```
/**********************************
 * Headerfilename: UE11_echo.c     *
 * Created on    : Dec 11, 2018    *
 * Author        : Christian Zahner*
 **********************************/


  #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 der Uebungsheader

#include "UE11_Echo.h"


 // In der Praxis werden die folgenden Defines vom Chip-Hersteller mit der
 // Entwicklungsumgebung zur Verfügung gestellt. Siehe Datei MCF52259_GPIO.h
 // Nur aus didaktischen Gründen sollten Sie die Definitionen selbst machen.
 // Hier in der Musterlösung entsprechend:


/* Definitionen für MCF_GPIO Port NQ */
#define MNP_GPIO_PORTNQ        0x40100008
#define MNP_GPIO_DDRNQ         0x40100020
#define MNP_GPIO_SETNQ         0x40100038
#define MNP_GPIO_CLRNQ         0x40100050
#define MNP_GPIO_PNQPAR        0x40100068
#define MNP_GPIO_PORTNQ_NQ1    0x02
#define MNP_GPIO_PORTNQ_NQ5    0x20


/* Definitionen für  MCF_GPIO Port TC */
#define MNP_GPIO_PORTTC        0x4010000F
#define MNP_GPIO_DDRTC         0x40100027
#define MNP_GPIO_SETTC         0x4010003F
#define MNP_GPIO_CLRTC         0x40100057
#define MNP_GPIO_PTCPAR        0x4010006F
#define MNP_GPIO_PORTTC_TC0    0x1
#define MNP_GPIO_PORTTC_TC1    0x2
#define MNP_GPIO_PORTTC_TC2    0x4
#define MNP_GPIO_PORTTC_TC3    0x8


/* 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 echo(){

    asm{

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

```
*/

/* Enable Switches  ==================================== */
/* MCF52259RM.pdf
- Pin Assignment auf GPIO 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)) */


  move.w  MNP_GPIO_PNQPAR,d0   // NQ5 und NQ1 auf IRQ Funktion (01)
  and.l   #0xFFF0, d0
  or.l    #0x0004, d0
  move.w  d0, MNP_GPIO_PNQPAR

  move.b  MNP_GPIO_DDRNQ, d0   //NQ5 and NQ1 löschen für input Funktion
  and.l   #0xFFFD,d0
  move.b  d0, MNP_GPIO_DDRNQ

  move.b  MNP_GPIO_PORTNQ,d0   //NQ5+NQ1 löschen
  and.l   #0xFFFD,d0
  move.b  d0, MNP_GPIO_PORTNQ

  /* LEDs als digitale Ausgabe konfigurieren ========================== */
  /* 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    MNP_GPIO_PTCPAR       //GPIO Funktion (=0)

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

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


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

  /* Einstellen rising/falling edge detection,
   Interrupts enable,
   "interrupt priotity mask" im Statusregister setzen */

  // Einstellen auf Level bzw. Falling Edge sensitivity (MCF52259RM 17.4.1)
   move.w  MNP_EPORT_EPPAR, d1
   and.l   #(~0x0000000c), d1     // level sensitivity
   or.l    #(0x00000008), d1      // falling edge active
   move.w  d1, MNP_EPORT_EPPAR

  // enable EPORT interrupts (MCF52259RM 17.4.3)
   move.b  MNP_EPORT_EPIER, d1
   or.l    #0x00000002, d1       // 0x20=IRQ5, 0x2=IRQ1
   move.b  d1, MNP_EPORT_EPIER

  // enable IRQ1+5 (MCF52259RM 16.3.2)
   move.l  MNP_INTC0_IMRL, d1
```

```
     and.l   #(~0x00000002), d1    // 0x20=IRQ5, 0x2=IRQ1
     move.l  d1, MNP_INTC0_IMRL

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


     move.l  0xaffeaffe, d0
     move.l  0x11111111,d1


 loop:                    // Das ist unser Leerlaufprozess
   //divs.l d0,d1
     bra     loop



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

     move.b   MNP_GPIO_PORTTC,d1
     or.l     #MNP_GPIO_PORTTC_TC0,d1
     move.b   d1, MNP_GPIO_PORTTC   // LED (=Ausgangssignal) einschalten

// Interupt zurücksetzen (MCF52259 17.4.6)
     move.b  #02, d0                // 0x02 für IRQ1
     move.b  d0, MNP_EPORT_EPFR     // Schreiben von 1 löscht die Bits!

    #if 0
    move.l   0xaffeaffe, d0 // max. ca. 62 kHZ mit dieser Division
             move.l 0x11111111,d1  // 720 kHz ohne
             divs.l d0,d1
    #endif

    #if 0
             nop
             nop
             nop
             nop
    #endif

    move.b   MNP_GPIO_PORTTC,d1
    and.l    #~(MNP_GPIO_PORTTC_TC0),d1
    move.b   d1, MNP_GPIO_PORTTC     // LED (=Ausgangssignal) ausschalten


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

    }

}
```