```c
/* ************************************************************************** */
/*                                                                            */
/*   Nome do arquivo:       aquecedorECooler.c                                */
/*                                                                            */
/*   Descricao:             Arquivo contendo a implementacao                  */
/*                        das funcoes de interface do microcontrolador        */
/*                        com a resistencia de aquecimento e cooler do kit    */
/*                                                                            */
/*   Autores:              Gustavo Lino e Giacomo Dollevedo                   */
/*   Criado em:             21/04/2020                                        */
/*   Ultima revisao em:     29/07/2020                                        */
/* ************************************************************************** */

/* REVISÃƒO: */
/* ALTERADO PARA "&=" no UP_COUNTING, e sua mascara no board.h */
/* ADICIONADO DISABLE_COUNTER */
/* ALTERADO PARA "&=" no EDGE_ALIGNED, e sua mascara no board.h */
/* ADICIONADO UM CLEAR PARA "TPM_CNT" na inicializacao*/
/* ADICIONADA UMA FUNCAO PARA RESETAR O "TPM_CNT" (PWM_clearCounter)*/
/* ADICIONADO "CLEAR_16" no board.h */
/* ALTERADAS AS MASCARAS PARA "CLOCK_DIVIDER" no board.h*/
/* ALTERADO PARA "&=" no CLOCK_DIVIDER */


#include "board.h"
#include "aquecedorECooler.h"


/* ************************************************************************** */
/* Nome do metodo:         PWM_init                                           */
/* Descricao:              Inicializa os registradores para funcionamento do PWM   */
/*                       entre 5 e 20Hz                                       */
/*                                                                            */
/* Parametros de entrada:    n/a                                              */
/*                                                                            */
/* Parametros de saida:      n/a                                              */
/* ************************************************************************** */
void PWM_init(void){


    /* Liberando o Clock para o timer/pwm*/
    SIM_SCGC6 |= TPM1_CLOCK_GATE;

    /*Divisor pro clock*/
    TPM1_SC   |= CLOCK_DIVIDER_64;

    /*Selecao do clock de 32kHz*/
    /*MCGIRCLK == internal reference clock*/
    SIM_SOPT2 |= MCGIRCLK_SELECT;


    /*Desabilitando o LPTPM Counter para poder alterar o modo de contagem*/
    TPM1_SC &= DISABLE_COUNTER;

    /*Modo de up-counting */
    TPM1_SC &= PWM_UP_COUNTING;

    /*Incrementa a cada pulso*/
    TPM1_SC   |= PWM_EVERY_CLOCK;

    /*Modulo configurado para 49 (chegar numa freq de 10Hz)*/
    /*Portanto, conta ate 50 (0 a 49)*/
    TPM1_MOD  |= 0x0031;

    /*Configurando modo Edge Aligned PWM e High True Pulses nos canais 0 e 1*/
    TPM1_C0SC &= EDGE_ALIGNED_HIGH_TRUE;
```

```
67      TPM1_C1SC &= EDGE_ALIGNED_HIGH_TRUE;
68
69      /*DUTY CYCLE 50%*/
70      /*Inverte o sinal apos contar 25 vezes*/
71      TPM1_C0V |= 0x0019;
72      TPM1_C1V |= 0x0019;
73
74      TPM1_CNT &= CLEAR_16;
75
76      coolerfan_init();
77      heater_init();
78
79
80  }
81
82
83  /* ************************************************************************** */
84  /* Nome do metodo:          PWM_clearCounter                                  */
85  /* Descricao:               Reseta o contador TPM1_CNT para nao haver overflow */
86  /*                                                                            */
87  /* Parametros de entrada:    n/a                                              */
88  /*                                                                            */
89  /* Parametros de saida:      n/a                                              */
90  /* ************************************************************************** */
91  void PWM_clearCounter(void){
92
93      if(TPM1_CNT >= 0x7FFF)
94          TPM1_CNT &= CLEAR_16;
95
96  }
97
98  /* ************************************************************************** */
99  /* Nome do metodo:          coolerfan_init                                    */
100 /* Descricao:               Configura a liberacao do sinal PWM no pino PTA13  */
101 /*                                                                            */
102 /* Parametros de entrada:    n/a                                              */
103 /*                                                                            */
104 /* Parametros de saida:      n/a                                              */
105 /* ************************************************************************** */
106 void coolerfan_init(void){
107
108     SIM_SCGC5 |= PORTA_CLOCK_GATE;
109
110     PORTA_PCR13 |= MUX_ALT3;
111
112 }
113
114 /* ************************************************************************** */
115 /* Nome do metodo:          heater_init                                       */
116 /* Descricao:               Configura a liberacao do sinal PWM no pino PTA12  */
117 /*                                                                            */
118 /* Parametros de entrada:    n/a                                              */
119 /*                                                                            */
120 /* Parametros de saida:      n/a                                              */
121 /* ************************************************************************** */
122 void heater_init(void){
123
124     SIM_SCGC5 |= PORTA_CLOCK_GATE;
125
126     PORTA_PCR12 |= MUX_ALT3;
127 }
128
129
130 /* ************************************************************************** */
131 /* Nome do metodo:          coolerfan_PWMDuty                                 */
132 /* Descricao:               Configura o Duty Cycle do PWM para o cooler       */
133 /*                                                                            */
```

```
/* Parametros de entrada:   fCoolerDuty -> valor entre 0 e 1 indicando o Duty Cycle */
/*                                                                                  */
/* Parametros de saida:     n/a                                                     */
/* ******************************************************************************** */
void coolerfan_PWMDuty(float fCoolerDuty){

    unsigned char ucDuty = 0;

    if(0 <= fCoolerDuty && 1 >= fCoolerDuty){
        ucDuty = 50*fCoolerDuty;
    }

    else{
        ucDuty = 0x0019;
    }

    TPM1_C1V &= CLEAR_16;
    TPM1_C1V |= ucDuty;

}

/* ******************************************************************************** */
/* Nome do metodo:          heater_PWMDuty                                          */
/* Descricao:               Configura o Duty Cycle do PWM para o cooler             */
/*                                                                                  */
/* Parametros de entrada:   fHeaterDuty -> valor entre 0 e 1 indicando o Duty Cycle */
/*                                                                                  */
/* Parametros de saida:     n/a                                                     */
/* ******************************************************************************** */
void heater_PWMDuty(float fHeaterDuty){

    unsigned char ucDuty = 0;

    if(0 <= fHeaterDuty && 1 >= fHeaterDuty){
         ucDuty = 50*fHeaterDuty;
    }

    else{
        ucDuty = 0x0019;
    }

    TPM1_C0V &= CLEAR_16;
    TPM1_C0V |= ucDuty;

}
```