```c
/*****************************************************************************/
/* BLINKY.C: LED Flasher                                                     */
/*****************************************************************************/
/* This file is part of the uVision/ARM development tools.                   */
/* Copyright (c) 2005-2006 Keil Software. All rights reserved.               */
/* This software may only be used under the terms of a valid, current,       */
/* end user licence from KEIL for a compatible version of KEIL software      */
/* development tools. Nothing else gives you the right to use this software. */
/*****************************************************************************/

#include <stdio.h>
#include <LPC23xx.H>                    /* LPC23xx definitions               */
#include "LCD.h"                        /* Graphic LCD function prototypes   */


/* Function that initializes LEDs                                            */
void LED_Init(void) {
  PINSEL10 = 0;                         /* Disable ETM interface, enable LEDs */
  FIO2DIR  = 0x000000FF;                /* P2.0..7 defined as Outputs        */
  FIO2MASK = 0x00000000;
}

/* Function that turns on requested LED                                      */
void LED_On (unsigned int num) {
  FIO2SET = (1 << num);
}

/* Function that turns off requested LED                                     */
void LED_Off (unsigned int num) {
  FIO2CLR = (1 << num);
}

/* Function that outputs value to LEDs                                       */
void LED_Out(unsigned int value) {
  FIO2CLR = 0xFF;                       /* Turn off all LEDs                 */
  FIO2SET = (value & 0xFF);             /* Turn on requested LEDs            */
}


/* Function for displaying bargraph on the LCD display                       */
void Disp_Bargraph(int pos_x, int pos_y, int value) {
  int i;

  set_cursor (pos_x, pos_y);
  for (i = 0; i < 16; i++)  {
    if (value > 5)  {
      lcd_putchar (0x05);
      value -= 5;
    }  else  {
      lcd_putchar (value);
      value = 0;
    }
  }
}



/* Import external IRQ handlers from IRQ.c file                              */
extern __irq void T0_IRQHandler  (void);
extern __irq void ADC_IRQHandler (void);

/* Import external functions from Serial.c file                              */
extern       void init_serial    (void);

/* Import external variables from IRQ.c file                                 */
extern short AD_last;
extern unsigned char clock_700us_Puls;
extern unsigned char clock_700us_Pause;
```

```
extern unsigned char clock_1s_Puls;
extern unsigned char clock_1s_Pause;

/*============================================================================
=*/

int main (void) {
  int i, t, y, z, x, s;

  int werte_array[84];
  short AD_old, AD_value, AD_print, AD_wert;

  LED_Init();                             /* LED Initialization               */

  /* Enable and setup timer interrupt, start timer                             */
  T0MR0         = 1198;                  /* 1msec = 12000-1 at 12.0 MHz */
  T0MCR         = 3;                      /* Interrupt and Reset on MR0  */
  T0TCR         = 1;                      /* Timer0 Enable               */
  VICVectAddr4  = (unsigned long)T0_IRQHandler;/* Set Interrupt Vector         */
  VICVectCntl4  = 15;                     /* use it for Timer0 Interrupt */
  VICIntEnable  = (1  << 4);              /* Enable Timer0 Interrupt     */

  /* Power enable, Setup pin, enable and setup AD converter interrupt          */
  PCONP        |= (1 << 12);              /* Enable power to AD block    */
  PINSEL1       = 0x4000;                 /* AD0.0 pin function select   */
  AD0INTEN      = (1 <<  0);              /* CH0 enable interrupt        */
  AD0CR         = 0x00200301;             /* Power up, PCLK/4, sel AD0.0 */
  VICVectAddr18 = (unsigned long)ADC_IRQHandler;/* Set Interrupt Vector        */
  VICVectCntl18 = 14;                     /* use it for ADC Interrupt    */
  VICIntEnable  = (1  << 18);             /* Enable ADC Interrupt        */

  init_serial();                          /* Init UART                   */

  lcd_init();
  lcd_clear();
  lcd_print ("  IR Tester  ");
  set_cursor (0, 1);     //(spalte, zeile)
  lcd_print ("    ");

  for (i = 0; i < 200000; i++);           /* Wait for initial display          */

  for(i = 0; i<84; i++){                  /* LCD mit Nullen auffüllen          */
       werte_array[i]=0;
       }

   LED_Off(6);
   t=0; y=0; z=6;

  while (1) { /* Loop forever                        */
    AD_value = AD_last;                   /* Read AD_last value                */
    if (AD_value != AD_last)              /* Make sure that AD interrupt did   */
      AD_value = AD_last;                 /* not interfere with value reading  */
    AD_print  = AD_value;                 /* Get unscaled value for printout   */
    AD_value /= 13;                       /* Scale to AD_Value to 0 - 78       */
    if (AD_old != AD_value)  {            /* If AD value has changed           */
        AD_old = AD_value;
      //    Disp_Bargraph(0, 1, AD_value);    /* Display bargraph according to AD  */
       }


//Timer mit 700µs
    if (clock_700us_Puls){
          clock_700us_Puls = 0;
          t++; y++;
          AD_wert = AD_value;   //Holt alle 0,7ms den Abtastwert und speichert ihn nach
 AD_wert
          LED_On(5);
```

```
//==========NULL und EINS Auswertung=============================================
==================

          if(AD_wert < 10){     //ist der Wert kleiner 10, so ist das ein Puls
            x = 1;
          }
          if(AD_wert > 60){     //Ist der Wert größer als 60, so ist das eine Pause
            x = 0;
          }

//==========Synchronisation======================================================
==================

     if(t==12 && x*x == 1){    //Wenn bei der zwölften Abtastung x immernoch 1 ist, dan
n ist das das Startbit
         s=1;
     }

     if(s==1){                 //s ist quasi die Startvariable, hier beginnt die Abtast
ung

        for(i=0;i<66;i++){     //Beschreibt das Array mit den log. Werten der Abtastung
            werte_array[i]=x;
            z++;
            }
     }

     if(werte_array[25] == 1 && werte_array[26] == 0){  //25 ist im Diagramm die 1. Abt
astung des Kommandoblocks
        LED_On(2);
        }


    if(clock_700us_Pause){
     clock_700us_Pause = 0;
       LED_Off(5);


        }

        if(z >= 72){z = 0; LED_Off(0); s=0;}  //Rücksetzen des Zählers nach 72 Abtastun
gen
        if(y >= 84){y = 0;}                   //Rücksetzen des Zählers nach 84 Abtastun
gen
        if(t >= 84){t = 0;LED_Off(2);}        //Rücksetzen des Zählers nach 84 Abtastun
gen
    }


//Timer für die Serielle Datenausgabe
        if(clock_1s_Puls){
         clock_1s_Puls = 0;
        //printf ("AD value = 0x%03x\n\r", AD_print);  //Ausgabe der Daten an COM
            LED_On(7);
        }
        if(clock_1s_Pause){
         clock_1s_Pause = 0;
            LED_Off(7);
        }
    }
}
```