



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:

Daniel Dello Russo Oliveira

Davi Rodrigues

RA

101918

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-K125Z, 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


 ES670 - Projeto de Sistemas Embarcados Projeto Prático - 1o semestre/2015 Diagrama de Requisitos de Sistema		
«Requirement» Requisito TECLADO ID = REQ1A O sistema deve permitir o monitoramento de 4 chaves tipo "push button".	«Requirement» Requisito LED ID = REQ1B O sistema deve ser capaz de acionar 4 LEDs.	«Requirement» Requisito DISPLAY7SEG ID = REQ1C O sistema deve ser capaz de acionar 4 displays de 7 segmentos.
«Requirement» Requisito PROTOCOLO ID = REQ2 O sistema deve possuir um protocolo de comunicação serial capaz de receber comandos e retornar status dos periféricos do sistema.	«Requirement» Requisito LCD ID = REQ3 O sistema deve ser capaz de exibir mensagens alfanuméricas num LCD.	«Requirement» Requisito VELOCIDADE ID = REQ4 O sistema deve ser capaz de realizar medições de velocidade de rotação.
«Requirement» Requisito PWM ID = REQ5 O sistema deve ser capaz de gerar um sinal de PWM.	«Requirement» Requisito ADC ID = REQ6 O sistema deve ser capaz de realizar a leitura de sinal analógico e também a conversão A/D deste sinal.	«Requirement» Requisito CONTROLE ID = REQ7 O sistema deve ser capaz de realizar a funcionalidade de controlador digital.

Figura 1: Diagrama de requisitos

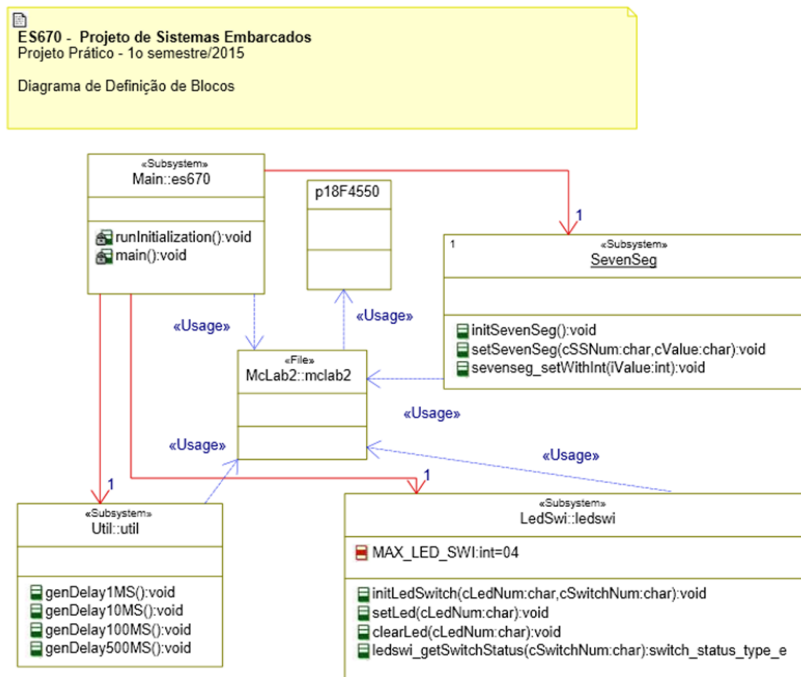


Figura 2: Diagrama de definição de blocos

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

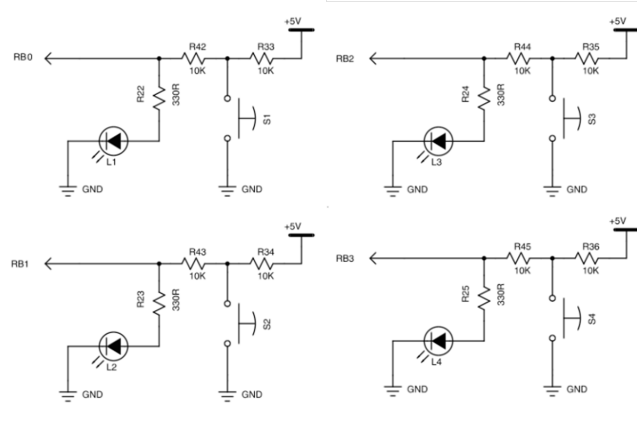


Figura 3: Esquema teclado e LEDs

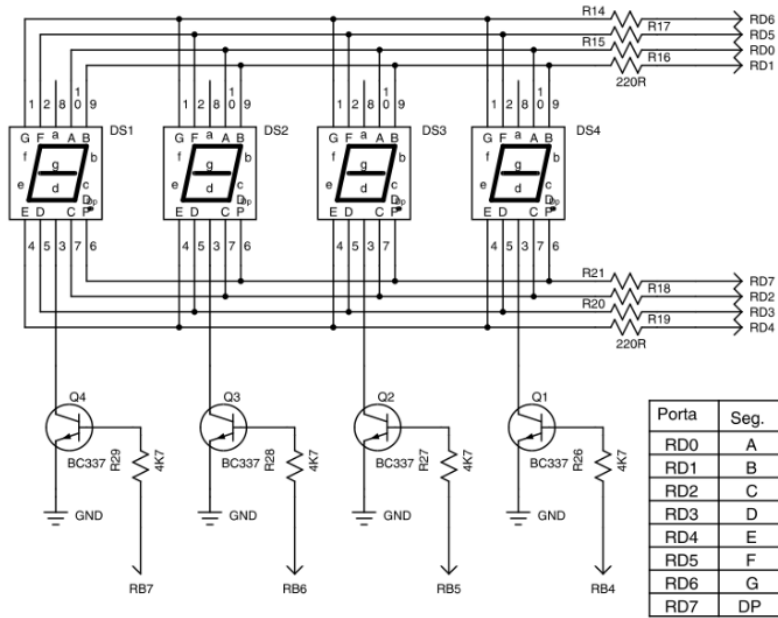


Figura 4: Esquema sete segmentos

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	<code>ledswi.c</code> - <code>void ledswi_initLedSwitch(char cLedNum, char cSwitchNum)</code> - <code>switch_status_type_e ledswi_getSwitchStatus(char cSwitchNum)</code>
REQ1B	<code>ledswi.c</code> - <code>void ledswi_initLedSwitch(char cLedNum, char cSwitchNum)</code> - <code>void ledswi_setLed(char cLedNum)</code> - <code>void ledswi_clearLed(char cLedNum)</code>
REQ1C	<code>sevenSeg.c</code> - <code>void sevenseg_init(void)</code> - <code>void sevenseg_set(char cSSNum, sevenseg_value_e eValue)</code> - <code>void sevenseg_setWithInt (int iValue)</code>

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

```
1  /*
    *****
    */
/* File name:      ledswi_hal.h
    */
3 /* File description: Header file containing the function/
   methods        */
/*                prototypes of ledswi.c
    */
5 /* Author name:   dloubach
    */
/* Creation date:   09jan2015
    */
7 /* Revision date:  13abr2016
    */
/*
    *****
    */
9
11 #ifndef SOURCES_LEDSWI_LEDSWI_HAL_H_
12 #define SOURCES_LEDSWI_LEDSWI_HAL_H_
13
14 #define MAX_LED_SWI      04
15
16 typedef enum
```

```

17     {
        SWITCHOFF,
        SWITCHON
19     } switch_status_type_e;

21     /**
        * As the hardware board was designed with LEDs/Switches
        * sharing
23     * the same pins, this method configures how many LEDS
        * and switches
        * will be available for the application
25     * @param cLedNum num of LEDs
        * @param cSwitchNum num of Switches (cLedNum +
        * cSwitchNum <= MAX_LED_SWI)
27     */
    void ledswi_initLedSwitch(char cLedNum, char cSwitchNum);

29

31     /**
        * set the led ON
33     * @param cLedNum which LED {1..4}
        */
35     void ledswi_setLed(char cLedNum);

37

39     /**
        * set the led OFF
        * @param cLedNum which LED {1..4}
41     */
    void ledswi_clearLed(char cLedNum);

43

45     /**
        * return the switch status
47     * @param cSwitchNum which switch
        * @return If the switch is ON or OFF

```



```

49  */
switch_status_type_e ledswi_getSwitchStatus(char
    cSwitchNum);
51
53 #endif /* SOURCES_LEDSWI_LEDSWI_HAL_H */

```

7.2 ../Sources/LedSwi/ledswi_hal.c

```

1  /*
    *****
    */
/* File name:      ledswi_hal.c
    */
3 /* File description: This file has a couple of useful
    functions to */
/*                control LEDs and Switches from
    peripheral board */
5 /* Author name:    dloubach
    */
/* Creation date:    20jan2015
    */
7 /* Revision date:   13abr2016
    */
/*
    *****
    */
9
#include "ledswi_hal.h"
11 #include "KL25Z/es670_peripheral_board.h"
13 #define USING_OPENSDA_DEBUG
15 /**
    * As the hardware board was designed with LEDs/Switches
    sharing

```

```

17  * the same pins, this method configures how many LEDS
    and switches
    * will be available for the application
19  * @param cLedNum num of LEDS
    * @param cSwitchNum num of Switches (cLedNum +
        cSwitchNum <= MAX_LED_SWI)
21  */
void ledswi_initLedSwitch(char cLedNum, char cSwitchNum)
23 {
    /* un-gate port clock*/
25     SIM_SCGC5 = SIM_SCGC5_PORTA(CGC_CLOCK_ENABLED);

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

35
    /* check if the number to configured is according to
37     hardware dev kit */
    if((cLedNum + cSwitchNum) <= MAX_LED_SWI)
39     {
        /* max number of peripherals to configure is ok,
        carry on */
41         switch(cSwitchNum)
        {
43             case 0:
                /* no switches in system configuration */
                /* all leds */
45                 GPIOA_PDDR |= GPIO_PDDR_PDD(
                    LS1_DIR_OUTPUT | LS2_DIR_OUTPUT | LS3_DIR_OUTPUT |
                    LS4_DIR_OUTPUT);
47                 break;

```

```

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

           case 2:
           /* just 2 switches */
           GPIOA_PDDR |= GPIO_PDDR_PDD(
55     LS3_DIR_OUTPUT | LS4_DIR_OUTPUT);
           GPIOA_PDDR &= ~GPIO_PDDR_PDD(
57     LS1_DIR_INPUT | LS2_DIR_INPUT);
           break;
59         case 3:
           /* 3 switches */
           GPIOA_PDDR |= GPIO_PDDR_PDD(
61     LS4_DIR_OUTPUT);
           GPIOA_PDDR &= ~GPIO_PDDR_PDD(
63     LS1_DIR_INPUT | LS2_DIR_INPUT | LS3_DIR_INPUT);
           break;
65         case 4:
           /* 4 switches */
           GPIOA_PDDR &= ~GPIO_PDDR_PDD(
67     LS1_DIR_INPUT | LS2_DIR_INPUT | LS3_DIR_INPUT |
69     LS4_DIR_INPUT);
           break;
71     } /* switch(cSwitchNum) */

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

75 }

```

```

77
79 /**
   * set the led ON
81 * @param cLedNum which LED {1..4}
   */
83 void ledswi_setLed(char cLedNum)
   {
85     /* sanity check */
     if (cLedNum <= MAX_LED_SWI)
87     {
         switch (cLedNum)
89         {
             case 1:
101             GPIOA_PSOR = GPIO_PSOR_PTISO( (0x01U <<
LS1_PIN) );
                 break;
103             case 2:
                 GPIOA_PSOR = GPIO_PSOR_PTISO( (0x01U <<
LS2_PIN) );
105                 break;
                 case 3:
107                 GPIOA_PSOR = GPIO_PSOR_PTISO( (0x01U <<
LS3_PIN) );
                 break;
                 case 4:
                 GPIOA_PSOR = GPIO_PSOR_PTISO( (0x01U <<
LS4_PIN) );
                 break;
             } /* switch(cLedNum) */
         } /* if (cLedNum <= MAX_LED_SWI) */
     }
}

```

```

109 /**
    * set the led OFF
111 * @param cLedNum which LED {1..4}
    */
113 void ledswi_clearLed(char cLedNum)
    {
115     /* sanity check */
    if(cLedNum <= MAX_LED_SWI)
117     {
        switch(cLedNum)
119         {
            case 1:
121             GPIOA_PCOR = GPIO_PCOR_PTCO( (0x01U <<
LS1_PIN) );
                break;
123             case 2:
                GPIOA_PCOR = GPIO_PCOR_PTCO( (0x01U <<
LS2_PIN) );
                break;
125             case 3:
                GPIOA_PCOR = GPIO_PCOR_PTCO( (0x01U <<
LS3_PIN) );
                break;
127             case 4:
                GPIOA_PCOR = GPIO_PCOR_PTCO( (0x01U <<
LS4_PIN) );
                break;
129             } /* switch(cLedNum) */
131     } /* if(cLedNum <= MAX_LED_SWI) */
133 }
135 }
137
139 /**

```

```

141  * return the switch status
142  * @param cSwitchNum which switch
143  * @return If the switch is ON or OFF
144  */
145  switch_status_type_e ledswi_getSwitchStatus(char
    cSwitchNum)
146  {
147      switch_status_type_e sstReturn = SWITCH_OFF;
148
149      /* sanity check */
150      if (cSwitchNum <= MAX_LED_SWI)
151      {
152          switch (cSwitchNum)
153          {
154              case 1:
155                  if (SWITCH_ON == ((GPIOA_PDIR &
156                      LS1_DIR_INPUT) >> LS1_PIN) )
157                      sstReturn = SWITCH_ON;
158                  break;
159
160              case 2:
161                  if (SWITCH_ON == ((GPIOA_PDIR &
162                      LS2_DIR_INPUT) >> LS2_PIN) )
163                      sstReturn = SWITCH_ON;
164                  break;
165
166              case 3:
167                  if (SWITCH_ON == ((GPIOA_PDIR &
168                      LS3_DIR_INPUT) >> LS3_PIN) )
169                      sstReturn = SWITCH_ON;
170                  break;
171
172              case 4:
173                  if (SWITCH_ON == ((GPIOA_PDIR &
174                      LS4_DIR_INPUT) >> LS4_PIN) )
175                      sstReturn = SWITCH_ON;
176          }
177      }
178  }

```

```

171         break;

173     } /* switch(cSwitchNum) */

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

177 /* return the result */
178 return(sstReturn);
179 }

```

7.3 ../Sources/Mcg/mcg_hal.h

```

/*
*****
*/
2 /* File name:      mcg_hal.h
   */
/* File description: Header file containing the functions
   /methods */
4 /*
   interfaces for handling the
   Multipurpose clock */
/*
   generator module
   */
6 /* Author name:     dloubach
   */
/* Creation date:     21out2015
   */
8 /* Revision date:    25fev2016
   */
/*
*****
*/
10
12 #ifndef SOURCES_MCG_HAL_H
#define SOURCES_MCG_HAL_H

```

```

14 /* ***** */
/* Method name:      mcg_clockInit */
16 /* Method description: main board clk configuration */
/* Input params:      n/a */
18 /* Output params:    n/a */
/* ***** */
20 void mcg_clockInit(void);
22
24 #endif /* SOURCES_MCG_HAL_H */

```

7.4 ../Sources/Mcg/mcg_hal.c

```

/*
*****
*/
2 /* File name:      mcg_hal.c
   */
/* File description: Multipurpose clk generator hardware
   abstraction */
4 /*
   layer. Enables the clock
   configuration */
/*
   */
6 /*
   Modes of Operation
   */
/*
   FLL Engaged Internal (FEI) =
   DEFAULT */
8 /*
   FLL Engaged External (FEE)
   */
/*
   FLL Bypassed Internal (FBI)
   */
10 /*
   FLL Bypassed External (FBE)
   */

```



```

12 /*          PLL Engaged External (PEE)
    */
13 /*          PLL Bypassed External (PBE)
    */
14 /*          Bypassed Low Power Internal (BLPI)
    */
15 /*          Bypassed Low Power External (BLPE)
    */
16 /*          Stop
    */
17 /*          For clock definitions , check the
chapter */
18 /*          5.4 Clock definitions from
    */
19 /*          KL25 Sub-Family Reference Manual
    */
20 /*
    */
21 /* Author name:      dloubach
    */
22 /* Creation date:    21out2015
    */
23 /* Revision date:    13abr2016
    */
24 /*
*****
    */

26 #include "mcg_hal.h"

28 /* systems include */
#include "fsl_smc_hal.h"
30 #include "fsl_port_hal.h"
#include "fsl_clock_manager.h"

```

```

32
/* EXTAL0 PTA18 */
34 #define EXTAL0_PORT          PORTA
   #define EXTAL0_PIN          18U
36 #define EXTAL0_PINMUX        kPortPinDisabled

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

42
/* OSC0 configuration */
44 #define OSC0_INSTANCE        0U
   #define OSC0_XTAL_FREQ      8000000U /* 08 MHz*/
46 #define OSC0_SC2P_ENABLE_CONFIG false
   #define OSC0_SC4P_ENABLE_CONFIG false
48 #define OSC0_SC8P_ENABLE_CONFIG false
   #define OSC0_SC16P_ENABLE_CONFIG false
50 #define MCG_HGO0             kOscGainLow
   #define MCG_RANGE0          kOscRangeVeryHigh
52 #define MCG_EREFS0           kOscSrcOsc

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

62
/* RTC.CLKIN PTC1 */
64 #define RTC_CLKIN_PORT        PORTC
   #define RTC_CLKIN_PIN        1U
66 #define RTC_CLKIN_PINMUX      kPortMuxAsGpio

```

```

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

72 #ifndef CLOCK_INIT_CONFIG
    #define CLOCK_INIT_CONFIG CLOCK_RUN
74 #endif

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

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

92         .pll0EnableInFllMode = false,                // PLL0
            disable

```

```

    .pll0EnableInStop = false ,           // PLL0
    disable in STOP mode
94     .prdiv0          = 0U,
    .vdiv0             = 0U,
96 },
    .simConfig =
98 {
    .pllFllSel = kClockPl1FllSelFll ,     //
    PLLFLLSEL select FLL
100     .er32kSrc = kClockEr32kSrcLpo ,    //
    ERCLK32K selection , use LPO
    .outdiv1     = 0U,
102     .outdiv4   = 4U,
    },
104     .oscerConfig =
    {
106         .enable      = true ,           //
        OSCERCLK enable
        .enableInStop = false ,           //
        OSCERCLK disable in STOP mode
108     }
};

110 /* Configuration for enter RUN mode, Core clock = 40 MHz
    */
112 /*
    * 24.5.1.1 Initializing the MCG
114 * KL25 Sub-Family Reference Manual, Rev. 3, September
    2012
    *
116 * Refer also to
    * Table 24-18. MCG modes of operation
118 *
    * On L-series devices the MCGFLLCLK frequency is limited
    to 48 MHz max

```

```

120  * The DCO is limited to the two lowest range settings (
    MCG_C4[DRST_DRS] must be set to either 0b00 or 0b01).
    *
122  * FEE (FLL engaged external)
    * fext / FLL_R must be in the range of 31.25 kHz to
      39.0625 kHz
124  * 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
126  *
    * (fext / FLL_R) * F = (8 MHz / 256 ) * 1280 = 40 MHz
128  *
    * */
130  const clock_manager_user_config_t g_defaultClockConfigRun
    =
    {
132      /* ----- multipurpose clock generator
    configurations ----- */
        .mcgConfig =
134        {
            .mcg_mode          = kMcgModeFEE,          // Work
            in FEE mode
136
            /* ----- MCGIRCCLK settings
    ----- */
138            .irclkEnable      = true ,                //
            MCGIRCCLK enable
            .irclkEnableInStop = false ,                //
            MCGIRCCLK disable in STOP mode
140            .ircs             = kMcgIrcSlow ,         //
            Select IRC 32kHz
            .fcrdiv            = 0U,                    //
            FCRDIV is 0
142

```

```

/* ----- MCG FLL settings
----- */
144     .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
146     .drs      = kMcgDcoRangeSelMid,           //
frequency range
     .dmx32     = kMcgDmx32Default,           // DCO
has a default range of 25%

148     /* ----- MCG PLL settings
----- */
150     .pll0EnableInFllMode = false,           // PLL0
disable
     .pll0EnableInStop  = false,           // PLL0
disabLe in STOP mode
152     .prdiv0          = 0x0U,
     .vdiv0             = 0x0U,
154     },
/* ----- system integration module
configurations ----- */
156     .simConfig =
{
158     .pllFllSel = kClockPl1FllSelFll,           //
PLLFLLSEL select PLL
     .er32kSrc   = kClockEr32kSrcLpo,           //
ERCLK32K selection, use LPO
160     .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
162     },

```

```

164     /* ----- system oscillator output
configurations ----- */
164     .oscerConfig =
166     {
166         .enable          = true ,                //
OSCERCLK enable
166         .enableInStop = false ,                //
OSCERCLK disable in STOP mode
168     }
168 };
170
172 /**
172  * Oscillator configuration
174  */
174 void mcg_initOsc0(void)
176 {
176     /* OSC0 configuration */
178     osc_user_config_t osc0Config =
178     {
180         .freq              = OSC0_XTAL_FREQ,
180         .hgo               = MCG_HGO0,
182         .range             = MCG_RANGE0,
182         .erefs             = MCG_EREFS0,
184         .enableCapacitor2p = OSC0_SC2P_ENABLE_CONFIG,
184         .enableCapacitor4p = OSC0_SC4P_ENABLE_CONFIG,
186         .enableCapacitor8p = OSC0_SC8P_ENABLE_CONFIG,
186         .enableCapacitor16p = OSC0_SC16P_ENABLE_CONFIG,
188     };
188
190     /* oscillator initialization */
190     CLOCK_SYS_OscInit(OSC0_INSTANCE, &osc0Config);
192 }
194

```

```

196  /**
    * Function to initialize RTC external clock base on
    * board configuration
198  */
void mcg_initRtcOsc(void)
200 {

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

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

212     rtc_osc_user_config_t rtcOscConfig =
214     {
        .freq                = RTC_XTAL_FREQ,
216        .enableCapacitor2p   = RTC_SC2P_ENABLE_CONFIG,
        .enableCapacitor4p   = RTC_SC4P_ENABLE_CONFIG,
218        .enableCapacitor8p   = RTC_SC8P_ENABLE_CONFIG,
        .enableCapacitor16p  = RTC_SC16P_ENABLE_CONFIG,
220        .enableOsc           = RTC_OSC_ENABLE_CONFIG,
    };

222
    /* OSC RTC initialization */
224    CLOCK_SYS_RtcOscInit(0U, &rtcOscConfig);
}

```



```

226
228
229 /**
230  * System clock configuration
231  */
232 void mcg_initSystemClock(void)
233 {
234     /* Set system clock configuration. */
235     #if (CLOCK_INIT_CONFIG == CLOCK_VLPR)
236         CLOCK_SYS_SetConfiguration(&
g_defaultClockConfigVlpr);
237     #else
238         CLOCK_SYS_SetConfiguration(&
g_defaultClockConfigRun);
239     #endif
240 }
241
242
243
244 /* ***** */
245 /* Method name:      mcg_clockInit */
246 /* Method description: main board clk configuration */
247 /* Input params:      n/a */
248 /* Output params:     n/a */
249 /* ***** */
250 void mcg_clockInit(void)
251 {
252     /* enable clock for PORTs */
253     CLOCK_SYS_EnablePortClock(PORTA_IDX);
254     CLOCK_SYS_EnablePortClock(PORTC_IDX);
255     CLOCK_SYS_EnablePortClock(PORTE_IDX);
256
257     /* set allowed power mode to allow all */
258     SMC_HAL_SetProtection(SMC, kAllowPowerModeAll);

```

```

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

264  /* setup OSC0 */
    mcg_initOsc0();

266

268  /* setup OSC RTC */
    mcg_initRtcOsc();

270  /* setup system clock */
    mcg_initSystemClock();
272 }

```

7.5 ../Sources/PIT/pit_hal.h

```

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

```

```

/*
*****
*/

10
#ifndef SOURCES_PIT_PIT_HAL_H_
12 #define SOURCES_PIT_PIT_HAL_H_

14 /**
 * Enables Periodic Interruption Timer module.
16 * (With the stop on debug flag set to on)
 */
18 void pit_enable(void);

20 /**
 * Start interruptions for given timer, unchained mode.
22 * Timer interruptions are masked.
 *
24 * @param usTimer_num The number for the desired timer
 * (0,1)
 * @param uiTimer_period The number of bus_clock cycles
 * between interrupts
26 * @param fpInterrupt_handler Timer interrupt handler
 * routine address pointer
 */
28 void pit_start_timer_interrupt(unsigned short
 * usTimer_num, unsigned int uiTimer_period, void (*
 * fpInterrupt_handler)(void));

30 /**
 * Stop interruptions for given timer, unchained mode.
32 *
 * @param usTimer_num The number for the desired timer
 * (0,1)
34 */
void pit_stop_timer_interrupt(unsigned short usTimer_num
);

```

```

36  /**
38  * Mark interruption as handled for the given timer, this
    should be called by timer
    * interruption handlers once they are finished.
40  *
    * @param usTimer_numb  The number for the desired timer
    (0,1)
42  */
void pit_mark_interrupt_handled(unsigned short
    usTimer_numb);
44
    /**
46  * Pit interruption handler. Checks what timer caused the
    interruption and call the
    * correct timer interruption handler.
48  */
void PIT_IRQHandler(void);
50 #endif /* SOURCES_PIT_PIT_HAL_H */

```

7.6 ../Sources/PIT/pit_hal.c

```

    /*
        *****
        */
2  /* File name:      pit_hal.c
        */
    /* File description: File containing the functions/
        methods      */
4  /*
        for handling the Periodic
        Interruption    */
    /*
        Timer module
        */
6  /* Author name:    ddello
        */

```

```

/* Creation date:    10abr2016
    */
8 /* Revision date:    13abr2016
    */

/*
    *****
    */
10 //Careful when handling PIT DOC! Bit endianness is
    inverted in relation to GPIO doc

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

14 #define PIT_IRQ_NUMBER PIT_IRQn

16 /**
18  *Default timer interruption handler. Does nothing.
    */
20 static void _nop_handler(void){
    PIT_TFLG0 |= PIT_TFLG_TIF(0x1u);
22    PIT_TFLG1 |= PIT_TFLG_TIF(0x1u);
}

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

28 /**
    * Pit interruption handler. Checks what timer caused the
        interruption and call the
30    * correct timer interruption handler.
    */
32 void PIT_IRQHandler(void){
    if(PIT_TFLG0){
34        (*fpTimer0Handler)();
    }
36    if(PIT_TFLG1){

```

```

        (*fpTimer1Handler)();
38     }
    }

40
    /**
42     * Enables Periodic Interruption Timer module.
    * (With the stop on debug flag set to on)
44     */
    void pit_enable(void){
46         SIM_SCGC6 |= SIM_SCGC6_PIT_MASK;
        PIT_MCR &= ~PIT_MCR_MDIS(0x1u);
48         //Freeze in debug mode
        PIT_MCR |= PIT_MCR_FRZ(0x1u);
50         NVIC_ClearPendingIRQ(PIT_IRQ_NUMBER);
        NVIC_EnableIRQ(PIT_IRQ_NUMBER);
52     }

54     /**
    * Start interruptions for given timer, unchained mode.
56     * Timer interruptions are masked.
    *
58     * @param usTimer_numb The number for the desired timer
        (0,1)
    * @param uiTimer_period The number of bus_clock cycles
        between interrupts
60     * @param fpInterrupt_handler Timer interrupt handler
        routine address pointer
    */
62     void pit_start_timer_interrupt(unsigned short
        usTimer_numb, unsigned int uiTimer_period, void (*
        fpInterrupt_handler)(void)){
        if(!usTimer_numb){
64             timer0Handler = fpInterrupt_handler;
            PIT_LDVAL0 = PIT_LDVAL_TSV(uiTimer_period);
66             PIT_TCTRL0 &= ~PIT_TCTRL_CHN(0x1u);    /*Disable chain
                mode*/

```

```

        PIT_TCTRL0 |= PIT_TCTRL_TIE(0x1u);    /*Enable
interrupts for timer 0*/
68     PIT_TCTRL0 |= PIT_TCTRL_TEN(0x1u);    /*Enable timer
0*/
    }else{
70         timer1Handler = fpInterrupt_handler;
        PIT_LDVAL1 = PIT_LDVAL_TSV(uiTimer_period);
72         PIT_TCTRL1 &= ~PIT_TCTRL_CHN(0x1u);    /*Disable chain
mode*/
        PIT_TCTRL1 |= PIT_TCTRL_TIE(0x1u);    /*Enable
interrupts for timer 1*/
74         PIT_TCTRL1 |= PIT_TCTRL_TEN(0x1u);    /*Enable timer
1*/
    }
76 }

78 /**
 * Stop interruptions for given timer , unchained mode.
80 *
 * @param usTimer_numb  The number for the desired timer
 *                       (0,1)
82 */
void pit_stop_timer_interrupt(unsigned short usTimer_numb
    ){
84     if (!usTimer_numb){
        PIT_TCTRL0 &= ~PIT_TCTRL_TIE(0x1u);
86         PIT_TCTRL0 &= ~PIT_TCTRL_TEN(0x1u);
    }else{
88         PIT_TCTRL1 &= ~PIT_TCTRL_TIE(0x1u);
        PIT_TCTRL1 &= ~PIT_TCTRL_TEN(0x1u);
90     }
    }
92

94 /**
 * Mark interruption as handled for the given timer , this
   should be called by timer

```

```

    * interruption handlers once they are finished.
96  *
    * @param usTimer_numb  The number for the desired timer
    *                       (0,1)
98  */
void pit_mark_interrupt_handled(unsigned short
    usTimer_numb){
100  if (!usTimer_numb){
    PIT_TFLG0 |= PIT_TFLG_TIF(0x1u);
102  } else{
    PIT_TFLG1 |= PIT_TFLG_TIF(0x1u);
104  }
}

```

7.7 ../Sources/SevenSeg/sevensseg_hal.h

```

1  /*
    *****
    */
    /* File name:      sevensseg_hal.h
    */
3  /* File description: Header file containing the functions
    /methods */
    /*                interfaces for handling SEVEN
    SEGMENT DISPLAY */
5  /*                from the peripheral board
    */
    /* Author name:    ddello
    */
7  /* Creation date:   18mar2016
    */
    /* Revision date:   13abr2016
    */
9  /*
    *****
    */

```



```

11 #ifndef SOURCES_SEVEN_SEGMENT_HAL_H_
12 #define SOURCES_SEVEN_SEGMENT_HAL_H_
13
14 #include "KL25Z/es670_peripheral_board.h"
15
16 #define MAX_SEGMENT_NUMBER 8
17 #define MAX_DISP_NUMBER 4
18
19
20 typedef enum
21 {
22     SEG_A = SEG_A_PIN,
23     SEG_B = SEG_B_PIN,
24     SEG_C = SEG_C_PIN,
25     SEG_D = SEG_D_PIN,
26     SEG_E = SEG_E_PIN,
27     SEG_F = SEG_F_PIN,
28     SEG_G = SEG_G_PIN,
29     SEG_DP = SEG_DP_PIN,
30     SEG_END = -1
31 } seven_segment_seg_type_e;
32
33 typedef enum
34 {
35     DISP_1 = SEG_DISP1_PIN,
36     DISP_2 = SEG_DISP2_PIN,
37     DISP_3 = SEG_DISP3_PIN,
38     DISP_4 = SEG_DISP4_PIN,
39 } seven_segment_disp_type_e;
40
41 /**
42  * Initialize the seven segment display
43  */
44 void sevenseg_init(void);
45

```

```

47  /**
   * 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)
49  */
void sevenseg_setSegs(seven_segment_seg_type_e*
    epSet_segments);

51
52  /**
53  * Shows the value written in the segment pins to the
   * given display after clearing the others
55  * @param eDisplay the display to initialize.
   */
57 void sevenseg_setDisp(seven_segment_disp_type_e eDisplay)
    ;

59  /**
   * Shows the passed value in hexadecimal format in the
   * seven segment display.
61  * @param uiHex the value to be printed
   */
63 void sevenseg_printHex(unsigned int uiHex);

65  /**
   * Shows the passed value in decimal format in the seven
   * segment display.
67  * @param uiDec the value to be printed
   */
69 void sevenseg_printDec(unsigned int uiDec);

71  /**
   * Converts the less significative decimal digit of the
   * argument into it's seven
73  * segment display configuration

```

```

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

81 /**
    * Converts the less significant hexadecimal digit of
      the argument into it's seven
83 * segment display configuration
    * @param usHex the value to be converted (-1 if none
      should be displayed)
85 * @param epRet address for results (should be a
      allocated array of minimal 9 elements)
    *
87 * @return epRet
    */
89 seven_segment_seg_type_e* sevenseg_hex2segArray(unsigned
      short usHex, seven_segment_seg_type_e* epRet);

91 #endif /* SOURCES_SEVEN_SEGMENT_HAL_H */

```

7.8 ../Sources/SevenSeg/sevenseg_hal.c

```

1  /*
      *****
    */
/* File name:          sevenseg_hal.c
      */
3 /* File description: File containing the functions/
      methods          */

```

```

/*          for handling SEVEN SEGMENT DISPLAY
          */
5 /*          from the peripheral board
          */
/* Author name:      ddello
          */
7 /* Creation date:   18mar2016
          */
/* Revision date:    13abr2016
          */
9 /*
   *****
   */

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

17 #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

19
static unsigned short usIsHex = 0;
21 static unsigned int uiPrintVal = -1;

23 /**
   * Interrupt handler for updating in display
   configuration
25 */

```

```

void _sevenseg_interrupt_handler(void){
27   static seven_segment_disp_type_e epDisplays[] = {DISP_1
      , DISP_2, DISP_3, DISP_4};
   static seven_segment_seg_type_e epSeg_array[9];
29   static unsigned short usCur_disp = 0;
   if(usIsHex){
31       sevenseg_hex2segArray(uiPrintVal/pow(16,usCur_disp),
          epSeg_array);
   }else{
33       sevenseg_dec2segArray(uiPrintVal/pow(10,usCur_disp),
          epSeg_array);
   }
35   sevenseg_setSegs(epSeg_array);
   sevenseg_setDisp(epDisplays[usCur_disp]);
37   usCur_disp = (usCur_disp+1)%4;
   pit_mark_interrupt_handled(SEV_SEG_PIT_TIMER_NUMB);
39 }

41 /**
   * Initialize the seven segment display
43 */
void sevenseg_init(void){
45   GPIO_UNGATE_PORT(SEV_SEG_PORT_ID);

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

```

```

57  GPIO_INIT_PIN(SEV_SEG_PORT_ID, SEG_DISP1_PIN,
    GPIO_OUTPUT);
    GPIO_INIT_PIN(SEV_SEG_PORT_ID, SEG_DISP2_PIN,
    GPIO_OUTPUT);
59  GPIO_INIT_PIN(SEV_SEG_PORT_ID, SEG_DISP3_PIN,
    GPIO_OUTPUT);
    GPIO_INIT_PIN(SEV_SEG_PORT_ID, SEG_DISP4_PIN,
    GPIO_OUTPUT);

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

67

69  /**
    * Sets only the selected segments as high. Setting the
    * others as low
71  * @param epDet_segments = Array with the segments that
    * should be set as on (Last element should be SEG.END)
    */
73 void sevenseg_setSegs(seven_segment_seg_type_e*
    epSet_segments){
    //Clear all segments.
75  GPIO_WRITE_MASK(SEV_SEG_PORT_ID, SEV_SEG_SEGMENT_MASK,
    GPIO_LOW);
    //Set the selected segments to high
77  for(unsigned short usCounter = 0; epSet_segments[
    usCounter] != SEG.END; usCounter++){
    GPIO_WRITE_PIN(SEV_SEG_PORT_ID, epSet_segments[
    usCounter], GPIO_HIGH);
79  }
}

81

```

```

83  /**
   * Shows the value written in the segment pins to the
   *   given display after clearing the others
85  * @param eDisplay the display to initialize.
   */
87 void sevenseg_setDisp(seven_segment_disp_type_e eDisplay)
   {
   //Clear all displays
89   GPIO_WRITE_MASK(SEV_SEG_PORT_ID, SEV_SEG_DISP_MASK,
        GPIO_LOW);
   //Activate the selected display
91   GPIO_WRITE_PIN(SEV_SEG_PORT_ID, eDisplay, GPIO_HIGH);
   }
93
95  /**
   * Shows the passed value in hexadecimal format in the
   *   seven segment display.
   * @param uiHex the value to be printed
97  */
   void sevenseg_printHex(unsigned int uiHex){
99     usIsHex = 1;
        uiPrintVal = uiHex;
101 }
103
105  /**
   * Shows the passed value in decimal format in the seven
   *   segment display.
   * @param uiDec the value to be printed
   */
107 void sevenseg_printDec(unsigned int uiDec){
        usIsHex = 0;
109     uiPrintVal = uiDec;
   }
111
   /**

```

```

113  * Converts the less significative decimal digit of the
      argument into it's seven
      * segment display configuration
115  * @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)
117  *
      * @return epRet
119  */
      seven_segment_seg_type_e* sevenseg_dec2segArray( unsigned
          short usDec, seven_segment_seg_type_e* epRet){
121  if( usDec < 0){
          epRet[0] = SEG_END;
123  return epRet;
      }
125  epRet[0] = SEG_A;
      epRet[1] = SEG_B;
127  epRet[2] = SEG_C;
      epRet[3] = SEG_D;
129  epRet[4] = SEG_E;
      epRet[5] = SEG_F;
131  epRet[6] = SEG_G;
      epRet[7] = SEG_END;
133  switch( usDec%10){
      case 0:
135      //{SEG_A,SEG_B,SEG_C,SEG_D,SEG_G,SEG_E,SEG_F,SEG_END
          };
          epRet[7] = SEG_END;
137      break;
      case 1:
139      //{SEG_B,SEG_C,SEG_END};
          epRet[0] = SEG_B;
141      epRet[1] = SEG_C;
          epRet[2] = SEG_END;
143      break;

```



```

145     case 2:
        // {SEG_A, SEG_B, SEG_G, SEG_D, SEG_E, SEG_END};
        epRet [2] = SEG_G;
147     epRet [5] = SEG_END;
        break;
149     case 3:
        // {SEG_A, SEG_B, SEG_C, SEG_D, SEG_G, SEG_END}
151     epRet [4] = SEG_G;
        epRet [5] = SEG_END;
153     break;
    case 4:
155     // {SEG_G, SEG_B, SEG_C, SEG_F, SEG_END}
        epRet [0] = SEG_G;
157     epRet [3] = SEG_F;
        epRet [4] = SEG_END;
159     break;
    case 5:
161     // {SEG_A, SEG_G, SEG_C, SEG_D, SEG_F, SEG_END}
        epRet [1] = SEG_G;
163     epRet [4] = SEG_F;
        epRet [5] = SEG_END;
165     break;
    case 6:
167     // {SEG_A, SEG_G, SEG_C, SEG_D, SEG_E, SEG_F, SEG_END}
        epRet [1] = SEG_G;
169     epRet [6] = SEG_END;
        break;
171     case 7:
        // {SEG_A, SEG_B, SEG_C, SEG_END}
173     epRet [3] = SEG_END;
        break;
175     case 8:
        // {SEG_A, SEG_B, SEG_C, SEG_D, SEG_E, SEG_F, SEG_G, SEG_END}
177     break;
    case 9:
179     // SEG_A, SEG_B, SEG_C, SEG_F, SEG_G, SEG_END}

```

```

    epRet[3] = SEG_F;
181    epRet[4] = SEG_G;
    epRet[5] = SEG_END;
183    break;
}
185    return epRet;
}
187
/**
189 * Converts the less significative hexadecimal digit of
    the argument into it's seven
    * segment display configuration
191 * @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)
    *
193 * @return epRet
195 */
seven_segment_seg_type_e* sevenseg_hex2segArray(unsigned
    short usHex, seven_segment_seg_type_e* epRet){
197    if(usHex < 0){
        epRet[0] = SEG_END;
199        return epRet;
    }
201    epRet[0] = SEG_A;
    epRet[1] = SEG_B;
203    epRet[2] = SEG_C;
    epRet[3] = SEG_D;
205    epRet[4] = SEG_E;
    epRet[5] = SEG_F;
207    epRet[6] = SEG_G;
    epRet[7] = SEG_END;
209    switch(usHex%16){
        case 0:
211        case 1:

```

```

213     case 2:
214     case 3:
215     case 4:
216     case 5:
217     case 6:
218     case 7:
219     case 8:
220     case 9:
221         return sevenseg_dec2segArray(usHex%16, epRet);
222         break;
223     case 10: //A
224         //{SEG_A,SEG_B,SEG_C,SEG_G,SEG_E,SEG_F,SEG_END}
225         epRet[3] = SEG_G;
226         epRet[6] = SEG_END;
227         break;
228     case 11: //B (b)
229         //{SEG_G,SEG_F,SEG_C,SEG_D,SEG_E,SEG_END}
230         epRet[0] = SEG_G;
231         epRet[1] = SEG_F;
232         epRet[5] = SEG_END;
233         break;
234     case 12: //C
235         //{SEG_A,SEG_E,SEG_F,SEG_D,SEG_END}
236         epRet[1] = SEG_E;
237         epRet[2] = SEG_F;
238         epRet[4] = SEG_END;
239         break;
240     case 13: //D (d)
241         //{SEG_G,SEG_B,SEG_C,SEG_D,SEG_E,SEG_END}
242         epRet[0] = SEG_G;
243         epRet[5] = SEG_END;
244         break;
245     case 14: //E
246         //{SEG_A,SEG_G,SEG_F,SEG_D,SEG_E,SEG_END}
247         epRet[1] = SEG_G;
248         epRet[2] = SEG_F;

```

```

    epRet[5] = SEG_END;
249     break;
    case 15: //F
251         //{SEG_A,SEG_E,SEG_F,SEG_G,SEG_END}
        epRet[1] = SEG_E;
253         epRet[2] = SEG_F;
        epRet[3] = SEG_G;
255         epRet[4] = SEG_END;
        break;
257     }
    return epRet;
259 }

```

7.9 ../Sources/GPIO/gpio_hal.h

```

1  /*
    *****
    */
    /* File name:      gpio_hal.h
        */
3  /* File description: This file has a couple of useful
    macros to        */
    /*                control and init the GPIO pins from
        the KLM25Z    */
5  /* Author name:     ddello
        */
    /* Creation date:   01abr2016
        */
7  /* Revision date:   13abr2016
        */
    /*
    *****
    */
9
#define SOURCES_GPIO_GPIO_HAL_H
11 #define SOURCES_GPIO_GPIO_HAL_H

```

```

13 #include "KL25Z/es670_peripheral_board.h"

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

19 #define GPIO_MUX_ALT        0x01u

21 #define GPIO_HIGH      1
#define GPIO_LOW         0
23
25 /**
 * Ungates the clock for a gpio port
 * @param PORT_ID the GPIO port id(A,B)
27 */
#define GPIO_UNGATE_PORT(PORT_ID)\
29 _GPIO_UNGATE_PORT(PORT_ID)

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

35
37 /**
 * inits a pin as GPIO in the given direction
 * @param PORT_ID the GPIO port id(A,B)
39 * @param PIN_NUM pin number in port
 * @param DIR pin direction (GPIO_HIGH, GPIO_LOW)
41 */
#define GPIO_INIT_PIN(PORT_ID, PIN_NUM, DIR)\
43 _GPIO_INIT_PIN(PORT_ID, PIN_NUM, DIR)

```

```

45 //Wrapper macro above is needed for argument expansion
    when using concatenation
#define _GPIO_INIT_PIN(PORT_ID, PIN_NUM, DIR)\
47     /* set pin as gpio */\
    PORT ## PORT_ID ## _PCR ## PIN_NUM = PORT_PCR_MUX(
    GPIO_MUX_ALT);\
49     /* Set pin direction */\
    if(DIR == GPIO_OUTPUT){\
51         GPIO ## PORT_ID ## _PDDR |= GPIO_PDDR_PDD(0x01 <<
    PIN_NUM);\
        }else{\
53         GPIO ## PORT_ID ## _PDDR &= ~GPIO_PDDR_PDD(0x01 <<
    PIN_NUM);\
        }
55
57 /**
    * Writes a pin with the given value
59 * @param PORT_ID the GPIO port id(A,B)
    * @param PIN_NUM pin number in port
61 * @param VAL pin value (GPIO_HIGH, GPIO_LOW)
    */
63 #define GPIO_WRITE_PIN(PORT_ID, PIN_NUM, VAL)\
    _GPIO_WRITE_PIN(PORT_ID, PIN_NUM, VAL)
65
    //Wrapper macro above is needed for argument expansion
    when using concatenation
67 #define _GPIO_WRITE_PIN(PORT_ID, PIN_NUM, VAL)\
    if(VAL == GPIO_HIGH){\
69     GPIO ## PORT_ID ## _PSOR = GPIO_PSOR_PTSO( (0x01U <<
    PIN_NUM) );\
        }else{\
71     GPIO ## PORT_ID ## _PCOR = GPIO_PCOR_PTCO( (0x01U <<
    PIN_NUM) );\
        }
73

```

```

75  /**
   * Writes the given value to the pins given in the MASK
   * @param PORT_ID the GPIO port id(A,B)
77  * @param MASK 31 bit Mask with 1 in the bits
   *     corresponding to the pins of interest.
   * @param VAL pins value (GPIO_HIGH, GPIO_LOW)
79  */
#define GPIO_WRITE_MASK(PORT_ID, MASK, VAL)\
81     _GPIO_WRITE_MASK(PORT_ID, MASK, VAL)

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

89

91 /**
   * Reads the status of a GPIO PIN
93  * @param PORT_ID the GPIO port id(A,B)
   * @param PIN_NUM pin number in port
95  * @param VAL pin value (GPIO_HIGH, GPIO_LOW)
   */
97 #define GPIO_READ_PIN(PORT_ID, PIN_NUM)\
   _GPIO_READ_PIN(PORT_ID, PIN_NUM)

99

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

103

#endif /* SOURCES_GPIO_GPIO_HAL_H */

```

7.10 ../Sources/Main/es670.c

```
1 #include "fsl_device_registers.h"
2 #include "KL25Z/es670_peripheral_board.h"
3 #include "LedSwi/ledswi_hal.h"
4 #include "Mcg/mcg_hal.h"
5 #include "Buzzer/buzzer_hal.h"
6 #include "SevenSeg/sevenseg_hal.h"
7 #include "PIT/pit_hal.h"
8 #include "Util/util.h"
9
10
11 int main(void)
12 {
13     mcg_clockInit();
14     buzzer_init();
15     ledswi_initLedSwitch(3,1);
16     sevenseg_init();
17     sevenseg_printHex(0xABCDu);
18     unsigned short usPrintHex = 1;
19     unsigned short usLedOn = 1;
20     while(1){
21         if(ledswi_getSwitchStatus(3) == SWITCH_ON){
22             usPrintHex = !usPrintHex;
23             usLedOn = !usLedOn;
24             if(usPrintHex){
25                 sevenseg_printHex(0xABCDu);
26             }else{
27                 sevenseg_printDec(0xABCDu);
28             }
29             if(usLedOn){
30                 ledswi_setLed(4);
31             }else{
32                 ledswi_clearLed(4);
33             }
34         }
35     }
```



```

36     /* Never leave main */
    return 0;
38 }

```

7.11 ../Sources/Buzzer/buzzer_hal.c

```

/*
*****
*/
2 /* File name:      buzzer_hal.c
    */
/* File description: File dedicated to the hardware
    abstraction layer*/
4 /*
    related buzzer from the peripheral
    board
    */
/* Author name:      dloubach
    */
6 /* Creation date:   12jan2016
    */
/* Revision date:    13abr2016
    */
8 /*
*****
*/

10 #include "GPIO/gpio_hal.h"
#include "buzzer_hal.h"
12 #include "KL25Z/es670_peripheral_board.h"
#include "PIT/pit_hal.h"
14
16 /**
 * Initialize the buzzer device
 */
18 void buzzer_init(void)
{
20     GPIO_UNGATE_PORT(BUZZER_PORT_ID);

```

```

    GPIO_INIT_PIN(BUZZER_PORT_ID, BUZZER_PIN, GPIO_OUTPUT);
22  pit_enable();
    }
24
26
    /**
28  * Clear the buzzer
    */
30 void buzzer_clearBuzz(void)
    {
32     GPIO_WRITE_PIN(BUZZER_PORT_ID, BUZZER_PIN, GPIO_LOW);
    }
34
36
    /**
38  * Set the buzzer
    */
40 void buzzer_setBuzz(void)
    {
42     GPIO_WRITE_PIN(BUZZER_PORT_ID, BUZZER_PIN, GPIO_HIGH);
    }
44
    /**
46  * Handler for buzzer interruptions
    */
48 void _buzzer_interrupt_handler(void){
    buzzer_setBuzz();
50 buzzer_clearBuzz();
    pit_mark_interrupt_handled(BUZZER_PIT_TIMER_NUMB);
52 }

    /**
54  * Starts the buzzer with the specified period
56  *

```

```

    * @param uiPeriod The period of the buzzer signal, in
      clock cycles (40MHz)
58 */
void buzzer_initPeriodic(unsigned int uiPeriod){
60 //Init timer 1
    pit_start_timer_interrupt(BUZZER_PIT.TIMER_NUMB,
        uiPeriod, &_buzzer_interrupt_handler);
62 }

64 /**
    * Stops any periodic buzzer signal
66 */
void buzzer_stopPeriodic(void){
68     pit_stop_timer_interrupt(BUZZER_PIT.TIMER_NUMB);
}

```

7.12 ../Sources/Buzzer/buzzer_hal.h

```

1  /*
    *****
    */
    /* File name:      buzzer_hal.h
        */
3  /* File description: Header file containing the functions
    /methods */
    /*                interfaces for handling BUZZER from
    the                */
5  /*                peripheral board
        */
    /* Author name:    dloubach
        */
7  /* Creation date:    12jan2016
        */
    /* Revision date:   13abr2016
        */

```

```

9  /*
    *****
    */

11 #ifndef SOURCES_BUZZER_HAL_H_
    #define SOURCES_BUZZER_HAL_H_

13
15 /**
    * Initialize the buzzer device
    */
17 void buzzer_init(void);

19
21 /**
    * Clear the buzzer
    */
23 void buzzer_clearBuzz(void);

25
27 /**
    * Set the buzzer
    */
29 void buzzer_setBuzz(void);

31 /**
    * Starts the buzzer with the specified period
    *
33     * @param uiPeriod The period of the buzzer signal, in
        clock cycles (40MHz)
    */
35 void buzzer_initPeriodic(unsigned int uiPeriod);

37
39 /**
    * Stops any periodic buzzer signal
    */
41 void buzzer_stopPeriodic(void);

```

43

```
#endif /* SOURCES_BUZZER_HAL_H */
```

7.13 ../Sources/KL25Z/es670_peripheral_board.h

```

/*
    *****
    */
2 /* File name:          es670_peripheral_board.h
    */
/* File description: Header file containing the
    peripherals mapping */
4 /*                of the peripheral board for the
    ES670 hardware*/
/* Author name:      dloubach
    */
6 /* Creation date:    16out2015
    */
/* Revision date:     25fev2016
    */
8 /*
    *****
    */

10 #ifndef SOURCES_ES670_PERIPHERAL_BOARD_H_
#define SOURCES_ES670_PERIPHERAL_BOARD_H_

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

16 /*                General uC definitions
    */

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

```

```

20 #define CGC_CLOCK_ENABLED          0x01U

22 /* GPIO DIRECTION */
#define GPIO_OUTPUT          0x01U

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

32 /*                      END OF General uC definitions
   */

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

40 #define BUZZER_PIT_TIMER_NUMB      1

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

46

```

```

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

52
    /* THIS PIN CONFLICTS WITH PTA1 USED AS UART0_RX IN THE
    OPENSDA SERIAL DEBUG PORT */
54 #define LS1_PIN                  1U
    /* led/switch #1 pin */
#define LS1_DIR_OUTPUT             (GPIO_OUTPUT <<
    LS1_PIN)
56 #define LS1_DIR_INPUT            (GPIO_OUTPUT <<
    LS1_PIN)
#define LS1_ALT                    0x01u
    /* GPIO alternative */

58
    /* THIS PIN CONFLICTS WITH PTA2 USED AS UART0_TX IN THE
    OPENSDA SERIAL DEBUG PORT */
60 #define LS2_PIN                  2U
    /* led/switch #2 pin */
#define LS2_DIR_OUTPUT             (GPIO_OUTPUT <<
    LS2_PIN)
62 #define LS2_DIR_INPUT            (GPIO_OUTPUT <<
    LS2_PIN)
#define LS2_ALT                    LS1_ALT

64
#define LS3_PIN                    4U
    /* led/switch #3 pin */
66 #define LS3_DIR_OUTPUT             (GPIO_OUTPUT <<
    LS3_PIN)
#define LS3_DIR_INPUT             (GPIO_OUTPUT <<
    LS3_PIN)

```

```

68 #define LS3_ALT                                LS1_ALT

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

74 /*
    */
    /*
    */
76 /*
    */
    /*
    */
78 #define SEV_SEG_PORT_BASE_PNT                    PORTC
    /* peripheral port base pointer */
#define SEV_SEG_PORT_ID                            C
    /* peripheral port identifier*/
80 #define SEV_SEG_GPIO_BASE_PNT                    PTC
    /* peripheral gpio base pointer */

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

84
#define SEGA_PIN                                    0
    /* Segment A*/
86 #define SEGA_DIR_OUTPUT                            (GPIO_OUTPUT <<
    SEGA_PIN)
#define SEGA_ALT                                    0x01u
    /* GPIO alternative */

88
#define SEGB_PIN                                    1

```


90	<code>#define SEGB_DIR.OUTPUT</code>	<code>(GPIO.OUTPUT <<</code>
	<code>SEGB_PIN)</code>	
	<code>#define SEGB_ALT</code>	<code>SEGA_ALT</code>
92		
	<code>#define SEGC_PIN</code>	<code>2</code>
94	<code>#define SEGC_DIR.OUTPUT</code>	<code>(GPIO.OUTPUT <<</code>
	<code>SEGC_PIN)</code>	
	<code>#define SEGC_ALT</code>	<code>SEGA_ALT</code>
96		
	<code>#define SEGD_PIN</code>	<code>3</code>
98	<code>#define SEGD_DIR.OUTPUT</code>	<code>(GPIO.OUTPUT <<</code>
	<code>SEGD_PIN)</code>	
	<code>#define SEGD_ALT</code>	<code>SEGA_ALT</code>
100		
	<code>#define SEGE_PIN</code>	<code>4</code>
102	<code>#define SEGE_DIR.OUTPUT</code>	<code>(GPIO.OUTPUT <<</code>
	<code>SEGE_PIN)</code>	
	<code>#define SEGE_ALT</code>	<code>SEGA_ALT</code>
104		
	<code>#define SEGF_PIN</code>	<code>5</code>
106	<code>#define SEGF_DIR.OUTPUT</code>	<code>(GPIO.OUTPUT <<</code>
	<code>SEGF_PIN)</code>	
	<code>#define SEGF_ALT</code>	<code>SEGA_ALT</code>
108		
	<code>#define SEGG_PIN</code>	<code>6</code>
110	<code>#define SEGG_DIR.OUTPUT</code>	<code>(GPIO.OUTPUT <<</code>
	<code>SEGG_PIN)</code>	
	<code>#define SEGG_ALT</code>	<code>SEGA_ALT</code>
112		
	<code>#define SEGDP_PIN</code>	<code>7</code>
114	<code>#define SEGDP_DIR.OUTPUT</code>	<code>(GPIO.OUTPUT <<</code>
	<code>SEGDP_PIN)</code>	
	<code>#define SEGDP_ALT</code>	<code>SEGA_ALT</code>
116		
	<code>#define SEG_DISP1_PIN</code>	<code>13</code>

```

118 #define SEG_DISP1_DIR_OUTPUT      (GPIO_OUTPUT <<
      SEG_DISP1_PIN)
    #define SEG_DISP1_ALT          SEGA_ALT
120
    #define SEG_DISP2_PIN          12
122 #define SEG_DISP2_DIR_OUTPUT      (GPIO_OUTPUT <<
      SEG_DISP2_PIN)
    #define SEG_DISP2_ALT          SEGA_ALT
124
    #define SEG_DISP3_PIN          11
126 #define SEG_DISP3_DIR_OUTPUT      (GPIO_OUTPUT <<
      SEG_DISP3_PIN)
    #define SEG_DISP3_ALT          SEGA_ALT
128
    #define SEG_DISP4_PIN          10
130 #define SEG_DISP4_DIR_OUTPUT      (GPIO_OUTPUT <<
      SEG_DISP4_PIN)
    #define SEG_DISP4_ALT          SEGA_ALT
132
    /*                      END of SEVEN SEGMENT DISPLAY
      Definitions                      */
134
136 #endif /* SOURCES_ES670_PERIPHERAL_BOARD.H */

```

7.14 ../Sources/Util/util.h

```

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

```

```

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

/*
   *****
   */
12
#define UTIL_H
14
/**
   * generates ~ 088 micro sec
18 */
void util_genDelay088us(void);
20

/**
   * generates ~ 250 micro sec
24 */
void util_genDelay250us(void);
26

/**
   * generates ~ 1 mili sec
30 */
void util_genDelay1ms(void);
32

```

```

34  /**
   * generates ~ 10 mili sec
36  */
void util_genDelay10ms(void);
38
40 #endif /* UTIL_H */

```

7.15 ../Sources/Util/util.c

```

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

```

```

12  /*
13      ****
14      */
15
16  #include "util.h"
17
18  /**
19   * generates ~ 088 micro sec
20   */
21  void util_genDelay088us( void )
22  {
23      char i;
24      for( i=0; i<120; i++)
25      {
26          __asm( "NOP" );
27          __asm( "NOP" );
28          __asm( "NOP" );
29          __asm( "NOP" );
30          __asm( "NOP" );
31          __asm( "NOP" );
32          __asm( "NOP" );
33          __asm( "NOP" );
34          __asm( "NOP" );
35          __asm( "NOP" );
36          __asm( "NOP" );
37          __asm( "NOP" );
38          __asm( "NOP" );
39          __asm( "NOP" );
40      }
41  }
42
43
44  /**
45   * generates ~ 250 micro sec

```

```

48  */
void util_genDelay250us(void)
{
50    char i;
    for(i=0; i<120; i++)
52    {
        __asm("NOP");
54        __asm("NOP");
        __asm("NOP");
56        __asm("NOP");
        __asm("NOP");
58        __asm("NOP");
        __asm("NOP");
60        __asm("NOP");
        __asm("NOP");
62        __asm("NOP");
    }
64    util_genDelay088us();
    util_genDelay088us();
66 }

68

70 /**
/* generates ~ 1 mili sec
72 */
void util_genDelay1ms(void)
74 {
    util_genDelay250us();
76    util_genDelay250us();
    util_genDelay250us();
78    util_genDelay250us();
}
80

82 /**

```

```

    * generates ~ 10 mili sec
84 */
void util_genDelay10ms(void)
86 {
    util_genDelay1ms();
88     util_genDelay1ms();
    util_genDelay1ms();
90     util_genDelay1ms();
    util_genDelay1ms();
92     util_genDelay1ms();
    util_genDelay1ms();
94     util_genDelay1ms();
    util_genDelay1ms();
96     util_genDelay1ms();
}
```