Universidade Estadual de Campinas

Faculdade de Engenharia Mecânica

ES670 - Projeto de Sistemas Embarcados

Relatório - Projeto Prático (Parte 1)

Requisitos de Teclado, LEDs e Display de Sete Segmentos

Nome:	*RA*
Daniel Dello Russo Oliveira	101918
Davi Rodrigues	116581

13 de abril de 2016

1 Objetivo

O objetivo do projeto é, de maneira incremental, implementar no target os requisitos apresentados no roteiro[1] inicialmente desenvolvendo o modelo e depois implementando cada requisito. Estes requisitos são referentes à configuração e implementação de entradas de teclado, acionamento de LEDs, display de sete segmentos, protocolo de comunicação, display LCD, medição de velocidade de rotação, PWM, ADC e Controlador.

2 Modelagem

Detectamos logo no início do projeto um defeito estrutural no código fornecido quando lidando com GPIO, o identificador da porta e o número do pino utilizado eram referenciados em diversos locais diferentes do código, dificultando de maneira agravante mudanças na configuração de hardware. Para resolver isso, inicialmente pensamos em utilizar o arquivo fsl_gpio_hal.h da biblioteca da FRDM-Kl25Z, como isso não nos foi permitido e calcular as posições na memória de cada registrador seria reimplementar a biblioteca, escolhemos por criar macros que geram o mesmo estilo de código utilizado no exemplo fornecido, utilizando o operador do pré processador de concatenação ##. Porém, esse operador apresenta algumas particularidades, por exemplo macros que o utilizam no seu corpo não tem seus argumentos expandidos [4], para circular essa dificuldade criamos uma outra macro que funciona como uma wrapper para essa

Utilizando o Rational Rhapsody Modeler e tomando como base os requisitos propostos mostrados na figura 1, complementamos o modelo inicial[2] (requisitos de teclado e LEDs) adicionando um bloco ao modelo referente aos displays de sete segmentos (REQ1C), conforme mostrado na figura 2. Adicionamos também alguns blocos auxiliares relacionados ao gerenciamento de pinos GPIO, e a interrupções periódicas, que foram utilizados para nossa implementação

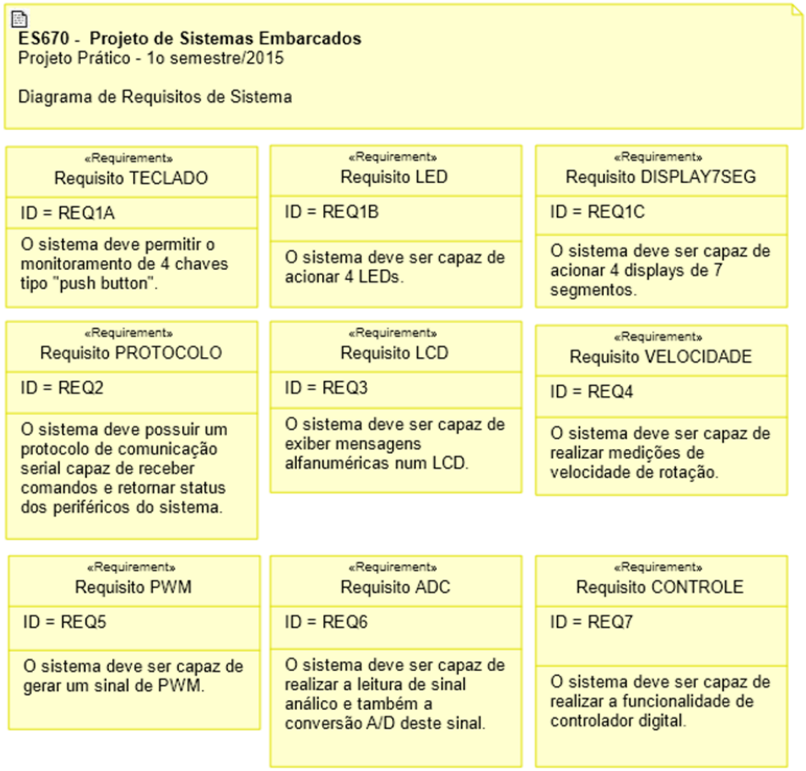


Figura 1: Diagrama de requisitos

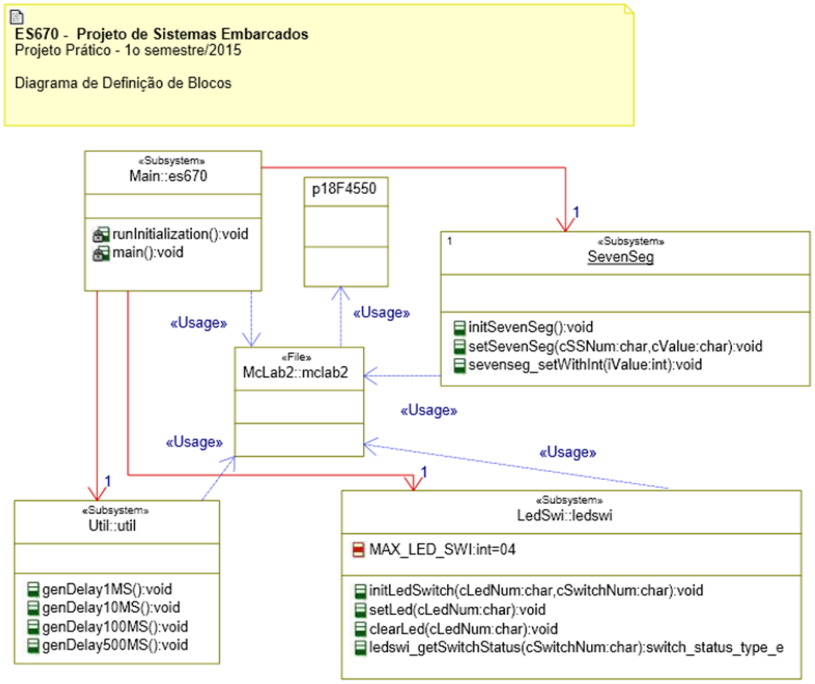


Figura 2: Diagrama de definição de blocos

O bloco SevenSeg possui três operações: inicialização, definição de saída (recebendo qual dos displays será usado e qual valor deve ser exibido) e definição de saída com inteiro (recebendo apenas o valor inteiro a ser exibido no conjunto de quatro displays).

3 Diagramas Esquemáticos

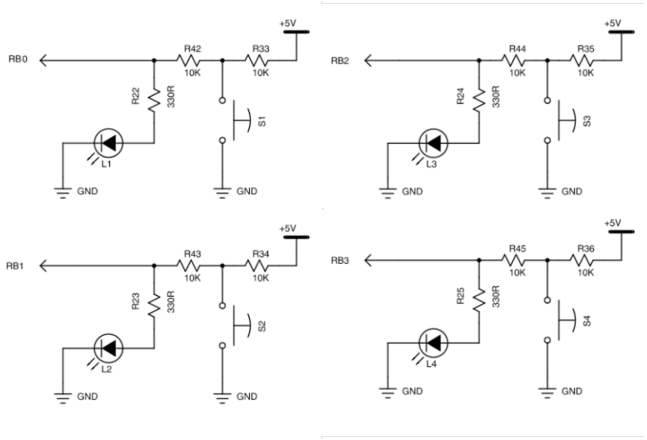
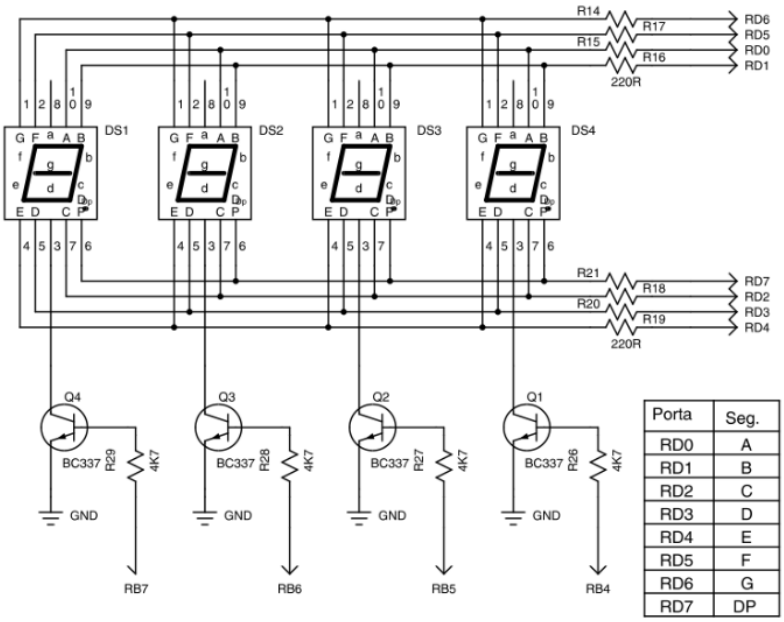


Figura 3: Esquema teclado e LEDs



Porta	Seg.
RD0	A
RD1	B
RD2	C
RD3	D
RD4	E
RD5	F
RD6	G
RD7	DP

Figura 4: Esquema sete segmentos

Como pode ser visto na figura 4, é necessário fazer um gerenciamento das portas RB4-7 e RD0-7 para que sejam mostrados os valores desejados nos displays de

sete segmentos. Para isso, é preciso alternar qual RB está ativo fazendo a mudança nos RD0-7 para que cada display esteja mostrando um valor diferente. É importante lembrar que a frequência dessa alternância seja escolhida de modo que o olho humano não perceba que os displays estão ligando e desligando.

4 Matriz de Rastreabilidade

A matriz de rastreabilidade apresentada na tabela 1 relaciona cada um dos requisitos com a sua implementação.

Tabela 1: Matriz de Rastreabilidade

ID do Requisito	**Implementação**

REQ1A	`ledswi.c`
	`- void ledswi_initLedSwitch(char cLedNum, char cSwitchNum)`
	`- switch_status_type_e ledswi_getSwitchStatus(char cSwitchNum)`

REQ1B	`ledswi.c`
	`- void ledswi_initLedSwitch(char cLedNum, char cSwitchNum)`
	`- void ledswi_setLed(char cLedNum)`
	`- void ledswi_clearLed(char cLedNum)`

REQ1C	`sevenSeg.c`
	`- void sevenseg_init(void)`
	`- void sevenseg_set(char cSSNum, sevenseg_value_e eValue)`
	`- void sevenseg_setWithInt (int iValue)`

5 Notas

Os maiores problemas identificados foram a dificuldade de conseguir conectar a placa com o computador e alguns mal contatos relacionados aos botões e do próprio PIC. Além disso, houve alguns conflitos com a IDE, pois muitas funcionalidades poderiam ter sido implementadas, já que a função de auto-completar é bem comum em IDE's e o duplo clique poderia selecionar a palavra em vez de colocar um breakpoint.

6 Referências

[1] Roteiro de Laboratório - Semanas 04 e 05 (disponibilizado para os alunos)

[2] Projeto do Modelo Inicial do Sistema (disponibilizado para os alunos)

[3] Código Fonte Inicial em Linguagem C (disponibilizado para os alunos)

[4] The C Preprocessor (Concatenation) `https://gcc.gnu.org/onlinedocs/cpp/Concatenation.html#Concatenation`

7 Apêndice

Listagem dos códigos fonte:

7.1 ../Sources/LedSwi/ledswi˙hal.h

```
/*
    ************************************************************
     */
/* File name:        ledswi_hal.h
                 */
/* File description: Header file containing the function/
    methods      */
/*                   prototypes of ledswi.c
                 */
/* Author name:      dloubach
                 */
/* Creation date:    09jan2015
                 */
/* Revision date:    13abr2016
                 */
/*
    ************************************************************
     */

#ifndef SOURCES_LEDSWI_LEDSWI_HAL_H_
#define SOURCES_LEDSWI_LEDSWI_HAL_H_

#define MAX_LED_SWI          04

typedef enum
```

```
{
    SWITCH_OFF,
    SWITCH_ON
} switch_status_type_e;

/**
 * As the hardware board was designed with LEDs/Switches sharing
 *  the same pins, this method configures how many LEDS and switches
 *  will be available for the application
 * @param cLedNum num of LEDs
 * @param cSwitchNum num of Switches (cLedNum + cSwitchNum <= MAX_LED_SWI)
 */
void ledswi_initLedSwitch(char cLedNum, char cSwitchNum);


/**
 * set the led ON
 * @param cLedNum which LED {1..4}
 */
void ledswi_setLed(char cLedNum);


/**
 * set the led OFF
 * @param cLedNum which LED {1..4}
 */
void ledswi_clearLed(char cLedNum);


/**
 * return the switch status
 * @param cSwitchNum which switch
 * @return If the switch is ON or OFF
```

```
 */
switch_status_type_e ledswi_getSwitchStatus(char cSwitchNum);


#endif /* SOURCES_LEDSWI_LEDSWI_HAL_H_ */
```

7.2 ../Sources/LedSwi/ledswi˙hal.c

```
/*****************************************************************/
/* File name:        ledswi_hal.c                                */
/* File description: This file has a couple of useful functions to */
/*                   control LEDs and Switches from peripheral board */
/* Author name:      dloubach                                    */
/* Creation date:    20jan2015                                   */
/* Revision date:    13abr2016                                   */
/*****************************************************************/

#include "ledswi_hal.h"
#include "KL25Z/es670_peripheral_board.h"

#define USING_OPENSDA_DEBUG

/**
 * As the hardware board was designed with LEDs/Switches sharing
```

```
 *  the same pins, this method configures how many LEDS and switches
 *  will be available for the application
 * @param cLedNum num of LEDs
 * @param cSwitchNum num of Switches (cLedNum + cSwitchNum <= MAX_LED_SWI)
 */
void ledswi_initLedSwitch(char cLedNum, char cSwitchNum)
{
    /* un-gate port clock*/
    SIM_SCGC5 = SIM_SCGC5_PORTA(CGC_CLOCK_ENABLED);

    /* set pin as gpio */
#ifndef USING_OPENSDA_DEBUG
    PORTA_PCR1 = PORT_PCR_MUX(LS1_ALT);
    PORTA_PCR2 = PORT_PCR_MUX(LS2_ALT);
#endif
    PORTA_PCR4 = PORT_PCR_MUX(LS3_ALT);
    PORTA_PCR5 = PORT_PCR_MUX(LS4_ALT);


    /* check if the number to configured is according to hardware dev kit */
    if((cLedNum + cSwitchNum) <= MAX_LED_SWI)
    {
        /* max number of peripherals to configure is ok, carry on */
        switch(cSwitchNum)
        {
            case 0:
                /* no switches in system configuration */
                /* all leds */
                GPIOA_PDDR |= GPIO_PDDR_PDD(LS1_DIR_OUTPUT | LS2_DIR_OUTPUT | LS3_DIR_OUTPUT | LS4_DIR_OUTPUT);
                break;
```

```
            case 1:
                /* just 1 switch */
                GPIOA_PDDR |= GPIO_PDDR_PDD(LS2_DIR_OUTPUT | LS3_DIR_OUTPUT | LS4_DIR_OUTPUT);
                GPIOA_PDDR &= ~GPIO_PDDR_PDD(LS1_DIR_INPUT);
                break;

            case 2:
                /* just 2 switches */
                GPIOA_PDDR |= GPIO_PDDR_PDD(LS3_DIR_OUTPUT | LS4_DIR_OUTPUT);
                GPIOA_PDDR &= ~GPIO_PDDR_PDD(LS1_DIR_INPUT | LS2_DIR_INPUT);
                break;

            case 3:
                /* 3 switches */
                GPIOA_PDDR |= GPIO_PDDR_PDD(LS4_DIR_OUTPUT);
                GPIOA_PDDR &= ~GPIO_PDDR_PDD(LS1_DIR_INPUT | LS2_DIR_INPUT | LS3_DIR_INPUT);
                break;

            case 4:
                /* 4 switches */
                GPIOA_PDDR &= ~GPIO_PDDR_PDD(LS1_DIR_INPUT | LS2_DIR_INPUT | LS3_DIR_INPUT | LS4_DIR_INPUT);
                break;
        } /* switch(cSwitchNum) */

    } /* if((cLedNum + cSwitchNum) <= MAX_LED_SWI) */

}
```

```c


/**
 * set the led ON
 * @param cLedNum which LED {1..4}
 */
void ledswi_setLed(char cLedNum)
{
    /* sanity check */
    if(cLedNum <= MAX_LED_SWI)
    {
        switch(cLedNum)
        {
            case 1:
                GPIOA_PSOR = GPIO_PSOR_PTSO( (0x01U << LS1_PIN) );
                break;
            case 2:
                GPIOA_PSOR = GPIO_PSOR_PTSO( (0x01U << LS2_PIN) );
                break;
            case 3:
                GPIOA_PSOR = GPIO_PSOR_PTSO( (0x01U << LS3_PIN) );
                break;
            case 4:
                GPIOA_PSOR = GPIO_PSOR_PTSO( (0x01U << LS4_PIN) );
                break;
        } /* switch(cLedNum) */

    } /* if(cLedNum <= MAX_LED_SWI) */
}


```

```c
/**
 * set the led OFF
 * @param cLedNum which LED {1..4}
 */
void ledswi_clearLed(char cLedNum)
{
    /* sanity check */
    if(cLedNum <= MAX_LED_SWI)
    {
        switch(cLedNum)
        {
            case 1:
                GPIOA_PCOR = GPIO_PCOR_PTCO( (0x01U << LS1_PIN) );
                break;
            case 2:
                GPIOA_PCOR = GPIO_PCOR_PTCO( (0x01U << LS2_PIN) );
                break;
            case 3:
                GPIOA_PCOR = GPIO_PCOR_PTCO( (0x01U << LS3_PIN) );
                break;
            case 4:
                GPIOA_PCOR = GPIO_PCOR_PTCO( (0x01U << LS4_PIN) );
                break;
        } /* switch(cLedNum) */

    } /* if(cLedNum <= MAX_LED_SWI) */
}



/**
```

```
 * return the switch status
 * @param cSwitchNum which switch
 * @return If the switch is ON or OFF
 */
switch_status_type_e ledswi_getSwitchStatus(char cSwitchNum)
{
    switch_status_type_e sstReturn = SWITCH_OFF;

    /* sanity check */
    if(cSwitchNum <= MAX_LED_SWI)
    {
        switch(cSwitchNum)
        {
            case 1:
                if(SWITCH_ON == ((GPIOA_PDIR & LS1_DIR_INPUT) >> LS1_PIN) )
                    sstReturn = SWITCH_ON;
                break;

            case 2:
                if(SWITCH_ON == ((GPIOA_PDIR & LS2_DIR_INPUT) >> LS2_PIN) )
                    sstReturn = SWITCH_ON;
                break;

            case 3:
                if(SWITCH_ON == ((GPIOA_PDIR & LS3_DIR_INPUT) >> LS3_PIN) )
                    sstReturn = SWITCH_ON;
                break;

            case 4:
                if(SWITCH_ON == ((GPIOA_PDIR & LS4_DIR_INPUT) >> LS4_PIN) )
                    sstReturn = SWITCH_ON;
```

```
                break;

        } /* switch(cSwitchNum) */

    } /* if(cSwitchNum <= MAX_LED_SWI) */

    /* return the result */
    return(sstReturn);

}
```

7.3 ../Sources/Mcg/mcg˙hal.h

```
/*
    ************************************************************
     */
/* File name:        mcg_hal.h
                 */
/* File description: Header file containing the functions
    /methods    */
/*                   interfaces for handling the
    Multipurpose clock  */
/*                   generator module
                 */
/* Author name:      dloubach
                 */
/* Creation date:    21out2015
                 */
/* Revision date:    25fev2016
                 */
/*
    ************************************************************
     */

#ifndef SOURCES_MCG_HAL_H_
#define SOURCES_MCG_HAL_H_
```

```
/* ********************************************** */
/* Method name:        mcg_clockInit                  */
/* Method description: main board clk configuration */
/* Input params:       n/a                            */
/* Output params:      n/a                            */
/* ********************************************** */
void mcg_clockInit(void);



#endif /* SOURCES_MCG_HAL_H_ */
```

7.4 ../Sources/Mcg/mcg˙hal.c

```
/* ************************************************************** */
/* File name:        mgc_hal.c                                    */
/* File description: Multipurpose clk generator hardware abstraction */
/*                   layer. Enables the clock configuration       */
/*                                                                */
/*                   Modes of Operation                           */
/*                   FLL Engaged Internal (FEI)    = DEFAULT      */
/*                   FLL Engaged External (FEE)                   */
/*                   FLL Bypassed Internal (FBI)                  */
/*                   FLL Bypassed External (FBE)                  */
```

```
/*                      PLL Engaged External (PEE)
                  */
/*                      PLL Bypassed External (PBE)
                  */
/*                      Bypassed Low Power Internal (BLPI)
                  */
/*                      Bypassed Low Power External (BLPE)
                  */
/*                      Stop
                  */
/*
                  */
/*                      For clock definitions, check the
    chapter         */
/*                      5.4 Clock definitions from
                  */
/*                      KL25 Sub-Family Reference Manual
                  */
/*
                  */
/* Author name:         dloubach
                  */
/* Creation date:       21out2015
                  */
/* Revision date:       13abr2016
                  */
/*
    ***************************************************************
     */

#include "mcg_hal.h"

/* systems include */
#include "fsl_smc_hal.h"
#include "fsl_port_hal.h"
#include "fsl_clock_manager.h"
```

```
/* EXTAL0 PTA18 */
#define EXTAL0_PORT                    PORTA
#define EXTAL0_PIN                     18U
#define EXTAL0_PINMUX                  kPortPinDisabled

/* XTAL0 PTA19 */
#define XTAL0_PORT                     PORTA
#define XTAL0_PIN                      19U
#define XTAL0_PINMUX                   kPortPinDisabled

/* OSC0 configuration */
#define OSC0_INSTANCE                  0U
#define OSC0_XTAL_FREQ                 8000000U /* 08 MHz*/
#define OSC0_SC2P_ENABLE_CONFIG        false
#define OSC0_SC4P_ENABLE_CONFIG        false
#define OSC0_SC8P_ENABLE_CONFIG        false
#define OSC0_SC16P_ENABLE_CONFIG       false
#define MCG_HGO0                       kOscGainLow
#define MCG_RANGE0                     kOscRangeVeryHigh
#define MCG_EREFS0                     kOscSrcOsc

/* RTC external clock configuration. */
#define RTC_XTAL_FREQ                  0U
#define RTC_SC2P_ENABLE_CONFIG         false
#define RTC_SC4P_ENABLE_CONFIG         false
#define RTC_SC8P_ENABLE_CONFIG         false
#define RTC_SC16P_ENABLE_CONFIG        false
#define RTC_OSC_ENABLE_CONFIG          false
#define RTC_CLK_OUTPUT_ENABLE_CONFIG   false

/* RTC_CLKIN PTC1 */
#define RTC_CLKIN_PORT                 PORTC
#define RTC_CLKIN_PIN                  1U
#define RTC_CLKIN_PINMUX               kPortMuxAsGpio
```

```
#define CLOCK_VLPR                    1U /* very low power run mode */
#define CLOCK_RUN                     2U /* run mode */

#ifndef CLOCK_INIT_CONFIG
#define CLOCK_INIT_CONFIG CLOCK_RUN
#endif


/* Configuration for enter VLPR mode, Core clock = 4MHz */
const clock_manager_user_config_t g_defaultClockConfigVlpr =
{
    .mcgConfig =
    {
        .mcg_mode            = kMcgModeBLPI,        // Work in BLPI mode
        .irclkEnable         = true,                // MCGIRCLK enable
        .irclkEnableInStop   = false,               // MCGIRCLK disable in STOP mode
        .ircs                = kMcgIrcFast,         // Select IRC4M
        .fcrdiv              = 0U,                  // FCRDIV is 0

        .frdiv    = 0U,
        .drs      = kMcgDcoRangeSelLow,              // Low frequency range
        .dmx32    = kMcgDmx32Default,                // DCO has a default range of 25%

        .pll0EnableInFllMode = false,                // PLL0 disable
```

```
        .pll0EnableInStop  = false,                // PLL0 disalbe in STOP mode
        .prdiv0            = 0U,
        .vdiv0             = 0U,
    },
    .simConfig =
    {
        .pllFllSel = kClockPllFllSelFll,           // PLLFLLSEL select FLL
        .er32kSrc  = kClockEr32kSrcLpo,            // ERCLK32K selection, use LPO
        .outdiv1   = 0U,
        .outdiv4   = 4U,
    },
    .oscerConfig =
    {
        .enable       = true,                      // OSCERCLK enable
        .enableInStop = false,                     // OSCERCLK disable in STOP mode
    }
};

/* Configuration for enter RUN mode, Core clock = 40 MHz */
/*
 * 24.5.1.1 Initializing the MCG
 * KL25 Sub-Family Reference Manual, Rev. 3, September 2012
 *
 * Refer also to
 * Table 24-18. MCG modes of operation
 *
 * On L-series devices the MCGFLLCLK frequency is limited to 48 MHz max
```

```
 * The DCO is limited to the two lowest range settings (MCG_C4[DRST_DRS] must be set to either 0b00 or 0b01).
 *
 * FEE (FLL engaged external)
 * fext / FLL_R must be in the range of 31.25 kHz to 39.0625 kHz
 * FLL_R is the reference divider selected by the C1[FRDIV] bits
 * F is the FLL factor selected by C4[DRST_DRS] and C4[DMX32] bits
 *
 * (fext / FLL_R) * F = (8 MHz / 256 ) * 1280 = 40 MHz
 *
 * */
const clock_manager_user_config_t g_defaultClockConfigRun =
{
    /* ------------ multipurpose clock generator configurations --------------- */
    .mcgConfig =
    {
        .mcg_mode            = kMcgModeFEE,        // Work in FEE mode

        /* ------------------- MCGIRCCLK settings ------------------------ */
        .irclkEnable         = true,               // MCGIRCLK enable
        .irclkEnableInStop   = false,              // MCGIRCLK disable in STOP mode
        .ircs                = kMcgIrcSlow,        // Select IRC 32kHz
        .fcrdiv              = 0U,                 // FCRDIV is 0

```

```
        /* ---------------------- MCG FLL settings ------------------------ */
        .frdiv    = 0b011,                          // Divide Factor is 256 (EXT OSC 8 MHz / 256 = 31.250 kHz)
                                                    // The resulting frequency must be in the range 31.25 kHz to 39.0625 kHz
        .drs      = kMcgDcoRangeSelMid,             // frequency range
        .dmx32    = kMcgDmx32Default,               // DCO has a default range of 25%

        /* ---------------------- MCG PLL settings ------------------------ */
        .pll0EnableInFllMode = false,               // PLL0 disable
        .pll0EnableInStop  = false,                 // PLL0 disabLe in STOP mode
        .prdiv0            = 0x0U,
        .vdiv0             = 0x0U,
    },
    /* ------------ system integration module configurations ---------------- */
    .simConfig =
    {
        .pllFllSel = kClockPllFllSelFll,            // PLLFLLSEL select PLL
        .er32kSrc  = kClockEr32kSrcLpo,             // ERCLK32K selection, use LPO
        .outdiv1   = 0U,                            // core/system clock, as well as the bus/flash clocks.
        .outdiv4   = 1U,                            // bus and flash clock and is in addition to the System clock divide ratio
    },
```

```
    /* ------------ system oscillator output configurations ---------------- */
    .oscerConfig =
    {
        .enable       = true,                    // OSCERCLK enable
        .enableInStop = false,                   // OSCERCLK disable in STOP mode
    }
};


/**
 * Oscillator configuration
 */
void mcg_initOsc0(void)
{
    /* OSC0 configuration */
    osc_user_config_t osc0Config =
    {
        .freq                 = OSC0_XTAL_FREQ,
        .hgo                  = MCG_HGO0,
        .range                = MCG_RANGE0,
        .erefs                = MCG_EREFS0,
        .enableCapacitor2p    = OSC0_SC2P_ENABLE_CONFIG,
        .enableCapacitor4p    = OSC0_SC4P_ENABLE_CONFIG,
        .enableCapacitor8p    = OSC0_SC8P_ENABLE_CONFIG,
        .enableCapacitor16p   = OSC0_SC16P_ENABLE_CONFIG,
    };

    /* oscillator initialization */
    CLOCK_SYS_OscInit(OSC0_INSTANCE, &osc0Config);
}
```

```c
/**
 * Function to initialize RTC external clock base on board configuration
 */
void mcg_initRtcOsc(void)
{

#if RTC_XTAL_FREQ
    // If RTC_CLKIN is connected, need to set pin mux. Another way for
    // RTC clock is set RTC_OSC_ENABLE_CONFIG to use OSC0, please check
    // reference manual for details
    PORT_HAL_SetMuxMode(RTC_CLKIN_PORT, RTC_CLKIN_PIN, RTC_CLKIN_PINMUX);
#endif

#if ((OSC0_XTAL_FREQ != 32768U) && (RTC_OSC_ENABLE_CONFIG))
#error Set RTC_OSC_ENABLE_CONFIG will override OSC0 configuration and OSC0 must be 32k.
#endif

    rtc_osc_user_config_t rtcOscConfig =
    {
        .freq                 = RTC_XTAL_FREQ,
        .enableCapacitor2p    = RTC_SC2P_ENABLE_CONFIG,
        .enableCapacitor4p    = RTC_SC4P_ENABLE_CONFIG,
        .enableCapacitor8p    = RTC_SC8P_ENABLE_CONFIG,
        .enableCapacitor16p   = RTC_SC16P_ENABLE_CONFIG,
        .enableOsc            = RTC_OSC_ENABLE_CONFIG,
    };

    /* OSC RTC initialization */
    CLOCK_SYS_RtcOscInit(0U, &rtcOscConfig);
}
```

```
/**
 * System clock configuration
 */
void mcg_initSystemClock(void)
{
    /* Set system clock configuration. */
    #if (CLOCK_INIT_CONFIG == CLOCK_VLPR)
        CLOCK_SYS_SetConfiguration(&g_defaultClockConfigVlpr);
    #else
        CLOCK_SYS_SetConfiguration(&g_defaultClockConfigRun);
    #endif
}


/* ********************************************** */
/* Method name:        mcg_clockInit              */
/* Method description: main board clk configuration */
/* Input params:       n/a                        */
/* Output params:      n/a                        */
/* ********************************************** */
void mcg_clockInit(void)
{
    /* enable clock for PORTs */
    CLOCK_SYS_EnablePortClock(PORTA_IDX);
    CLOCK_SYS_EnablePortClock(PORTC_IDX);
    CLOCK_SYS_EnablePortClock(PORTE_IDX);

    /* set allowed power mode to allow all */
    SMC_HAL_SetProtection(SMC, kAllowPowerModeAll);
```

```
    /* configure OSC0 pin mux */
    PORT_HAL_SetMuxMode(EXTAL0_PORT, EXTAL0_PIN, EXTAL0_PINMUX);
    PORT_HAL_SetMuxMode(XTAL0_PORT, XTAL0_PIN, XTAL0_PINMUX);

    /* setup OSC0 */
    mcg_initOsc0();

    /* setup OSC RTC */
    mcg_initRtcOsc();

    /* setup system clock */
    mcg_initSystemClock();
}
```

7.5 ../Sources/PIT/pit'hal.h

```
/*
    ************************************************************
    */
/* File name:        pit_hal.h
                    */
/* File description: Header file containing the functions
    /methods     */
/*                   interfaces for handling the Periodic
    Interruption*/
/*                   timer module
                 */
/* Author name:      ddello
                    */
/* Creation date:    10abr2016
                    */
/* Revision date:    10abr2016
                    */
```

```
/*
    ************************************************************
     */

#ifndef SOURCES_PIT_PIT_HAL_H_
#define SOURCES_PIT_PIT_HAL_H_

/**
 * Enables Periodic Interruption Timer module.
 * (With the stop on debug flag set to on)
 */
void pit_enable(void);

/**
 * Start interruptions for given timer, unchained mode.
 * Timer interruptions are masked.
 *
 * @param usTimer_numb  The number for the desired timer
    (0,1)
 * @param uiTimer_period  The number of bus_clock cycles
    between interrupts
 * @param fpInterrupt_handler     Timer interrupt handler
     routine address pointer
 */
void pit_start_timer_interrupt(unsigned short
    usTimer_numb, unsigned int uiTimer_period, void (*
    fpInterrupt_handler)(void));

/**
 * Stop interruptions for given timer, unchained mode.
 *
 * @param usTimer_numb  The number for the desired timer
    (0,1)
 */
void pit_stop_timer_interrupt(unsigned short usTimer_numb
    );
```

```c

/**
 * Mark interruption as handled for the given timer, this should be called by timer
 * interruption handlers once they are finished.
 *
 * @param usTimer_numb  The number for the desired timer (0,1)
 */
void pit_mark_interrupt_handled(unsigned short usTimer_numb);

/**
 * Pit interruption handler. Checks what timer caused the interruption and call the
 * correct timer interruption handler.
 */
void PIT_IRQHandler(void);
#endif /* SOURCES_PIT_PIT_HAL_H_ */
```

7.6 ../Sources/PIT/pit˙hal.c

```c
/* ************************************************************ */
/* File name:          pit_hal.c                                */
/* File description: File containing the functions/methods      */
/*                     for handling the Periodic Interruption   */
/*                     Timer module                             */
/* Author name:        ddello                                   */
```

```
/* Creation date:     10abr2016
                   */
/* Revision date:     13abr2016
                   */
/*
    ************************************************************
     */
//Careful when handling PIT DOC! Bit endianness is
    inverted in relation to GPIO doc

#include "pit_hal.h"
#include "KL25Z/es670_peripheral_board.h"

#define PIT_IRQ_NUMBER PIT_IRQn

/**
 *Default timer interruption handler. Does nothing.
 */
static void _nop_handler(void){
  PIT_TFLG0 |= PIT_TFLG_TIF(0x1u);
  PIT_TFLG1 |= PIT_TFLG_TIF(0x1u);
}

static void (*fpTimer0Handler)(void) = &_nop_handler;
static void (*fpTimer1Handler)(void) = &_nop_handler;

/**
 * Pit interruption handler. Checks what timer caused the
     interruption and call the
 * correct timer interruption handler.
 */
void PIT_IRQHandler(void){
  if(PIT_TFLG0){
    (*fpTimer0Handler)();
  }
  if(PIT_TFLG1){
```

```
    (*fpTimer1Handler)();
  }
}

/**
 * Enables Periodic Interruption Timer module.
 * (With the stop on debug flag set to on)
 */
void pit_enable(void){
  SIM_SCGC6 |= SIM_SCGC6_PIT_MASK;
  PIT_MCR &= ~PIT_MCR_MDIS(0x1u);
  //Freeze in debug mode
  PIT_MCR |= PIT_MCR_FRZ(0x1u);
  NVIC_ClearPendingIRQ(PIT_IRQ_NUMBER);
    NVIC_EnableIRQ(PIT_IRQ_NUMBER);
}

/**
 * Start interruptions for given timer, unchained mode.
 * Timer interruptions are masked.
 *
 * @param usTimer_numb  The number for the desired timer
   (0,1)
 * @param uiTimer_period  The number of bus_clock cycles
   between interrupts
 * @param fpInterrupt_handler     Timer interrupt handler
    routine address pointer
 */
void pit_start_timer_interrupt(unsigned short
   usTimer_numb, unsigned int uiTimer_period, void (*
   fpInterrupt_handler)(void)){
  if(!usTimer_numb){
    timer0Handler = fpInterrupt_handler;
    PIT_LDVAL0 = PIT_LDVAL_TSV(uiTimer_period);
    PIT_TCTRL0 &= ~PIT_TCTRL_CHN(0x1u);    /*Disable chain
    mode*/
```

```
    PIT_TCTRL0 |= PIT_TCTRL_TIE(0x1u);     /*Enable interrupts for timer 0*/
    PIT_TCTRL0 |= PIT_TCTRL_TEN(0x1u);     /*Enable timer 0*/
  }else{
    timer1Handler = fpInterrupt_handler;
    PIT_LDVAL1 = PIT_LDVAL_TSV(uiTimer_period);
    PIT_TCTRL1 &= ~PIT_TCTRL_CHN(0x1u);    /*Disable chain mode*/
    PIT_TCTRL1 |= PIT_TCTRL_TIE(0x1u);     /*Enable interrupts for timer 1*/
    PIT_TCTRL1 |= PIT_TCTRL_TEN(0x1u);     /*Enable timer 1*/
  }
}

/**
 * Stop interruptions for given timer, unchained mode.
 *
 * @param usTimer_numb  The number for the desired timer (0,1)
 */
void pit_stop_timer_interrupt(unsigned short usTimer_numb){
  if(!usTimer_numb){
    PIT_TCTRL0 &= ~PIT_TCTRL_TIE(0x1u);
    PIT_TCTRL0 &= ~PIT_TCTRL_TEN(0x1u);
  }else{
    PIT_TCTRL1 &= ~PIT_TCTRL_TIE(0x1u);
    PIT_TCTRL1 &= ~PIT_TCTRL_TEN(0x1u);
  }
}

/**
 * Mark interruption as handled for the given timer, this should be called by timer
```

```
 * interruption handlers once they are finished.
 *
 * @param usTimer_numb  The number for the desired timer (0,1)
 */
void pit_mark_interrupt_handled(unsigned short usTimer_numb){
  if(!usTimer_numb){
    PIT_TFLG0 |= PIT_TFLG_TIF(0x1u);
  }else{
    PIT_TFLG1 |= PIT_TFLG_TIF(0x1u);
  }
}
```

7.7 ../Sources/SevenSeg/sevenseg˙hal.h

```
/*************************************************************/
/* File name:        sevenseg_hal.h                          */
/* File description: Header file containing the functions/methods */
/*                   interfaces for handling SEVEN SEGMENT DISPLAY */
/*                    from the peripheral board              */
/* Author name:      ddello                                  */
/* Creation date:    18mar2016                               */
/* Revision date:    13abr2016                               */
/*************************************************************/
```

```c
#ifndef SOURCES_SEVEN_SEGMENT_HAL_H_
#define SOURCES_SEVEN_SEGMENT_HAL_H_

#include "KL25Z/es670_peripheral_board.h"

#define MAX_SEGMENT_NUMBER 8
#define MAX_DISP_NUMBER 4


typedef enum
{
    SEG_A = SEGA_PIN,
    SEG_B = SEGB_PIN,
    SEG_C = SEGC_PIN,
    SEG_D = SEGD_PIN,
    SEG_E = SEGE_PIN,
    SEG_F = SEGF_PIN,
    SEG_G = SEGG_PIN,
    SEG_DP = SEGDP_PIN,
  SEG_END = -1
} seven_segment_seg_type_e;

typedef enum
{
    DISP_1 = SEG_DISP1_PIN,
  DISP_2 = SEG_DISP2_PIN,
  DISP_3 = SEG_DISP3_PIN,
  DISP_4 = SEG_DISP4_PIN,
} seven_segment_disp_type_e;

/**
* Initialize the seven segment display
*/
void sevenseg_init(void);

```

```
/**
 * Sets only the selected segments as high. Setting the
    others as low
 * @param epDet_segments = Array with the segments that
    should be set as on (Last element should be SEG_END)
 */
void sevenseg_setSegs(seven_segment_seg_type_e*
    epSet_segments);

/**
 * Shows the value written in the segment pins to the
 *  given display after clearing the others
 * @param eDisplay the display to initialize.
 */
void sevenseg_setDisp(seven_segment_disp_type_e eDisplay)
    ;

/**
 * Shows the passed value in hexadecimal format in the
    seven segment display.
 * @param uiHex the value to be printed
 */
void sevenseg_printHex(unsigned int uiHex);

/**
 * Shows the passed value in decimal format in the seven
    segment display.
 * @param uiDec the value to be printed
 */
void sevenseg_printDec(unsigned int uiDec);

/**
 * Converts the less significative decimal digit of the
    argument into it's seven
 * segment display configuration
```

```
 * @param usDec the value to be converted (-1 if none should be displayed)
 * @param epRet address for results (should be a allocated array of minimal 9 elements)
 *
 * @return epRet
 */
seven_segment_seg_type_e* sevenseg_dec2segArray(unsigned short usDec, seven_segment_seg_type_e* epRet);

/**
 * Converts the less significative hexadecimal digit of the argument into it's seven
 * segment display configuration
 * @param usHex the value to be converted (-1 if none should be displayed)
 * @param epRet address for results (should be a allocated array of minimal 9 elements)
 *
 * @return epRet
 */
seven_segment_seg_type_e* sevenseg_hex2segArray(unsigned short usHex, seven_segment_seg_type_e* epRet);

#endif /* SOURCES_SEVEN_SEGMENT_HAL_H_ */
```

7.8 ../Sources/SevenSeg/sevenseg˙hal.c

```
/*
    ***********************************************************
     */
/* File name:        sevenseg_hal.c
                 */
/* File description: File containing the functions/methods          */
```

```
/*                    for handling SEVEN SEGMENT DISPLAY
                */
/*                     from the peripheral board
                */
/* Author name:       ddello
                */
/* Creation date:     18mar2016
                */
/* Revision date:     13abr2016
             */
/*
    ************************************************************
     */

#include "GPIO/gpio_hal.h"
#include "sevenseg_hal.h"
#include "math.h"
#include "KL25Z/es670_peripheral_board.h"
#include "PIT/pit_hal.h"

#define SEV_SEG_SEGMENT_MASK GPIO_HIGH << SEGA_PIN |
    GPIO_HIGH << SEGB_PIN | GPIO_HIGH << SEGC_PIN |
    GPIO_HIGH << SEGD_PIN | GPIO_HIGH << SEGE_PIN |
    GPIO_HIGH << SEGF_PIN | GPIO_HIGH << SEGG_PIN |
    GPIO_HIGH << SEGDP_PIN
#define SEV_SEG_DISP_MASK GPIO_HIGH << SEG_DISP1_PIN |
    GPIO_HIGH << SEG_DISP2_PIN | GPIO_HIGH <<
    SEG_DISP3_PIN | GPIO_HIGH << SEG_DISP4_PIN

static unsigned short usIsHex = 0;
static unsigned int uiPrintVal = -1;

/**
 * Interrupt handler for updating in display
    configuration
 */
```

```
void _sevenseg_interrupt_handler(void){
  static seven_segment_disp_type_e epDisplays[] = {DISP_1, DISP_2, DISP_3, DISP_4};
  static seven_segment_seg_type_e epSeg_array[9];
  static unsigned short usCur_disp = 0;
  if(usIsHex){
    sevenseg_hex2segArray(uiPrintVal/pow(16,usCur_disp), epSeg_array);
  }else{
    sevenseg_dec2segArray(uiPrintVal/pow(10,usCur_disp), epSeg_array);
  }
  sevenseg_setSegs(epSeg_array);
  sevenseg_setDisp(epDisplays[usCur_disp]);
  usCur_disp = (usCur_disp+1)%4;
  pit_mark_interrupt_handled(SEV_SEG_PIT_TIMER_NUMB);
}

/**
* Initialize the seven segment display
*/
void sevenseg_init(void){
  GPIO_UNGATE_PORT(SEV_SEG_PORT_ID);

  // Init the Seven Segment segment control pins as OUTPUT
  GPIO_INIT_PIN(SEV_SEG_PORT_ID, SEGA_PIN, GPIO_OUTPUT);
  GPIO_INIT_PIN(SEV_SEG_PORT_ID, SEGB_PIN, GPIO_OUTPUT);
  GPIO_INIT_PIN(SEV_SEG_PORT_ID, SEGC_PIN, GPIO_OUTPUT);
  GPIO_INIT_PIN(SEV_SEG_PORT_ID, SEGD_PIN, GPIO_OUTPUT);
  GPIO_INIT_PIN(SEV_SEG_PORT_ID, SEGE_PIN, GPIO_OUTPUT);
  GPIO_INIT_PIN(SEV_SEG_PORT_ID, SEGF_PIN, GPIO_OUTPUT);
  GPIO_INIT_PIN(SEV_SEG_PORT_ID, SEGG_PIN, GPIO_OUTPUT);
  GPIO_INIT_PIN(SEV_SEG_PORT_ID, SEGDP_PIN, GPIO_OUTPUT);
  // Init the Seven Segment segment display pins as OUTPUT
```

```
  GPIO_INIT_PIN(SEV_SEG_PORT_ID, SEG_DISP1_PIN, GPIO_OUTPUT);
  GPIO_INIT_PIN(SEV_SEG_PORT_ID, SEG_DISP2_PIN, GPIO_OUTPUT);
  GPIO_INIT_PIN(SEV_SEG_PORT_ID, SEG_DISP3_PIN, GPIO_OUTPUT);
  GPIO_INIT_PIN(SEV_SEG_PORT_ID, SEG_DISP4_PIN, GPIO_OUTPUT);

  //Init pit interrupts
  pit_enable();
  //Init timer 0
  pit_start_timer_interrupt(SEV_SEG_PIT_TIMER_NUMB, SEVEN_SEG_PIT_PERIOD, &_sevenseg_interrupt_handler);
}


/**
 * Sets only the selected segments as high. Setting the others as low
 * @param epDet_segments = Array with the segments that should be set as on (Last element should be SEG_END)
 */
void sevenseg_setSegs(seven_segment_seg_type_e* epSet_segments){
  //Clear all segments.
  GPIO_WRITE_MASK(SEV_SEG_PORT_ID, SEV_SEG_SEGMENT_MASK, GPIO_LOW);
  //Set the selected segments to high
  for(unsigned short usCounter = 0; epSet_segments[usCounter] != SEG_END; usCounter++){
    GPIO_WRITE_PIN(SEV_SEG_PORT_ID, epSet_segments[usCounter], GPIO_HIGH);
  }
}

```

```
/**
 * Shows the value written in the segment pins to the
 *  given display after clearing the others
 * @param eDisplay the display to initialize.
 */
void sevenseg_setDisp(seven_segment_disp_type_e eDisplay)
    {
  //Clear all displays
  GPIO_WRITE_MASK(SEV_SEG_PORT_ID, SEV_SEG_DISP_MASK,
    GPIO_LOW);
  //Activate the selected display
  GPIO_WRITE_PIN(SEV_SEG_PORT_ID, eDisplay, GPIO_HIGH);
}

/**
 * Shows the passed value in hexadecimal format in the
    seven segment display.
 * @param uiHex the value to be printed
 */
void sevenseg_printHex(unsigned int uiHex){
  usIsHex = 1;
  uiPrintVal = uiHex;
}

/**
 * Shows the passed value in decimal format in the seven
    segment display.
 * @param uiDec the value to be printed
 */
void sevenseg_printDec(unsigned int uiDec){
  usIsHex = 0;
  uiPrintVal = uiDec;
}

/**
```

```
 * Converts the less significative decimal digit of the argument into it's seven
 * segment display configuration
 * @param usDec the value to be converted (-1 if none should be displayed)
 * @param epRet address for results (should be a allocated array of minimal 9 elements)
 *
 * @return epRet
 */
seven_segment_seg_type_e* sevenseg_dec2segArray(unsigned short usDec, seven_segment_seg_type_e* epRet){
  if(usDec < 0){
    epRet[0] = SEG_END;
    return epRet;
  }
  epRet[0] = SEG_A;
  epRet[1] = SEG_B;
  epRet[2] = SEG_C;
  epRet[3] = SEG_D;
  epRet[4] = SEG_E;
  epRet[5] = SEG_F;
  epRet[6] = SEG_G;
  epRet[7] = SEG_END;
  switch(usDec%10){
  case 0:
    //{SEG_A,SEG_B,SEG_C,SEG_D,SEG_G,SEG_E,SEG_F,SEG_END};
    epRet[7] = SEG_END;
    break;
  case 1:
    //{SEG_B,SEG_C,SEG_END};
    epRet[0] = SEG_B;
    epRet[1] = SEG_C;
    epRet[2] = SEG_END;
    break;
```

```
case 2:
  //{SEG_A,SEG_B,SEG_G,SEG_D,SEG_E,SEG_END};
  epRet[2] = SEG_G;
  epRet[5] = SEG_END;
  break;
case 3:
  //{SEG_A,SEG_B,SEG_C,SEG_D,SEG_G,SEG_END}
  epRet[4] = SEG_G;
  epRet[5] = SEG_END;
  break;
case 4:
  //{SEG_G, SEG_B,SEG_C,SEG_F, SEG_END}
  epRet[0] = SEG_G;
  epRet[3] = SEG_F;
  epRet[4] = SEG_END;
  break;
case 5:
  //{SEG_A,SEG_G,SEG_C,SEG_D,SEG_F,SEG_END}
  epRet[1] = SEG_G;
  epRet[4] = SEG_F;
  epRet[5] = SEG_END;
  break;
case 6:
  //{SEG_A,SEG_G,SEG_C,SEG_D,SEG_E,SEG_F,SEG_END}
  epRet[1] = SEG_G;
  epRet[6] = SEG_END;
  break;
case 7:
  //{SEG_A,SEG_B,SEG_C,SEG_END}
  epRet[3] = SEG_END;
  break;
case 8:
  //{SEG_A,SEG_B,SEG_C,SEG_D,SEG_E,SEG_F,SEG_G,SEG_END}
  break;
case 9:
  //SEG_A,SEG_B,SEG_C,SEG_F,SEG_G,SEG_END}
```

```
    epRet[3] = SEG_F;
    epRet[4] = SEG_G;
    epRet[5] = SEG_END;
    break;
  }
  return epRet;
}

/**
 * Converts the less significative hexadecimal digit of the argument into it's seven
 * segment display configuration
 * @param usHex the value to be converted (-1 if none should be displayed)
 * @param epRet address for results (should be a allocated array of minimal 9 elements)
 *
 * @return epRet
 */
seven_segment_seg_type_e* sevenseg_hex2segArray(unsigned short usHex, seven_segment_seg_type_e* epRet){
  if(usHex < 0){
    epRet[0] = SEG_END;
    return epRet;
  }
  epRet[0] = SEG_A;
  epRet[1] = SEG_B;
  epRet[2] = SEG_C;
  epRet[3] = SEG_D;
  epRet[4] = SEG_E;
  epRet[5] = SEG_F;
  epRet[6] = SEG_G;
  epRet[7] = SEG_END;
  switch(usHex%16){
    case 0:
    case 1:
```

```
    case 2:
    case 3:
    case 4:
    case 5:
    case 6:
    case 7:
    case 8:
    case 9:
      return sevenseg_dec2segArray(usHex%16, epRet);
      break;
    case 10: //A
      //{SEG_A,SEG_B,SEG_C,SEG_G,SEG_E,SEG_F,SEG_END}
      epRet[3] = SEG_G;
      epRet[6] = SEG_END;
      break;
    case 11: //B (b)
      //{SEG_G,SEG_F,SEG_C,SEG_D,SEG_E,SEG_END}
      epRet[0] = SEG_G;
      epRet[1] = SEG_F;
      epRet[5] = SEG_END;
      break;
    case 12: //C
      //{SEG_A,SEG_E,SEG_F,SEG_D,SEG_END}
      epRet[1] = SEG_E;
      epRet[2] = SEG_F;
      epRet[4] = SEG_END;
      break;
    case 13: //D (d)
      //{SEG_G,SEG_B,SEG_C,SEG_D,SEG_E,SEG_END}
      epRet[0] = SEG_G;
      epRet[5] = SEG_END;
      break;
    case 14: //E
      //{SEG_A,SEG_G,SEG_F,SEG_D,SEG_E,SEG_END}
      epRet[1] = SEG_G;
      epRet[2] = SEG_F;
```

```
      epRet[5] = SEG_END;
      break;
    case 15: //F
      //{SEG_A,SEG_E,SEG_F,SEG_G,SEG_END}
      epRet[1] = SEG_E;
      epRet[2] = SEG_F;
      epRet[3] = SEG_G;
      epRet[4] = SEG_END;
      break;
  }
  return epRet;
}
```

7.9 ../Sources/GPIO/gpio˙hal.h

```
/*
    ************************************************************
     */
/* File name:        gpio_hal.h
                 */
/* File description: This file has a couple of useful
    macros to       */
/*                   control and init the GPIO pins from
    the KLM25Z  */
/* Author name:      ddello
                 */
/* Creation date:    01abr2016
                 */
/* Revision date:    13abr2016
                 */
/*
    ************************************************************
     */

#ifndef SOURCES_GPIO_GPIO_HAL_H_
#define SOURCES_GPIO_GPIO_HAL_H_
```

```
#include "KL25Z/es670_peripheral_board.h"

/* GPIO input / output */
#define GPIO_INPUT                    0x00U
#define GPIO_OUTPUT                   0x01U

#define GPIO_MUX_ALT                  0x01u

#define GPIO_HIGH      1
#define GPIO_LOW       0

/**
 * Ungates the clock for a gpio port
 * @param PORT_ID the GPIO port id(A,B)
 */
#define GPIO_UNGATE_PORT(PORT_ID)\
  _GPIO_UNGATE_PORT(PORT_ID)

//Wrapper macro above is needed for argument expansion when using concatenation
#define _GPIO_UNGATE_PORT(PORT_ID)\
  /* un-gate port clock*/\
    SIM_SCGC5 = SIM_SCGC5_PORT ## PORT_ID (CGC_CLOCK_ENABLED)

/**
 * inits a pin as GPIO in the given direction
 * @param PORT_ID the GPIO port id(A,B)
 * @param PIN_NUM pin number in port
 * @param DIR pin direction (GPIO_HIGH, GPIO_LOW)
 */
#define GPIO_INIT_PIN(PORT_ID, PIN_NUM, DIR)\
    _GPIO_INIT_PIN(PORT_ID, PIN_NUM, DIR)
```

```
//Wrapper macro above is needed for argument expansion when using concatenation
#define _GPIO_INIT_PIN(PORT_ID, PIN_NUM, DIR)\
    /* set pin as gpio */\
    PORT ## PORT_ID ## _PCR ## PIN_NUM = PORT_PCR_MUX(GPIO_MUX_ALT);\
    /* Set pin direction */\
    if(DIR == GPIO_OUTPUT){\
      GPIO ## PORT_ID ## _PDDR |= GPIO_PDDR_PDD(0x01 << PIN_NUM);\
    }else{\
      GPIO ## PORT_ID ## _PDDR &= ~GPIO_PDDR_PDD(0x01 << PIN_NUM);\
    }


/**
 * Writes a pin with the given value
 * @param PORT_ID the GPIO port id(A,B)
 * @oaram PIN_NUM pin number in port
 * @param VAL pin value (GPIO_HIGH, GPIO_LOW)
 */
#define GPIO_WRITE_PIN(PORT_ID, PIN_NUM, VAL)\
    _GPIO_WRITE_PIN(PORT_ID, PIN_NUM, VAL)

//Wrapper macro above is needed for argument expansion when using concatenation
#define _GPIO_WRITE_PIN(PORT_ID, PIN_NUM, VAL)\
  if(VAL == GPIO_HIGH){\
    GPIO ## PORT_ID ## _PSOR = GPIO_PSOR_PTSO( (0x01U << PIN_NUM) );\
  }else{\
    GPIO ## PORT_ID ## _PCOR = GPIO_PCOR_PTCO( (0x01U << PIN_NUM) );\
  }

```

```
/**
 * Writes the given value to the pins given in the MASK
 * @param PORT_ID the GPIO port id(A,B)
 * @param MASK 31 bit Mask with 1 in the bits
    corresponding to the pins of interest.
 * @param VAL pins value (GPIO_HIGH, GPIO_LOW)
 */
#define GPIO_WRITE_MASK(PORT_ID, MASK, VAL)\
    _GPIO_WRITE_MASK(PORT_ID, MASK, VAL)

#define _GPIO_WRITE_MASK(PORT_ID, MASK, VAL)\
  if(VAL == GPIO_HIGH){\
    GPIO ## PORT_ID ## _PSOR = GPIO_PSOR_PTSO(MASK);\
  }else{\
    GPIO ## PORT_ID ## _PCOR = GPIO_PCOR_PTCO(MASK);\
  }


/**
 * Reads the status of a GPIO PIN
 * @param PORT_ID the GPIO port id(A,B)
 * @param PIN_NUM pin number in port
 * @param VAL pin value (GPIO_HIGH, GPIO_LOW)
 */
#define GPIO_READ_PIN(PORT_ID, PIN_NUM)\
    _GPIO_READ_PIN(PORT_ID, PIN_NUM)

//Wrapper macro above is needed for argument expansion
    when using concatenation
#define _GPIO_READ_PIN(PORT_ID, PIN_NUM)\
    ((GPIO ## PORD_ID ## _PDIR & (0x01u << PIN_NUM)) >>
    PIN_NUM) )

#endif /* SOURCES_GPIO_GPIO_HAL_H_ */
```

7.10 ../Sources/Main/es670.c

```c
#include "fsl_device_registers.h"
#include "KL25Z/es670_peripheral_board.h"
#include "LedSwi/ledswi_hal.h"
#include "Mcg/mcg_hal.h"
#include "Buzzer/buzzer_hal.h"
#include "SevenSeg/sevenseg_hal.h"
#include "PIT/pit_hal.h"
#include "Util/util.h"


int main(void)
{
  mcg_clockInit();
  buzzer_init();
  ledswi_initLedSwitch(3,1);
  sevenseg_init();
  sevenseg_printHex(0xABCDu);
  unsigned short usPrintHex = 1;
  unsigned short usLedOn = 1;
  while(1){
    if(ledswi_getSwitchStatus(3) == SWITCH_ON){
      usPrintHex = !usPrintHex;
      usLedOn = !usLedOn;
      if(usPrintHex){
        sevenseg_printHex(0xABCDu);
      }else{
        sevenseg_printDec(0xABCDu);
      }
      if(usLedOn){
        ledswi_setLed(4);
      }else{
        ledswi_clearLed(4);
      }
    }
  }
```

```
    /* Never leave main */
    return 0;
}
```

7.11 ../Sources/Buzzer/buzzer˙hal.c

```
/*
    ************************************************************
     */
/* File name:        buzzer_hal.c
                 */
/* File description: File dedicated to the hardware
    abstraction layer*/
/*                   related buzzer from the peripheral
    board        */
/* Author name:      dloubach
                 */
/* Creation date:    12jan2016
                 */
/* Revision date:    13abr2016
                 */
/*
    ************************************************************
     */

#include "GPIO/gpio_hal.h"
#include "buzzer_hal.h"
#include "KL25Z/es670_peripheral_board.h"
#include "PIT/pit_hal.h"

/**
 * Initialize the buzzer device
 */
void buzzer_init(void)
{
  GPIO_UNGATE_PORT(BUZZER_PORT_ID);
```

```
  GPIO_INIT_PIN(BUZZER_PORT_ID, BUZZER_PIN, GPIO_OUTPUT);
  pit_enable();
}



/**
 * Clear the buzzer
 */
void buzzer_clearBuzz(void)
{
    GPIO_WRITE_PIN(BUZZER_PORT_ID, BUZZER_PIN, GPIO_LOW);
}



/**
 * Set the buzzer
 */
void buzzer_setBuzz(void)
{
  GPIO_WRITE_PIN(BUZZER_PORT_ID, BUZZER_PIN, GPIO_HIGH);
}

/**
 * Handler for buzzer interruptions
 */
void _buzzer_interrupt_handler(void){
  buzzer_setBuzz();
  buzzer_clearBuzz();
  pit_mark_interrupt_handled(BUZZER_PIT_TIMER_NUMB);
}

/**
 * Starts the buzzer with the specified period
 *
```

```
 * @param uiPeriod The period of the buzzer signal, in clock cycles (40MHz)
 */
void buzzer_initPeriodic(unsigned int uiPeriod){
  //Init timer 1
  pit_start_timer_interrupt(BUZZER_PIT_TIMER_NUMB, uiPeriod, &_buzzer_interrupt_handler);
}

/**
 * Stops any periodic buzzer signal
 */
void buzzer_stopPeriodic(void){
  pit_stop_timer_interrupt(BUZZER_PIT_TIMER_NUMB);
}
```

7.12 ../Sources/Buzzer/buzzer˙hal.h

```
/*
    ***************************************************************
     */
/* File name:        buzzer_hal.h
                 */
/* File description: Header file containing the functions /methods    */
/*                   interfaces for handling BUZZER from the          */
/*                   peripheral board
                 */
/* Author name:      dloubach
                 */
/* Creation date:    12jan2016
                 */
/* Revision date:    13abr2016
                 */
```

```c
/*
    ************************************************************
     */

#ifndef SOURCES_BUZZER_HAL_H_
#define SOURCES_BUZZER_HAL_H_

/**
 * Initialize the buzzer device
 */
void buzzer_init(void);


/**
 * Clear the buzzer
 */
void buzzer_clearBuzz(void);


/**
 * Set the buzzer
 */
void buzzer_setBuzz(void);

/**
 * Starts the buzzer with the specified period
 *
 * @param uiPeriod The period of the buzzer signal, in
    clock cycles (40MHz)
 */
void buzzer_initPeriodic(unsigned int uiPeriod);

/**
 * Stops any periodic buzzer signal
 */
void buzzer_stopPeriodic(void);
```

```
#endif /* SOURCES_BUZZER_HAL_H_ */
```

7.13 ../Sources/KL25Z/es670˙peripheral˙board.h

```
/*
    ************************************************************
     */
/* File name:        es670_peripheral_board.h
                 */
/* File description: Header file containing the
    peripherals mapping  */
/*                     of the peripheral board for the
    ES670 hardware*/
/* Author name:      dloubach
                 */
/* Creation date:    16out2015
                 */
/* Revision date:    25fev2016
                 */
/*
    ************************************************************
     */

#ifndef SOURCES_ES670_PERIPHERAL_BOARD_H_
#define SOURCES_ES670_PERIPHERAL_BOARD_H_

/* system includes */
#include <MKL25Z4.h>

/*                  General uC definitions
     */

/* Clock gate control */
#define   CGC_CLOCK_DISABLED          0x00U
```

```
#define  CGC_CLOCK_ENABLED              0x01U

/* GPIO DIRECTION     */
#define GPIO_OUTPUT            0x01U

/*  Workaround for PORT_ID macro expansion to stop at port level*/
typedef int A;
typedef int B;
typedef int C;
typedef int D;
typedef int E;

/*                  END OF General uC definitions                    */


/*                  BUZZER Definitions                               */
#define BUZZER_PORT_BASE_PNT          PORTD                    /* peripheral port base pointer */
#define BUZZER_GPIO_BASE_PNT          PTD                      /* peripheral gpio base pointer */
#define BUZZER_PORT_ID          D                /* peripheral port identifier*/

#define BUZZER_PIT_TIMER_NUMB     1

#define BUZZER_PIN                    1                        /* buzzer pin */
#define BUZZER_DIR                    kGpioDigitalOutput
#define BUZZER_ALT                    0x01u
/*                  END OF BUZZER definitions                        */

```

```
/*                    LED and SWITCH Definitions
            */
#define LS_PORT_BASE_PNT              PORTA
                        /* peripheral port base pointer */
#define LS_PORT_ID                    A
                        /* peripheral port identifier*/
#define LS_GPIO_BASE_PNT              PTA
                        /* peripheral gpio base pointer */

/* THIS PIN CONFLICTS WITH PTA1 USED AS UART0_RX IN THE
    OPENSDA SERIAL DEBUG PORT */
#define LS1_PIN                       1U
                        /* led/switch #1 pin */
#define LS1_DIR_OUTPUT                (GPIO_OUTPUT <<
    LS1_PIN)
#define LS1_DIR_INPUT                 (GPIO_OUTPUT <<
    LS1_PIN)
#define LS1_ALT                       0x01u
                        /* GPIO alternative */

/* THIS PIN CONFLICTS WITH PTA2 USED AS UART0_TX IN THE
    OPENSDA SERIAL DEBUG PORT */
#define LS2_PIN                       2U
                        /* led/switch #2 pin */
#define LS2_DIR_OUTPUT                (GPIO_OUTPUT <<
    LS2_PIN)
#define LS2_DIR_INPUT                 (GPIO_OUTPUT <<
    LS2_PIN)
#define LS2_ALT                       LS1_ALT

#define LS3_PIN                       4U
                        /* led/switch #3 pin */
#define LS3_DIR_OUTPUT                (GPIO_OUTPUT <<
    LS3_PIN)
#define LS3_DIR_INPUT                 (GPIO_OUTPUT <<
    LS3_PIN)
```

```
#define LS3_ALT                         LS1_ALT

#define LS4_PIN                         5U
                        /* led/switch #4 pin */
#define LS4_DIR_OUTPUT                  (GPIO_OUTPUT <<
    LS4_PIN)
#define LS4_DIR_INPUT                   (GPIO_OUTPUT <<
    LS4_PIN)
#define LS4_ALT                         LS1_ALT

/*                  END OF LED and SWITCH definitions
            */

/*                  SEVEN SEGMENT DISPLAY Definitions
                    */
#define SEV_SEG_PORT_BASE_PNT       PORTC
                        /* peripheral port base pointer */
#define SEV_SEG_PORT_ID             C
                        /* peripheral port identifier*/
#define SEV_SEG_GPIO_BASE_PNT       PTC
                        /* peripheral gpio base pointer */

#define SEV_SEG_PIT_TIMER_NUMB      0
    /* timer number for seven seg PIT */
#define SEVEN_SEG_PIT_PERIOD     0x0001E847
      /*125000 cycles = 3.125ms (40MHz)*/

#define SEGA_PIN                        0
                        /* Segment A*/
#define SEGA_DIR_OUTPUT                 (GPIO_OUTPUT <<
    SEGA_PIN)
#define SEGA_ALT                        0x01u
                        /* GPIO alternative */

#define SEGB_PIN                        1
```

```
#define SEGB_DIR_OUTPUT                (GPIO_OUTPUT <<
    SEGB_PIN)
#define SEGB_ALT                       SEGA_ALT

#define SEGC_PIN                       2
#define SEGC_DIR_OUTPUT                (GPIO_OUTPUT <<
    SEGC_PIN)
#define SEGC_ALT                       SEGA_ALT

#define SEGD_PIN                       3
#define SEGD_DIR_OUTPUT                (GPIO_OUTPUT <<
    SEGD_PIN)
#define SEGD_ALT                       SEGA_ALT

#define SEGE_PIN                       4
#define SEGE_DIR_OUTPUT                (GPIO_OUTPUT <<
    SEGE_PIN)
#define SEGE_ALT                       SEGA_ALT

#define SEGF_PIN                       5
#define SEGF_DIR_OUTPUT                (GPIO_OUTPUT <<
    SEGF_PIN)
#define SEGF_ALT                       SEGA_ALT

#define SEGG_PIN                       6
#define SEGG_DIR_OUTPUT                (GPIO_OUTPUT <<
    SEGG_PIN)
#define SEGG_ALT                       SEGA_ALT

#define SEGDP_PIN                      7
#define SEGDP_DIR_OUTPUT               (GPIO_OUTPUT <<
    SEGDP_PIN)
#define SEGDP_ALT                      SEGA_ALT

#define SEG_DISP1_PIN                  13
```

```
#define SEG_DISP1_DIR_OUTPUT            (GPIO_OUTPUT <<
    SEG_DISP1_PIN)
#define SEG_DISP1_ALT                   SEGA_ALT

#define SEG_DISP2_PIN                   12
#define SEG_DISP2_DIR_OUTPUT            (GPIO_OUTPUT <<
    SEG_DISP2_PIN)
#define SEG_DISP2_ALT                   SEGA_ALT

#define SEG_DISP3_PIN                   11
#define SEG_DISP3_DIR_OUTPUT            (GPIO_OUTPUT <<
    SEG_DISP3_PIN)
#define SEG_DISP3_ALT                   SEGA_ALT

#define SEG_DISP4_PIN                   10
#define SEG_DISP4_DIR_OUTPUT            (GPIO_OUTPUT <<
    SEG_DISP4_PIN)
#define SEG_DISP4_ALT                   SEGA_ALT

/*                  END of SEVEN SEGMENT DISPLAY
    Definitions                         */


#endif /* SOURCES_ES670_PERIPHERAL_BOARD_H_ */
```

7.14 ../Sources/Util/util.h

```
/*
    ***********************************************************
     */
/* File name:          util.h
                 */
/* File description: Header file containing the function/
    methods      */
/*                     prototypes of util.c
                 */
```

```
/*                    Those delays were tested under the
    following:   */
/*                    core clock @ 40MHz
                 */
/*                    bus clock @ 20MHz
                 */
/* Author name:       dloubach
                 */
/* Creation date:     09jan2015
                 */
/* Revision date:     13abr2016
                 */
/*
    ************************************************************
     */

#ifndef UTIL_H
#define UTIL_H

/**
 * generates ~ 088 micro sec
 */
void util_genDelay088us(void);


/**
 * generates ~   250 micro sec
 */
void util_genDelay250us(void);


/**
/* generates ~   1 mili sec
 */
void util_genDelay1ms(void);

```

```
/**
 * generates ~   10 mili sec
 */
void util_genDelay10ms(void);


#endif /* UTIL_H */
```

7.15 ../Sources/Util/util.c

```
/*
    ***************************************************************
     */
/* File name:        util.c
                */
/* File description: This file has a couple of useful
    functions to   */
/*                   make programming more productive
                */
/*
                */
/*                   Remarks: The soft delays consider
                */
/*                   core clock @ 40MHz
                */
/*                   bus clock @ 20MHz
                */
/*
                */
/* Author name:      dloubach
                */
/* Creation date:    09jan2015
                */
/* Revision date:    13abr2016
                */
```

```
/*
    ************************************************************
     */

#include "util.h"

/**
 * generates ~ 088 micro sec
 */
void util_genDelay088us(void)
{
    char i;
    for(i=0; i<120; i++)
    {
        __asm("NOP");
        __asm("NOP");
        __asm("NOP");
        __asm("NOP");
        __asm("NOP");
        __asm("NOP");
        __asm("NOP");
        __asm("NOP");
        __asm("NOP");
        __asm("NOP");
        __asm("NOP");
        __asm("NOP");
        __asm("NOP");
        __asm("NOP");
        __asm("NOP");
    }
}



/**
 * generates ~   250 micro sec
```

```
 */
void util_genDelay250us(void)
{
    char i;
    for(i=0; i<120; i++)
    {
        __asm("NOP");
        __asm("NOP");
        __asm("NOP");
        __asm("NOP");
        __asm("NOP");
        __asm("NOP");
        __asm("NOP");
        __asm("NOP");
        __asm("NOP");
        __asm("NOP");
    }
    util_genDelay088us();
    util_genDelay088us();
}



/**
/* generates ~   1 mili sec
 */
void util_genDelay1ms(void)
{
    util_genDelay250us();
    util_genDelay250us();
    util_genDelay250us();
    util_genDelay250us();
}


/**
```

```
 * generates ~   10 mili sec
 */
void util_genDelay10ms(void)
{
    util_genDelay1ms();
    util_genDelay1ms();
    util_genDelay1ms();
    util_genDelay1ms();
    util_genDelay1ms();
    util_genDelay1ms();
    util_genDelay1ms();
    util_genDelay1ms();
    util_genDelay1ms();
    util_genDelay1ms();
}
```