```c
/*****************************************************************/
/* Hochschule fuer Technik und Wirtschaft                        */
/* Fakultät fuer Ingenieurwissenschaften                         */
/* Labor fuer Eingebettete Systeme                               */
/* Mikroprozessortechnik                                         */
/*****************************************************************/
/*                                                               */
/*  C_Übung.C:                                                   */
/*    Programmrumpf fuer C-Programme mit dem Keil                */
/*    Entwicklungsprogramm uVision fuer ARM-Mikrocontroller      */
/*                                                               */
/*****************************************************************/
/*  Name / Matrikel-Nr.: *      Valentin Straßer    5014379      */
/*                       *      Michal Roziel       5012845      */
/*****************************************************************/
/*  Abgabedatum:         *      30.01.2025                       */
/*****************************************************************/

#include <LPC21xx.H>  // LPC21xx Mikrocontroller Definitionen
#include "C_Uebung.H"

// UART initialisieren
void uartInit(unsigned int baudRate, unsigned int dataBits, unsigned int stopBits, unsigned int
paritySelect, unsigned int parityEnable) {
    unsigned int uartConfig = 0;
    unsigned int Frequenzteiler;

    // UART-Konfiguration erstellen
    uartConfig = (paritySelect << 1) + parityEnable;
    uartConfig = (uartConfig << 1) + stopBits;
    uartConfig = (uartConfig << 2) + (dataBits-5);

    // UART0 an P0.0 (TxD0) und P0.1 (RxD0) aktivieren
    PINSEL0 |= 0x05;

    // Baudratenteiler berechnen
    Frequenzteiler = PCLOCK / (16 * baudRate);

    // UART-Register konfigurieren
    U0LCR = DLAB_BIT | uartConfig;          // DLAB-Bit setzen, UART-Konfiguration
    U0DLL = Frequenzteiler % 256;           // Niedriges Byte des Baudratenteilers
    U0DLM = Frequenzteiler / 256;           // Hohes Byte des Baudratenteilers
    U0LCR = uartConfig;                     // DLAB-Bit löschen
    U0FCR = UART_FIFO_ENABLE;               // FIFO aktivieren
}

// Schalterzustand von P0.16 -> S1 lesen
unsigned int readSwitchState1(void) {
    return (IOPIN0 >> 16) & 1;
}

// Schalterzustand von P0.17 -> S2 lesen
unsigned int readSwitchState2(void) {
    return (IOPIN0 >> 17) & 1;
}

// Schalterzustand von P1.25 -> S3 lesen
unsigned int readSwitchState3(void) {
    return (IOPIN1 >> 25) & 1;
}

// BCD-Eingang von P0.10-P0.13 lesen
unsigned int readInputBCD(void) {
    return (IOPIN0 >> 10) & 0xF;
}

// Menü über UART senden
void sendMenu(void) {
    uartSendString("\r\nStopp-Uhr\r\n");
    uartSendString("\tStart und Anhalten durch Druecken der Interrupt-Taste\r\n");
    uartSendString("\ts,S - Start/Stop\r\n");
    uartSendString("\ta,A - Anzeigen\r\n");
```

```
72      uartSendString("\tr,R - Reset\r\n");
73  }
74
75  // Baudrate initialisieren basierend auf BCD-Eingang
76  unsigned int initBaudrate(void) {
77      unsigned int index = readInputBCD();
78      // Wenn Index größer als 9 ist, baudrates[9] verw.
79      return (index > 9) ? baudrates[9] : baudrates[index];
80  }
81
82  // Einzelnes Zeichen über UART senden
83  void uartSendChar(char data) {
84      while ((U0LSR & UART_READY_BIT) == 0);  // Warten, bis UART bereit ist
85      U0THR = data;  // Zeichen senden
86  }
87
88  // String über UART senden
89  void uartSendString(char* str) {
90      int i = 0;
91      while (str[i] != '\0') {  // Bis zur Nullterminierung
92          uartSendChar(str[i]);  // Zeichenweise senden
93          i++;
94      }
95  }
96
97  // Zeichen über UART empfangen
98  char uartReadChar(void) {
99      while (!(U0LSR & UART_RX_READY));  // Warten, bis ein Zeichen empfangen wurde
100     return U0RBR;               // Empfangenes Zeichen zurückgeben
101 }
102
103 // Timer initialisieren
104 void initTimer(void) {
105     T0PR = 12500;               // Prescaler für den Timer - 12.5 MHz
106     T0TCR = 0x02;               // Timer zurücksetzen mit Bit 1
107     T0MCR = 0x03;               // Interrupt und Reset bei Übereinstimmung
108     T0MR0 = 1000;               // 1000 ms 1 Interrupt
109     T0TCR = 0x00;               // Timer anhalten mit Timer Control Reg.
110
111     // Interrupt-Konfiguration
112     VICVectAddr4 = (unsigned long)T0isr; // addr. von isr -> VIC
113     VICVectCntl4 = 0x24;        // VICVecCntl Bit 5 & 4 : Kanal4
114     VICIntEnable |= 0x10;
115 }
116
117 // Externen Interrupt initialisieren
118 void initExIn(void) {
119     PINSEL0  |= 0x80000000;      // EINT2 aktivieren -> Func. 01
120     EXTMODE  |= 0x04;            // Flankengesteuerter INT. Bit 2
121     EXTPOLAR |= 0x04;            // EINT2 -> rising edge
122     EXTINT   |= 0x04;            // Interrupt Flag
123     VICVectCntl0 = 0x30;         // Kanal 16 : Bit5 :VekKanal, 4: Int.ID
124     VICVectAddr0 = (unsigned long)myEXTINT; // save addr. der isr
125     VICIntEnable = 0x10000;      // EINT2 aktivieren
126 }
127
128 // Integer über UART senden
129 void sendInt(int value) {
130     char buffer[5];             // Bufffer um Int. int richtig. Reih. zu senden
131     int i = 0;                  // Laufindex
132
133                                 // Sonderfall: Wert ist 0, da sonst endlos
134     if (value == 0) {
135         buffer[i++] = '0';      // manuelles zeichen
136     }
137
138                                 // Zahlen in umgekehrter Reihenfolge
139                                 //in den Buffer schreiben
140     while (value > 0) {         // mod extrahiert letzte ziffer
141                                 // '0' -> ASCII Zeichen
142         buffer[i++] = (value % 10) + '0';
143         value /= 10;            // Ganzzahlige Integer Division entf. letzt. Z.
```

```
144       }
145
146                                   // Zahl korrekt herum ausgeben
147       while (i--) {               // Rückwärts durch Buffer zählen
148           uartSendChar(buffer[i]);
149       }
150   }
151
152   // Externer Interrupt-Handler
153   void myEXTINT(void) __irq {
154
155       T0TCR = (T0TCR == 0x01) ? 0x00 : 0x01;
156                                   // Timer starten oder anhalten
157                                   // Falls Timer Läuft Stoppen, sonst starten
158                                   // bei jedem EINT2 wechselt der Timer
159
160                                   // Message ob Timer läuft oder nicht
161
162       uartSendString((T0TCR == 0x01) ? "Timer gestartet!\r\n" : "Timer angehalten!\r\n");
163
164       EXTINT = 0x04;               // Interrupt-Flag löschen -> Setzt EINT2 zurück
165                                    // Sonst : EINT2 kann nicht erneut ausgeführt werden
166       VICVectAddr = 0x00;          // Signalisiert VIC, dass isr fertig
167   }
168
169   // Timer-Interrupt-Handler
170   void T0isr(void) __irq {
171       sek++;                       // vergangene Zeit speichern
172       IOCLR0 = SEGMENT;            // Anzeige zurücksetzen
173
174       IOSET0 = bcd[sek % 10];      // EinerDarstellung der aktuellen Sek.
175
176       T0IR |= 0x10;                // Interrupt-Flag löschen
177       VICVectAddr = 0x00;          // VIC signalisieren, dass isr fertig
178   }
179
180   // Zeit über UART senden
181   void sendTime(int time) {
182       int h = time / 3600;         // Stunden berechnen
183       int m = (time % 3600) / 60;  // Minuten berechnen
184       int s = time % 60;           // Sekunden berechnen
185
186       uartSendString("Zeit: ");
187       if (h < 10) sendInt(0);      // falls h< 10, führende Null
188       sendInt(h);                  // h senden
189       uartSendChar(':');
190       if (m < 10) sendInt(0);      // falls m < 10, führende Null
191       sendInt(m);
192       uartSendChar(':');
193       if (s < 10) sendInt(0);      // falls s < 10, führende Null
194       sendInt(s);
195       uartSendString("\r\n");
196   }
197
198   // Siebensegmentanzeige initialisieren
199   void initSeg(void) {
200       IODIR0 = SEGMENT;  // P0.18-P0.24 als Ausgang setzen
201       IOCLR0 = SEGMENT;  // Ausgang zurücksetzen
202       IOSET0 = bcd[0];   // Anzeige auf 0 setzen
203   }
204
205   // Hauptprogramm
206   int main(void) {
207       char choice;
208       sek = 215990;      // Initiale Zeit setzen
209
210       // Initialisierungen
211       uartInit(initBaudrate(), 8, readSwitchState3(), readSwitchState2(), readSwitchState1());
212                          // UART : baudrate, datenbits, stoppbits, parity
213       initSeg();
214       initExIn();
215       initTimer();
```

```
216
217         // Hauptschleife
218         while (1) {
219             sendMenu();     // menu senden
220             choice = uartReadChar();
221
222             switch (choice) {
223                 case 's': case 'S':
224                             // timer start-stop
225                     T0TCR = (T0TCR == 0x01) ? 0x00 : 0x01;
226                     uartSendString((T0TCR == 0x01) ? "Timer gestartet!\r\n" : "Timer angehalten!\r\n");
227                     break;
228                 case 'a': case 'A':
229                             // Zeit anzeigen lassen
230                     sendTime(sek);
231                     break;
232                             // Timer-Reset
233                 case 'r': case 'R':
234                     if (T0TCR == 0x01) {
235                         uartSendString("Sie muessen die Stoppuhr erst anhalten, erst dann duerfen Sie sie
    zuruecksetzen\r\n");
236                     } else {
237                         T0TCR = 0x02;  // Timer zurücksetzen
238                         sek = 0;
239                         uartSendString("Timer wurde erfolgreich zurueckgesetzt!\r\n");
240                     }
241                     break;
242                 default:
243                     uartSendString("Ungültige Eingabe!\r\n");
244                     break;
245             }
246
247             // Zeitgrenze prüfen und zurücksetzen
248             if (sek >= 215999) {
249                 T0TCR = 0x00;  // Timer anhalten
250                 T0TCR = 0x02;  // Timer zurücksetzen
251                 sek = 0;
252                 uartSendString("aktuelle Timer-Zeit: 59:59:59. Stoppuhr wurde angehalten und
    zurueckgesetzt\r\n");
253             }
254         }
255     }
256
```