a contagem (250ms)
 * Pisca o led de heartbeat
 * Atualiza o LCD
 */
void TMR6_MyInterruptHandler(void) {
    IO_RB6_Led_HeartBeat_Toggle();
    UpdateLCD = true;
}

//_____________________________________________________

void main(void) {
    SYSTEM_Initialize();
    TMR4_StopTimer();  // Desliga o timer 4 que controla o buzzer

    SPI2_Open(SPI2_DEFAULT);
    lcd_init();             // inicializacao do LCD
    Draw_Welcome_Screen();  // Desenha o ecra de boas vindas com o nome do projeto
e dos autores (x = 0 a 319, y = 0 a 239)

    INTERRUPT_GlobalInterruptHighEnable();  // Enable high priority global
interrupts
    INTERRUPT_GlobalInterruptLowEnable();   // Enable low priority global
interrupts
    INTERRUPT_PeripheralInterruptEnable();  // Enable peripheral interrupts

    INT0_SetInterruptHandler(INT0_MyInterruptHandler);  // handler para o botao RB0
    ADC_SetInterruptHandler(ADC_MyInterruptHandler);    // handler para o ADC

    TMR0_SetInterruptHandler(TMR0_MyInterruptHandler);  // 5ms (ADC trigger +
verificacao dos sensores)
    TMR0_StartTimer();                                  // Inicia o timer 0
```

```
    TMR1_SetInterruptHandler(TMR1_MyInterruptHandler);  // 250ms (Se o sistema de
controlo estiver ligado pisca led laranja)
    TMR1_StartTimer();                                  // Inicia o timer 1

    TMR2_SetInterruptHandler(TMR2_MyInterruptHandler);  // 5ms (Se o sistema de
controlo estiver ligado calculo do erro e ligar/desligar a bomba de agua)
    TMR2_StartTimer();                                  // Inicia o timer 2

    TMR6_SetInterruptHandler(TMR6_MyInterruptHandler);  // 250ms (Atualizar o LCD e
heartbeat LED)
    TMR6_StartTimer();                                  // Inicia o timer 6

    SistemaControloLigado = false;  // O sistema de controlo comeca desligado
    printf("O setup terminou!!!!\n");

    while (1) {
        if (prevSistemaControloLigado != SistemaControloLigado) {  // Check if
state has changed
            if (SistemaControloLigado) {                           // Se o sistema
de controlo estiver ligado
                TMR1_StartTimer();                                 // Liga o timer
1 que controla o led laranja
                TMR2_StartTimer();                                 // Liga o timer
2 que controla a bomba de agua através do erro entre o nivel de referencia e o
nivel real
                IO_RB3_LED_RED_SetLow();                           // Desliga o led
vermelho
            } else {                                               // Se o sistema
de controlo estiver desligado
                TMR1_StopTimer();                                  // Desliga o
timer 1 que controla o led laranja
                TMR2_StopTimer();                                  // Desliga o
timer 2 que controla a bomba de agua
                IO_RB4_LED_ORANGE_SetLow();                        // Desliga o led
laranja
                IO_RB3_LED_RED_SetHigh();                          // Acende o led
vermelho para indicar que o sistema esta desligado
            }
            prevSistemaControloLigado = SistemaControloLigado;  // atualiza o
estado anterior
        }

        if (UpdateLCD) {              // Atualiza o LCD a cada 250ms
            Draw_Interface_Screen();  // Desenha o ecra da interface com as barras
de nivel de agua e de referencia e os valores
            UpdateLCD = false;
        }

        CheckUSART();                                          // Verifica se ha
caracteres na EUSART para mostrar o menu no terminal / ler um valor para o programa
```

```
        if (carater_recebido || menu == '3' || menu == '4') {  // Se o caracter for
aceite ou se estiver a mostrar o nivel de agua ou o nivel de referencia
            ShowMenuInTerminal();                              // Mostra o menu no
terminal
            carater_recebido = 0;
        }

        if (nivel_real_percentagem >= 100) {  // Se o nivel de agua for igual ou
superior a 100%
            TMR4_StartTimer();                // Liga o timer 4 que controla o
buzzer
            PWM5_LoadDutyValue(50);           // Liga o buzzer com 50% de duty
cycle
        } else {
            TMR4_StopTimer();  // Desliga o timer 4 que controla o buzzer
        }
    }
}

/**
 * Funcao para mostrar o menu no terminal
 * tambem e usada para ler um valor para o programa
 */
void ShowMenuInTerminal() {
    switch (menu) {
        case '0':               // Apresenta o menu principal
            EUSART1_Write(12);  // Escreve na primeira linha do terminal
            printf("\r\n-- MENU PRINCIPAL --\r\n");
            if (BombaLigada == false) {  // Se a bomba de agua estiver desligada
                printf("\r\n1 - Ligar bomba de agua");
            } else {  // Se a bomba de agua estiver ligada
                printf("\r\n1 - Desligar bomba de agua");
            }

            if (SistemaControloLigado == false) {  // Se o sistema de controlo
estiver desligado
                printf("\r\n2 - Ativar controlo do nivel de agua");
            } else {  // Se o sistema de controlo estiver ligado
                printf("\r\n2 - Desativar controlo do nivel de agua");
            }
            printf("\r\n3 - Visualizar a percentagem do nivel de agua");
            printf("\r\n4 - Visualizar o nivel de referencia");
            if (MecanismoControlo == 2) {  // Se o mecanismo de controlo for a
interface serie
                printf("\r\n5 - Programar novo valor de referencia atraves do
terminal");
            }
            printf("\r\n0 - Voltar ao Menu Principal");
            printf("\r\nOpcao: ");
            menu = 0;
```

```
            break;
        case '1':  // Ligar/desligar a bomba de agua
            EUSART1_Write(12);
            if (SistemaControloLigado) {  // Se o sistema de controlo estiver
ligado
                printf("\r\nNao e possivel ligar/desligar a bomba de agua ");
                printf("\r\nenquanto o sistema de controlo estiver ligado");
            } else {                // Se o sistema de controlo estiver desligado
                if (BombaLigada) {  // Se a bomba de agua estiver ligada
                    printf("\r\nBomba de agua desligada !!");
                    IO_RB7_Motor_Control_SetLow();  // Desliga a bomba de agua
                    BombaLigada = false;
                } else {  // Se a bomba de agua estiver desligada
                    printf("\r\nBomba de agua ligada !!");
                    IO_RB7_Motor_Control_SetHigh();  // Liga a bomba de agua
                    BombaLigada = true;
                }
            }
            printf("\r\nPrima 0 para voltar ao Menu Principall");
            printf("\r\nOpcao: ");
            menu = 0;
            break;
        case '2':  // Ativar controlo do nivel de agua
            EUSART1_Write(12);
            if (SistemaControloLigado == false) {  // Se o sistema de controlo
estiver desligado
                printf("\r\nSistema de controlo ligado !!");
                printf("\r\nEscolha o mecanismo para controlar o nivel de agua:");
                printf("\r\n1 - Potenciometro");
                printf("\r\n2 - Interface serie");
                printf("\r\nOpcao: ");
                intro_valor = 2;
                cnt_char = 0;
            } else {  // Se o sistema de controlo estiver ligado
                printf("\r\nSistema de controlo desligado !!");
                printf("\r\n\nPrima 0 para voltar ao Menu Principal");
                printf("\r\nOpcao: ");
                SistemaControloLigado = false;  // Desliga o sistema de controlo
            }
            menu = 0;
            break;
        case '3':  // Visualizar a percentagem do nivel de agua
            EUSART1_Write(12);
            printf("\r\nPercentagem do nivel de agua: %hu %%",
nivel_real_percentagem);
            // printf("\r\nRaw: %d", nivel_real);
            printf("\r\n\nPrima 0 para voltar ao Menu Principal");
            printf("\r\nOpcao: ");
            menu = '3';
            break;
```

```
        case '4':  // Visualizar o nivel de referencia
            EUSART1_Write(12);
            printf("\r\nNivel de referencia: %hu %%",
nivel_referencia_percentagem);
            // printf("\r\nRaw: %d", nivel_referencia);
            printf("\r\n\nPrima 0 para voltar ao Menu Principal");
            printf("\r\nOpcao: ");
            menu = '4';
            break;
        case '5':  // Programar novo valor de referencia
            EUSART1_Write(12);
            if (MecanismoControlo == 2) {  // Se o mecanismo de controlo for a
interface serie
                printf("\r\nNivel de referencia atual: %hu %%",
nivel_referencia_percentagem);
                printf("\r\nIntroduza o novo valor de referencia (0-100): ");
                intro_valor = 1;
                cnt_char = 0;
            } else {
                printf("\r\nNao e possivel programar novo valor de referencia,
utilize o potenciometro");
            }
            menu = 0;
            break;
        case 1:  // opcao para ler um numero de referencia
            s[cnt_char] = rxData;
            if (cnt_char == 3 || rxData == 13) {  // recebeu 3 caracteres ou ENTER
                if (cnt_char == 3) {
                    cnt_char++;
                }
                s[cnt_char] = '\0';                                          //
Coloca o char de fim de string
                nivel_referencia_percentagem = (uint16_t)atoi((const char*)s);  //
Converte a string num inteiro
                EUSART1_Write(12);
                // so pode ser entre 0 e 100
                if (nivel_referencia_percentagem <= 100) {  // Se o valor
introduzido for valido
                    printf("\r\nNivel de referencia = %3d %%\r\n",
nivel_referencia_percentagem);
                } else {  // Se o valor introduzido for invalido
                    printf("\r\nNivel de referencia invalido\r\n");
                    nivel_referencia_percentagem = 0;
                }
                printf("\r\nPrima 0 para voltar ao Menu Principal\n");
                intro_valor = 0;
            } else {
                cnt_char++;
            }
            menu = 0;
```

```
            break;
        case 2:  // opcao para ler um numero do mecanismo de controlo
            s[cnt_char] = rxData;
            if (cnt_char == 1 || rxData == 13) {  // recebeu 1 caracter ou ENTER
                if (cnt_char == 1) {
                    cnt_char++;
                }
                s[cnt_char] = '\0';                                    // Coloca o
char de fim de string
                MecanismoControlo = (uint16_t)atoi((const char*)s);  // Converte a
string num inteiro
                EUSART1_Write(12);
                if (MecanismoControlo == 1) {  // se o mecanismo de controlo for o
potenciometro
                    printf("\r\nMecanismo de controlo => Potenciometro\r\n");
                    SistemaControloLigado = true;     // Liga o sistema de controlo
                } else if (MecanismoControlo == 2) {  // se o mecanismo de controlo
for a interface serie
                    printf("\r\nMecanismo de controlo => Interface serie\r\n");
                    SistemaControloLigado = true;  // Liga o sistema de controlo
                } else {                           // se o mecanismo de controlo
for invalido
                    printf("\r\nMecanismo de controlo invalido\r\n");
                    MecanismoControlo = 0;
                    SistemaControloLigado = false;  // Desliga o sistema de
controlo
                }
                printf("\r\nPrima 0 para voltar ao Menu Principal\n");
                intro_valor = 0;
            } else {
                cnt_char++;
            }
            menu = 0;
            break;
        default:  // Opcao Invalida
            EUSART1_Write(12);
            printf("\r\nOpcao Invalida!");
            printf("\r\nPrima 0 para voltar ao Menu Principal\n");
            menu = 0;
            break;
    }
}

/**
 * Funcao para verificar se ha caracteres na EUSART
 * Se houver, le o caracter e mostra-o no terminal
 */
void CheckUSART() {
    if (EUSART1_is_rx_ready()) {
        rxData = EUSART1_Read();                                  // Funcao que le
```

```
caracter da EUSART
        EUSART1_Write(rxData);                                  // Mostra caracter
recebido devolvendo-o ah EUSART
        if ((rxData >= '0' && rxData <= '9') || rxData == 13)  // Protecao / Se
estiver enrtre 0 e 9 ou ENTER
        {
            carater_recebido = 1;  // Indica caracter aceite
            menu = rxData;         // O caracter ser]a usado para o switch case
        } else {
            carater_recebido = 0;  // Caso contrario nao aceita o caracter e...
            menu = '0';            // ...volta a mostrar o menu principal opcao 0
        }
        if (intro_valor == 1)  // Se estiver em modo de ler um valor para o
programa e nao escolher um item do menu
        {
            menu = 1;                // Vai para o switch case 1 onde ira carregar
os digitos lidos numa string
        } else if (intro_valor == 2)  // Se estiver em modo de ler um valor para o
programa e nao escolher um item do menu
        {
            menu = 2;  // Vai para o switch case 2 onde ira carregar os digitos
lidos numa string
        }
    }
}

/**
 * Funcao para verificar o estado dos sensores de nivel de agua
 * Percorre o array dos sensores de nivel de agua
 * Se o sensor estiver a detetar agua incrementa o nivel de agua
 * Retorna o nivel de agua
 */
uint16_t CheckSensores() {
    int getValueFuncs[10];                    // Array dos sensores de nivel de
agua
    getValueFuncs[0] = IO_RD7_N01_GetValue();  // Sensor 1 PORTDbits.RD7
    getValueFuncs[1] = IO_RD6_N02_GetValue();  // Sensor 2 PORTDbits.RD6
    getValueFuncs[2] = IO_RD5_N03_GetValue();  // Sensor 3 PORTDbits.RD5
    getValueFuncs[3] = IO_RD3_N04_GetValue();  // Sensor 4 PORTDbits.RD3
    getValueFuncs[4] = IO_RD2_N05_GetValue();  // Sensor 5 PORTDbits.RD2
    getValueFuncs[5] = IO_RC5_N06_GetValue();  // Sensor 6 PORTCbits.RC5
    getValueFuncs[6] = IO_RC4_N07_GetValue();  // Sensor 7 PORTCbits.RC4
    getValueFuncs[7] = IO_RC3_N08_GetValue();  // Sensor 8 PORTCbits.RC3
    getValueFuncs[8] = IO_RC0_N09_GetValue();  // Sensor 9 PORTCbits.RC0
    getValueFuncs[9] = IO_RC1_N10_GetValue();  // Sensor 10 PORTCbits.RC1

    uint16_t nivel_agua = 0;
    for (int i = 0; i < 10; i++) {    // Percorre o array dos sensores de nivel de
agua
        if (getValueFuncs[i] == 0) {  // Se o sensor estiver a detetar agua
```

```
            nivel_agua++;             // Incrementa o nivel de agua
        }
    }
    return nivel_agua;  // Retorna o nivel de agua
}

/**
 * Funcao para desenhar o ecra de boas vindas com o nome do projeto e dos autores
 */
void Draw_Welcome_Screen() {
    lcd_draw_line(9, 210, 310, 210, ILI9341_WHITE);
    lcd_draw_string(40, 215, "ENGENHARIA ELETROTECNICA", BLACK, WHITE);
    snprintf(string, sizeof(string), "MICROPROCESSADORES");
    lcd_draw_string(60, 190, string, WHITE, BLACK);
    snprintf(string, sizeof(string), "2023 / 2024");
    lcd_draw_string(120, 165, string, WHITE, BLACK);
    snprintf(string, sizeof(string), "SISTEMA PARA CONTROLO");
    lcd_draw_string(20, 140, string, ILI9341_ORANGE, BLACK);
    snprintf(string, sizeof(string), "DO NIVEL DE AGUA");
    lcd_draw_string(20, 120, string, ILI9341_ORANGE, BLACK);
    snprintf(string, sizeof(string), "Autores: Pedro Ferreira");
    lcd_draw_string(20, 95, string, YELLOW, BLACK);
    snprintf(string, sizeof(string), "Bernardo Santos");
    lcd_draw_string(90, 75, string, YELLOW, BLACK);

    snprintf(string, sizeof(string), "Nivel de agua:");
    lcd_draw_string(20, 20, string, WHITE, BLACK);
    snprintf(string, sizeof(string), "Nivel de referencia:");
    lcd_draw_string(20, 40, string, WHITE, BLACK);
}

/**
 * Funcao para desenhar o ecra da interface
 * calcula a percentagem do nivel de agua e do nivel de referencia
 * desenha as barras de nivel de agua e de referencia
 * desenha os valores do nivel de agua e do nivel de referencia
 */
void Draw_Interface_Screen() {
    lcd_fill_rect(275, 10, 295, 192, WHITE);  // Barra branca para servir
background

    // Calculo da barra de agua
    uint16_t barra_agua = (nivel_real_percentagem * 180) / 100;
    if (barra_agua >= 180) {
        barra_agua = 180;
    } else if (barra_agua <= 0) {
        barra_agua = 1;
    }
    lcd_fill_rect(276, 11, 294, barra_agua + 10, BLUE);  // Barra de agua
```

```
    // Calculo da barra de referencia
    uint16_t barra_referencia = (nivel_referencia_percentagem * 180) / 100;
    if (barra_referencia >= 180) {
        barra_referencia = 180;
    } else if (barra_referencia <= 0) {
        barra_referencia = 1;
    }
    lcd_fill_rect(276, barra_referencia + 10, 294, barra_referencia + 12, RED);  //
Barra de referencia

    // print do valor do nivel de agua e do nivel de referencia
    snprintf(string, sizeof(string), "%d%%", nivel_referencia_percentagem);
    lcd_draw_string(180, 40, string, WHITE, BLACK);
    snprintf(string, sizeof(string), "%d%%", nivel_real_percentagem);
    lcd_draw_string(140, 20, string, WHITE, BLACK);
}

/**
 End of File
*/
```