ES670B - Projeto de Sistemas Embarcados
Relatório 01 - Laboratório 1/7

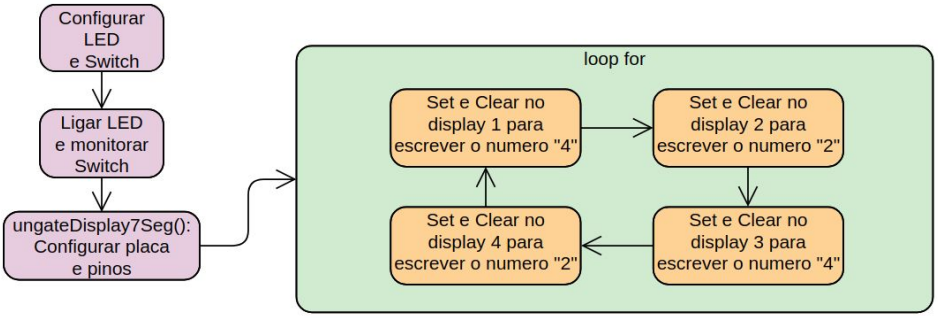
Laura Marchione - RA 156169
Victor Cintra Santos - RA 157461

Objetivos

Nesse laboratório temos os seguintes requisitos de projeto: monitorar uma chave “push button”; acionar um LED e acionar 4 displays de 7 segmentos.
Na Tarefa 01 do projeto tivemos que analisar os códigos fornecidos “ledswi_hal.h” e “ledswi_hal.c” para tratarmos a inicialização do LED e do switch e o seu controle.
Na Tarefa 02, tivemos que configurar a placa do controlador para conseguirmos acionar os 4 displays de 7 segmentos e mandarmos alguma informação para ser mostrada nos mesmos

Modelagem

Abaixo, segue o diagrama UML desenvolvido para tal projeto:



Matriz de Rastreabilidade

Requisito	**Implementação**
Monitorar chave “push button”	ledswi_hal.c - void ledswi_initLedSwitch(char cLedNum, char cSwitchNum); - switch_status_type_e ledswi_getSwitchStatus(char cSwitchNum);
Acionar LED	ledswi_hal.c - void ledswi_initLedSwitch(char cLedNum, char cSwitchNum); - void ledswi_setLed (char cLedNum);
Acionar 4 displays de 7 segmentos	main.c - ungateDisplay7Seg();

Notas

Sentimos dificuldade para compreender o funcionamento das funções do arquivo “ledswi_hal.c” já que não ficou muito claro o tipo de argumento que as funções necessitam e o uso delas (entendemos mais pela visão geral do projeto, o que possibilitou a continuação do mesmo). Isso ocorreu ao chamarmos a função “void ledswi_initLedSwitch(char cLedNum, char cSwitchNum)” ja que os parametros sao char, porem seria um numero a ser passado, no entanto passamos “1u” e “3u” que funcionariam como strings.

Tentamos criar uma função separada da main para automatizarmos as chamadas das linhas de código de set e clear dos displays de acordo com o numero que gostariamos de printar nos mesmos, para facilitar nos próximos laboratórios caso houvesse a necessidade de repetirmos esse tipo de lógica, porém houve dificuldades em trabalhar com o binário que é passado como parâmetro para essas funções; assim simplificamos, deixando no loop da main.

Apêndice

Na Figura 01 e 02 temos a inspeção da nossa variavel “switchStatus” que recebe o retorno da funcao “ledswi_getSwitchStatus” que mostra o status do switch. Na primeira figura mostra a saída quando o switch está apertado e na segunda quando não está apertado. Já na Figura 03, ha a execução do nosso código na placa, mostrando o LED 4 ligado e a saída “4242” nos displays de sete segmentos Na sequência há o código do desenvolvimento do projeto.

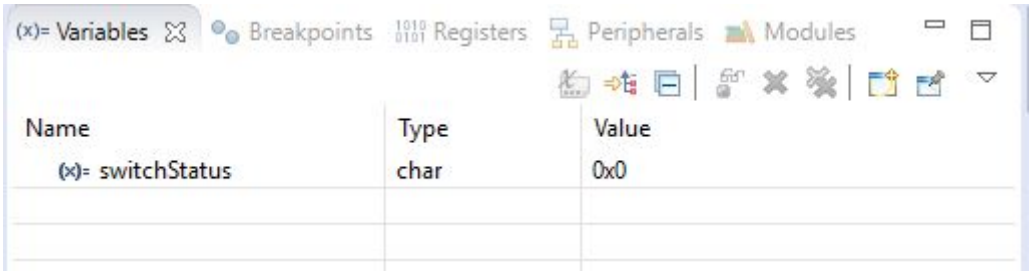


Figura 01

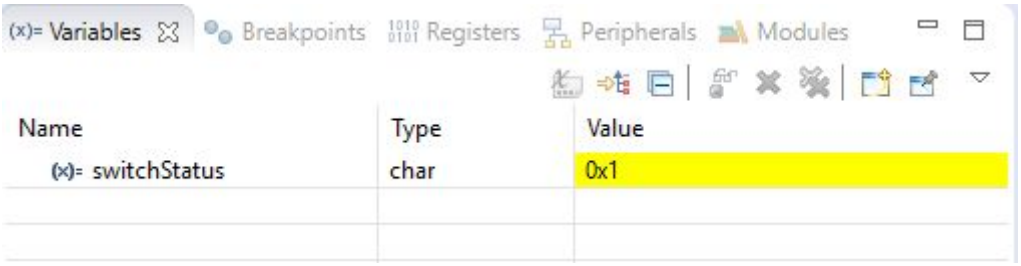


Figura 02

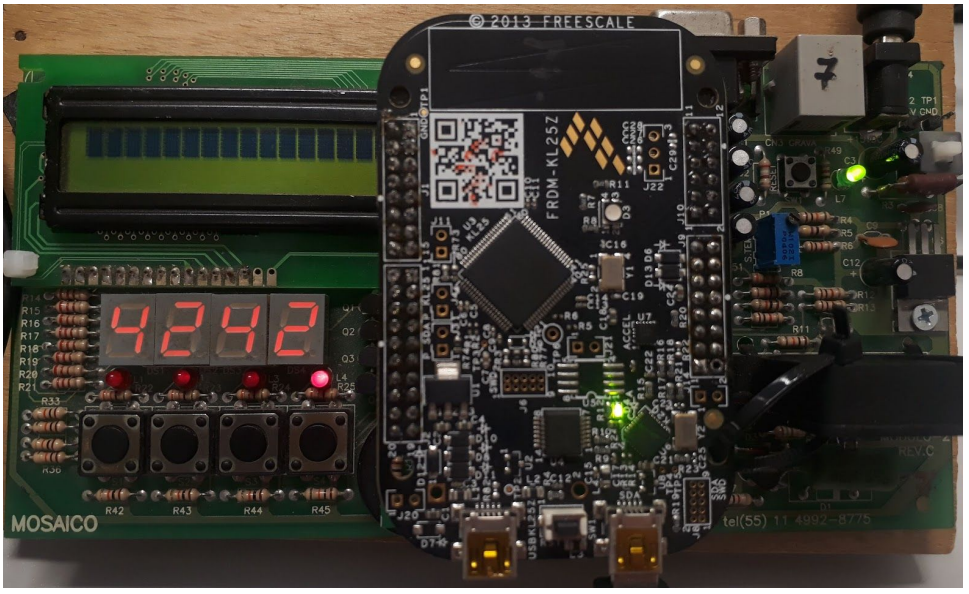


Figura 03

```c
#include "buzzer_hal.h"
#include "es670_peripheral_board.h"
#include "ledswi_hal.h"
#include "mcg_hal.h"
#include "util.h"
#include <MKL25Z4.h>

void ungateDisplay7Seg(){
    SIM_SCGC5  =  SIM_SCGC5_PORTC(0b11110011111111);  //  Liberacao  do  clock (ungate) da porta C e seus respectivos registradores
    PORTC_PCR0 = PORT_PCR_MUX(0x01); // Configurando registradores do display de 7seg da porta C como GPIO
    PORTC_PCR1 = PORT_PCR_MUX(0x01); //.
    PORTC_PCR2 = PORT_PCR_MUX(0x01); //.
    PORTC_PCR3 = PORT_PCR_MUX(0x01); //.
    PORTC_PCR4 = PORT_PCR_MUX(0x01); //.
    PORTC_PCR5 = PORT_PCR_MUX(0x01); //.
    PORTC_PCR6 = PORT_PCR_MUX(0x01); //.
    PORTC_PCR7 = PORT_PCR_MUX(0x01); //.
    PORTC_PCR13 = PORT_PCR_MUX(0x01); // Configurando enable display 01
    PORTC_PCR12 = PORT_PCR_MUX(0x01); // Configurando enable display 02
    PORTC_PCR11 = PORT_PCR_MUX(0x01); // Configurando enable display 03
    PORTC_PCR10 = PORT_PCR_MUX(0x01); // Configurando enable display 04
   GPIOC_PDDR = GPIO_PDDR_PDD(0b11110011111111); // Configurando registradores da porta C como outputs
}

int main(void){

    /* Tarefa 01 */
    char switchStatus;
    mcg_clockInit();
    ledswi_initLedSwitch(1u, 3u);
    ledswi_setLed(4u);
    switchStatus = ledswi_getSwitchStatus(3u);
```

```
    /* Tarefa 02 */
    ungateDisplay7Seg();

    for(;;){
        // Loop com a sequencia de ativacao e desativacao dos displays de 7seg de acorodo com o binario que controla os enables e os seguimentos que serao ligados
        GPIOC_PSOR = GPIO_PSOR_PTSO(0b10000001100110);
        util_genDelay088us();
        GPIOC_PCOR = GPIO_PCOR_PTCO(0b10000001100110);

        GPIOC_PSOR = GPIO_PSOR_PTSO(0b01000001011011);
        util_genDelay088us();
        GPIOC_PCOR = GPIO_PCOR_PTCO(0b01000001011011);

        GPIOC_PSOR = GPIO_PSOR_PTSO(0b00100001100110);
        util_genDelay088us();
        GPIOC_PCOR = GPIO_PCOR_PTCO(0b00100001100110);

        GPIOC_PSOR = GPIO_PSOR_PTSO(0b00010001011011);
        util_genDelay088us();
        GPIOC_PCOR = GPIO_PCOR_PTCO(0b00010001011011);
    }
    return 0;
}
```