```
/**********************************
 * Filename      : UE09_LED.h      *
 * Created on    : Dec 11, 2018    *
 * Author        : Christian Zahner*
 **********************************/

/* ###################################################################
**     Filename    : main.c
**     Project     : test9a
**     Processor   : MCF52259CAG80
**     Version     : Driver 01.00
**     Compiler    : CodeWarrior MCF C Compiler
**     Date/Time   : 2014-10-11, 15:20, # CodeGen: 0
**     Abstract    :
**         Main module.
**         This module contains user's application code.
**     Settings    :
**     Contents    :
**         No public methods
**
** ###################################################################*/
/*!
** @file main.c
** @version 01.00
** @brief
**         Main module.
**         This module contains user's application code.
*/
/* MODULE main */

#pragma compact_abi


#include "UART0.h"
#include "support_common.h"  // include peripheral declarations and more;
#include "uart_support.h"    // universal asynchronous receiver transmitter,
                             // (d.h. die serielle Schnittstelle)
#include "terminal_wrapper.h"

#include "UE09_LED.h"


// - Bitte darauf achten, dass am Coldfire-Serial Port ein
//   Terminal (Putty o.ä.) mit 19200kBaud angeschlossen ist.
// - Als Target muss   <projektname>_RAM_OSBDM   ausgewählt werden.



  #define IPS 0x40000000

  /* Definitions for MCF_GPIO Port NQ */
  #define PORTNQ                  0x40100008 // Output
  #define PNQPAR                  0x40100068 // GPIO einstellen
  #define DDRNQ                   0x40100020 // Datenrichtung/Output
  #define PINDATANQ               0x40100038 // Auslesen PIN
  #define CLRNQ                   0x40100068 // Schreiben an Port Taster

  #define PORTNQ1                 0x00000002
  #define PORTNQ5                 0x00000020


  /* Bit definitions for MCF_GPIO Port TC */
  #define PORTTC                  0x4010000F // Output
  #define PTCPAR                  0x4010006F // GPIO einstellen
```

```
#define DDRTC                   0x40100027 // Datenrichtung/Input
#define SETTC                   0x4010003F // Set Pin bzw. "LEDS"
#define CLRTC                   0x40100057 // Schreiben an "LEDS"

#define PORTTC0                 0x00000001 // LED 01
#define PORTTC1                 0x00000002 // LED 02
#define PORTTC2                 0x00000004 // LED 03
#define PORTTC3                 0x00000008 // LED 04


 /* Defines for interrupt */

/* Definitionen für MCF_Edge Post */
#define MNP_INTC0_IMRL          0x40000C0C  // Interrupt Mask Register
#define MNP_EPORT_EPPAR         0x40130000  // Edge control register
#define MNP_EPORT_EPIER         0x40130003  // Edge Port Interrupt Enable Reg
#define MNP_EPORT_EPFR          0x40130006  // Edge Port Flag Register

void ledOnOff(){

asm
{
  /* MCF52259RM.pdf
  - SW1 and SW2 are connected to PNQPAR5 and PNQPAR1 (Quad function pins!!)
  - LED's 1-4 are connected to DDRTC0-DDRTC3
  */

  /* Enable Switches to be pollable ====================================== */
  /* MCF52259RM.pdf
  - Pin Assignment must be set to GPIO function
    (15.6.5.3 Port NQ Pin Assignment Register (PNQPAR))
  - Port Data Direction must be cleared for input function
    (15.6.2 Port Data Direction Registers (DDRn))
  - Output Data Register must be cleared
    (15.6.1 Port Output Data Registers (PORTn))
   */

      // andi.w #0xF0F0, MNP_GPIO_PNQPAR  geht nicht, deshalb folgender Code
      move.w   PNQPAR,d0             //NQ5 and NQ1 löschen für GPIO Funktion
      and.l    #0xF0F0, d0
      move.w   d0, PNQPAR

      // andi.b #0xDD, MNP_GPIO_DDRNQ  geht nicht, deshalb folgender Code
      move.b   DDRNQ, d0              //NQ5 and NQ1 löschen für input Funktion
      andi.l   #0xDD,d0
      move.b   d0, DDRNQ


      // andi.b #0xDD, MNP_GPIO_CLRNQ  geht nicht, deshalb folgender Code
      move.b   CLRNQ,d0              //NQ5+NQ1 löschen
      andi.l   #0xDD,d0
      move.b   d0, CLRNQ


    /* Enable LEDs as digital outputs ==================================== */
    /* MCF52259RM.pdf
    - Port Data Direction must be set for output function
      15.6.2 Port Data Direction Registers (DDRn))
    - Pin Assignment must be set to GPIO function
      (15.6.5.1 Dual-Function Pin Assignment Registers)
     - Output Data Register must be set/reset
      (15.6.1 Port Output Data Registers (PORTn))
    */
```

```
        clr.b    PTCPAR            //GPIO Funktion (=0)

        move.b   #0xf, d0          //output Funktion (=1)
        move.b   d0, DDRTC

        clr.b    CLRTC             //LEDS OFF, siehe Figure 15-3

    loop:

    ////////////////////////// Taster 1 schaltet LED 1

       clr.l    d0                // Reset D0
       move.b   PINDATANQ, d0     // Taster via PINDATA_SETNQ abfragen
       andi.l   #PORTNQ5, d0      // Check NQ5 auf 0 (SW1 gedrückt)
       bne      LED1_OFF          // falls nicht gedrückt: Sprung

       move.l   #PORTTC0,d1
       move.b   d1, SETTC         // LED einschalten via set bit in GPIO_SETTC
       bra      LED1_END

    LED1_OFF:

       move.l   #~(PORTTC0),d1
       move.b   d1, CLRTC         // LED ausschalten via clear bit in GPIO_CLR

    LED1_END:

    // Nochmal dasselbe für Nr. 2:
    ////////////////////////// Taster 2 schaltet LED 2

       clr.l    d0
       move.b   PINDATANQ, d0     // Taster via PINDATA_SETNQ abfragen
       andi.l   #PORTNQ1,  d0     // Check NQ1 auf 0 (SW2 gedrückt)
       bne      LED2_OFF

       move.l   #PORTTC1,  d1
       move.b   d1, SETTC        // LED einschalten via set bit in GPIO_SET
       bra      LED2_END

    LED2_OFF:

       move.l   #~(PORTTC1),d1
       move.b   d1, CLRTC        // LED ausschalten via clear bit in GPIO_CLR


    LED2_END:

       bra     loop

    }

}
```

```
void ledLatched(){

  asm{

      /* MCF52259RM.pdf
      - SW1 and SW2 are connected to PNQPAR5 and PNQPAR1 (Quad function pins!!)
      - LED's 1-4 are connected to DDRTC0-DDRTC3
      */

      /* Enable Switches to be pollable ===================================== */
      /* MCF52259RM.pdf
      - Pin Assignment must be set to GPIO function
        (15.6.5.3 Port NQ Pin Assignment Register (PNQPAR))
      - Port Data Direction must be cleared for input function
        (15.6.2 Port Data Direction Registers (DDRn))
      - Output Data Register must be cleared
        (15.6.1 Port Output Data Registers (PORTn))
       */

      // andi.w #0xF0F0, MNP_GPIO_PNQPAR  geht nicht, deshalb folgender Code
      move.w   PNQPAR,d0       //NQ5 and NQ1 löschen für GPIO Funktion
      and.l    #0xF0F0, d0
      move.w   d0, PNQPAR

      // andi.b #0xDD, MNP_GPIO_DDRNQ  geht nicht, deshalb folgender Code
      move.b   DDRNQ, d0           //NQ5 and NQ1 löschen für input Funktion
      andi.l   #0xDD,d0
      move.b   d0, DDRNQ

      // andi.b #0xDD, MNP_GPIO_CLRNQ  geht nicht, deshalb folgender Code
      move.b   CLRNQ,d0           //NQ5+NQ1 löschen
      andi.l   #0xDD,d0
      move.b   d0, CLRNQ


       /* Enable LEDs as digital outputs ================================== */
       /* MCF52259RM.pdf
       - Port Data Direction must be set for output function
         (15.6.2 Port Data Direction Registers (DDRn))
       - Pin Assignment must be set to GPIO function
         (15.6.5.1 Dual-Function Pin Assignment Registers)
       - Output Data Register must be set/reset
         (15.6.1 Port Output Data Registers (PORTn))
        */


      clr.b    PTCPAR             //GPIO Funktion (=0)

      move.b   #0xf, d0           //output Funktion (=1)
      move.b   d0, DDRTC

      clr.b    CLRTC              //LEDS OFF, siehe Figure 15-3

   loop:

       // Schleife so lange Taste nicht gedrückt

       wait_for_pressed:

        clr.l    d0
        move.b   PINDATANQ, d0        // Poll the switches via SETNQ
        andi.l   #PORTNQ5, d0         // Check if NQ5 is unset (SW1 pressed)
        bne      wait_for_pressed
```

```
        wait_for_released:

        // Schleife so lange Taste gedrückt

         clr.l    d0
         move.b   PINDATANQ, d0        // Poll the switches via SETNQ
         andi.l   #PORTNQ5, d0         // Check if NQ5 is set (SW1 released)
         beq      wait_for_released

         // LED ein

         move.b   PORTTC, d0           // Inhalt in d0
         bchg     #PORTTC0, d0         // Toggle bit
         move.b   d0, PORTTC           // PORTTC Lampe ein

         bra      loop


      }

}

void ledCount(){

  asm{

      /* MCF52259RM.pdf
      - SW1 and SW2 are connected to PNQPAR5 and PNQPAR1 (Quad function pins!!)
      - LED's 1-4 are connected to DDRTC0-DDRTC3
      */

      /* Enable Switches to be pollable ====================================== */
      /* MCF52259RM.pdf
      - Pin Assignment must be set to GPIO function
        (15.6.5.3 Port NQ Pin Assignment Register (PNQPAR))
      - Port Data Direction must be cleared for input function
        (15.6.2 Port Data Direction Registers (DDRn))
      - Output Data Register must be cleared
        (15.6.1 Port Output Data Registers (PORTn))
      */

          // andi.w #0xF0F0, MNP_GPIO_PNQPAR  geht nicht, deshalb folgender Code
          move.w   PNQPAR,d0          //NQ5 and NQ1 löschen für GPIO Funktion
          and.l    #0xF0F0, d0
          move.w   d0, PNQPAR

          // andi.b #0xDD, MNP_GPIO_DDRNQ  geht nicht, deshalb folgender Code
          move.b   DDRNQ, d0              //NQ5 and NQ1 löschen für input Funktion
          andi.l   #0xDD,d0
          move.b   d0, DDRNQ

          // andi.b #0xDD, MNP_GPIO_CLRNQ  geht nicht, deshalb folgender Code
          move.b   CLRNQ,d0              //NQ5+NQ1 löschen
          andi.l   #0xDD,d0
          move.b   d0, CLRNQ

      /* Enable LEDs as digital outputs ====================================== */
      /* MCF52259RM.pdf
      - Port Data Direction must be set for output function
        (15.6.2 Port Data Direction Registers (DDRn))
      - Pin Assignment must be set to GPIO function
        (15.6.5.1 Dual-Function Pin Assignment Registers)
      - Output Data Register must be set/reset
```

```
        (15.6.1 Port Output Data Registers (PORTn))
      */


          clr.b    PTCPAR                //GPIO Funktion (=0)

          move.b   #0xf, d0              //output Funktion (=1)
          move.b   d0, DDRTC

          clr.b    CLRTC                 //LEDS OFF, siehe Figure 15-3

      loop:


          wait_for_pressed:

           clr.l    d0
           move.b   PINDATANQ, d0        // Poll the switches via SETNQ
           andi.l   #PORTNQ5, d0         // Check if NQ5 is unset (SW1 pressed)
           bne      wait_for_pressed

           clr.l    d0

           wait1:                        // loop to eliminate switch bouncing
             addq.l   #1, d0             // (s.b. = deutsch: "Schalterprellen")
             tst.w    d0
             bne      wait1

          wait_for_released:

           clr.l    d0
           move.b   PINDATANQ, d0        // Poll the switches via SETNQ
           andi.l   #PORTNQ5, d0         // Check if NQ5 is set (SW1 released)
           beq      wait_for_released

           clr.l    d0

           wait2:                         // loop to eliminate switch bouncing
             addq.l   #1, d0              // (s.b. = deutsch: "Schalterprellen")
             tst.w    d0
             bne      wait2


           move.b   PORTTC, d0            // Auslesen des "LED Registers"
           addq.l   #1, d0                // erhöhe aktuellen Inhalt um 1
           move.b   d0, PORTTC            // Übertrage an "LED Register"

           bra      loop


        }

}

void ledCountPlusMinus(){

  asm{

      /* MCF52259RM.pdf
      - SW1 and SW2 are connected to PNQPAR5 and PNQPAR1 (Quad function pins!!)
      - LED's 1-4 are connected to DDRTC0-DDRTC3
      */

      /* Enable Switches to be pollable ==================================== */
```

```
/* MCF52259RM.pdf
- Pin Assignment must be set to GPIO function
  (15.6.5.3 Port NQ Pin Assignment Register (PNQPAR))
- Port Data Direction must be cleared for input function
  (15.6.2 Port Data Direction Registers (DDRn))
- Output Data Register must be cleared
  (15.6.1 Port Output Data Registers (PORTn))
*/

    // andi.w #0xF0F0, MNP_GPIO_PNQPAR  geht nicht, deshalb folgender Code
    move.w   PNQPAR,d0          //NQ5 and NQ1 löschen für GPIO Funktion
    and.l    #0xF0F0, d0
    move.w   d0, PNQPAR

    // andi.b #0xDD, MNP_GPIO_DDRNQ  geht nicht, deshalb folgender Code
    move.b   DDRNQ, d0          //NQ5 and NQ1 löschen für input Funktion
    andi.l   #0xDD,d0
    move.b   d0, DDRNQ

    // andi.b #0xDD, MNP_GPIO_CLRNQ  geht nicht, deshalb folgender Code
    move.b   CLRNQ,d0          //NQ5+NQ1 löschen
    andi.l   #0xDD,d0
    move.b   d0, CLRNQ

/* Enable LEDs as digital outputs ================================== */
/* MCF52259RM.pdf
- Port Data Direction must be set for output function
  (15.6.2 Port Data Direction Registers (DDRn))
- Pin Assignment must be set to GPIO function
  (15.6.5.1 Dual-Function Pin Assignment Registers)
- Output Data Register must be set/reset
  (15.6.1 Port Output Data Registers (PORTn))
*/


    clr.b    PTCPAR               //GPIO Funktion (=0)

    move.b   #0xf, d0             //output Funktion (=1)
    move.b   d0, DDRTC

    clr.b    CLRTC                //LEDS OFF, siehe Figure 15-3

loop:


    wait_for_pressed:

     clr.l    d0
     move.b   PINDATANQ, d0         // Poll the switches via SETNQ
     andi.l   #PORTNQ5, d0          // Check if NQ5 is unset (SW1 pressed)
     beq      wait_for_released1

     clr.l    d0
     move.b   PINDATANQ, d0         // Poll the switches via SETNQ
     andi.l   #PORTNQ1, d0          // Check if NQ5 is unset (SW1 pressed)
     beq      wait_for_released2

     clr.l    d0

    wait1:                          // loop to eliminate switch bouncing
      addq.l   #1, d0               // (s.b. = deutsch: "Schalterprellen")
      tst.w    d0
      bne      wait1
```

```
            bra      wait_for_pressed


           wait_for_released1:

            clr.l    d0
            move.b   PINDATANQ, d0          // Poll the switches via SETNQ
            andi.l   #PORTNQ5, d0           // Check if NQ5 is set (SW1 released)
            beq      wait_for_released1


            clr.l    d0

            wait2:                          // loop to eliminate switch bouncing
             addq.l   #1, d0                // (s.b. = deutsch: "Schalterprellen")
             tst.w    d0
             bne      wait2

            move.b   PORTTC, d0             // Auslesen des "LED Registers"
            addq.l   #1, d0                 // erhöhe aktuellen Inhalt um 1
            move.b   d0, PORTTC             // Übertrage an "LED Register"

            bra      loop

           wait_for_released2:

            clr.l    d0
            move.b   PINDATANQ, d0          // Poll the switches via SETNQ
            andi.l   #PORTNQ1, d0           // Check if NQ5 is set (SW1 released)
            beq      wait_for_released2


            clr.l    d0

            wait3:                          // loop to eliminate switch bouncing
              addq.l   #1, d0               // (s.b. = deutsch: "Schalterprellen")
              tst.w    d0
              bne      wait3

            move.b   PORTTC, d0             // Auslesen des "LED Registers"
            subq.l   #1, d0                 // erhöhe aktuellen Inhalt um 1
            move.b   d0, PORTTC             // Übertrage an "LED Register"

            bra      loop


        }

}

void ledInterrupt(){

    asm
        {
             /* MCF52259RM.pdf
             - SW1 and SW2 verbunden mit PNQPAR5 and
             - PNQPAR1 (Quad function pins!!)
             - LED's 1-4 verbunden mit DDRTC0-DDRTC3
             */

            /* LEDs als digitale Ausgabe konfigurieren  bekannt         */
             /* MCF52259RM.pdf
             - Port Data Direction auf output Funktion setzen
               (15.6.2 Port Data Direction Registers (DDRn))
```

```
   - Pin Assignment auf GPIO Funktion setzen
     (15.6.5.1 Dual-Function Pin Assignment Registers)
   - Output Data Register zurücksetzen
     (15.6.1 Port Output Data Registers (PORTn))*/

    clr.b    PTCPAR               //GPIO Funktion (=0)
    move.b   #0xf, d0             //output Funktion (=1)
    move.b   d0, DDRTC
    clr.b    CLRTC               //LEDS OFF, siehe Figure 15-3
    /* LEDs als digitale Ausgabe konfigurieren ==bekannt (Ende) */



   /* Taster als digitale Eingabe konfigurieren  ============= */
   /* MCF52259RM.pdf
    - Pin Assignment auf IRQ function (primary function)
      (15.6.5.3 Port NQ Pin Assignment Register (PNQPAR))
    - Port Data Direction Löschen für input function
      (15.6.2 Port Data Direction Registers (DDRn))
    - Output Data Register löschen
      (15.6.1 Port Output Data Registers (PORTn))
   */


// Achtung: Andere Funktion für NQ1 und
// NQ5 (primary function (=IRQ), 01 statt 00)
    move.w   PNQPAR,d0            // NQ5 und NQ1 auf IRQ Funktion (01)
    and.l    #0xF0F0, d0
    or.l      #0x0404, d0
    move.w   d0, PNQPAR

    move.b   DDRNQ, d0            //NQ5 and NQ1 löschen für input Funktion
    andi.l   #0xDD,d0
    move.b   d0, DDRNQ

    move.b   CLRNQ,d0            //NQ5+NQ1 löschen
    andi.l   #0xDD,d0
    move.b   d0, CLRNQ

// Einhängen der Unterbrechungsantwortprogramme
// (interrupt service routines)
// (IRQ5=sw1 / IRQ1=sw2)
    lea     int_handler_IRQ1, a1
    move.l  a1, 0x20000100 + 1*4 // IRQ1 Vector

    lea     int_handler_IRQ5, a1
    move.l  a1, 0x20000100 + 5*4 // IRQ5 Vector


// Einstellen rising/falling edge detection,
// Einstellen auf falling edge active (MCF52259RM 17.4.1)
    move.w   MNP_EPORT_EPPAR, d1
    move.w   #0x0808, d1            // bits sensitive to falling edge
    move.w   d1, MNP_EPORT_EPPAR

// Enable EPORT Interrupts (MCF52259RM 17.4.3)           // ENABLE
    move.b   MNP_EPORT_EPIER, d1
    or.l     #0x00000022, d1       // 0x20=IRQ5, 0x2=IRQ1
    move.b   d1, MNP_EPORT_EPIER

// enable IRQ1+5  (MCF52259RM 16.3.2)                    // ÄUSSERE MASKE
    move.l   MNP_INTC0_IMRL, d1
    and.l    #(~0x00000022), d1    //  0x20=IRQ5, 0x2=IRQ1
    move.l   d1, MNP_INTC0_IMRL
```

```
            // Interrupts im Statusregister freigeben (CFPRM 1.5.1)   // INNERE MASKE
               move.w   sr,d1
               andi.l   #~0x00000700,d1
               move.w   d1,sr


          loop:                              // Das ist unser Leerlaufprozess
            bra       loop


////////////////////////////////////////////////////////////////////////////
          int_handler_IRQ5:
              move.l   d0, -(sp)             // WICHTIG!  Alle benutzten
                                             // Register retten
              move.l   d1, -(sp)

            LED1_ON:
            move.l  #PORTTC0,d1
            move.b  d1, SETTC                // LED einschalten

            // Interupt zurücksetzen (MCF52259 17.4.6)
            move.b   #0x20,d0                // Schreiben von 1 löscht die Bits!
            move.b   d0, MNP_EPORT_EPFR

            move.l   (sp)+,d1               // WICHTIG!  Alle benutzten Register
                                            // restaurieren
            move.l   (sp)+,d0
            rte


////////////////////////////////////////////////////////////////////////////
          int_handler_IRQ1:
              move.l   d0, -(sp)
              move.l   d1, -(sp)

            LED1_OFF:
            move.l  #~(PORTTC0),d1
            move.b  d1, CLRTC               // LED ausschalten

            // Interupt zurücksetzen (MCF52259 17.4.6)
            move.b   #0x02,d0               // Schreiben von 1 löscht die Bits!
            move.b   d0, MNP_EPORT_EPFR

            move.l   (sp)+,d1
            move.l   (sp)+,d0
            rte

////////////////////////////////////////////////////////////////////////////


          }
  }
```