```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     VICVectCntl0 = 0x30;         // Kanal 16 : Bit5 :VekKanal, 4: Int.ID
122     VICVectAddr0 = (unsigned long)myEXTINT; // save addr. der isr
123     VICIntEnable = 0x10000;      // EINT2 aktivieren
124 }
125
126 // Integer über UART senden
127 void sendInt(int value) {
128     char buffer[5];             // Bufffer um Int. int richtig. Reih. zu senden
129     int i = 0;                  // Laufindex
130
131                                 // Sonderfall: Wert ist 0, da sonst endlos
132     if (value == 0) {
133         buffer[i++] = '0';      // manuelles zeichen
134     }
135
136                                 // Zahlen in umgekehrter Reihenfolge
137                                 //in den Buffer schreiben
138     while (value > 0) {         // mod extrahiert letzte ziffer
139                                 // '0' -> ASCII Zeichen
140         buffer[i++] = (value % 10) + '0';
141         value /= 10;            // Ganzzahlige Integer Division entf. letzt. Z.
142     }
143
```

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

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