```c
/*****************************************************************/
/*  Hochschule fuer Technik und Wirtschaft                       */
/*  Fakultät fuer Ingenieurwissenschaften                        */
/*  Labor fuer Eingebettete Systeme                              */
/*  Mikroprozessortechnik                                        */
/*****************************************************************/
/*                                                               */
/*  C_Uebung.C:                                                   */
/*    Programmrumpf fuer C-Programme mit dem Keil                */
/*    Entwicklungsprogramm uVision fuer ARM-Mikrocontroller      */
/*                                                               */
/*****************************************************************/
/*  Aufgaben-Nr.:        *                                       */
/*                       *                                       */
/*****************************************************************/
/*  Name / Matrikel-Nr.: *      Valentin Straßer                 */
/*                       *      Michal Roziel                    */
/*                       *                                       */
/*****************************************************************/
/*  Abgabedatum:         *      Donnerstag 16.01.2025            */
/*                       *                                       */
/*****************************************************************/
#include <LPC21xx.H>

// Masken und Konstanten
#define     SEGMENT_MASK     0x1FC0000              // 7-Segment-Anzeige an P0.18-P0.23
#define     LED_MASK         0xFF0000               // LEDs an P1.16-P1.23
#define     SWITCH_MASK      0x30000                // Schalter an P0.16 und P0.17
#define     INPUT_MASK       0x3C00                 // Eingabemaske für BCD
#define     BCD_MAX          9                      // Maximalwert der BCD-Anzeige
#define     LED_MAX          0xFF                   // Maximales LED-Muster

// BCD-Codes für die Zahlen 0-9 auif dem 7 Segement Display
static const unsigned long BCD_CODES[10] = {
    0x0FC0000, 0x180000, 0x16C0000, 0x13C0000, 0x1980000,
    0x1B40000, 0x1F40000, 0x1C0000, 0x1FC0000, 0x1BC0000
};

// Volatile Variablen für ISR-Kommunikation
volatile unsigned int g_ledPattern = 0;             // Aktuelles LED-Muster
volatile unsigned int g_switchState = 0;            // Aktueller Schalter-Status

// Funktionsprototypen
void initLED(void);                                 // LED-Initialisierung
void initBCD(void);                                 // BCD-Anzeige-Initialisierung
void updateBCD(unsigned int value);                 // BCD-Anzeige aktualisieren
void updateLEDs(unsigned int pattern);              // LEDs aktualisieren
unsigned int readInputBCD(void);                    // Eingabe einlesen
void readSwitchState(void);                         // Schalter-Status einlesen
void initTimer(void);                               // Timer initialisieren
void T0isr(void) __irq;                             // Timer-Interrupt-Service-Routine

// LED-Ausgänge initialisieren
void initLED(void) {
    IODIR1 = LED_MASK;                              // P1.16-P1.23 als Ausgang definieren
    IOCLR1 = LED_MASK;                              // Alle LEDs initial ausschalten
}

// BCD-Anzeige initialisieren
void initBCD(void) {
    IODIR0 = SEGMENT_MASK;                          // P0.18-P0.24 als Ausgang definieren
    IOCLR0 = SEGMENT_MASK;                          // BCD-Anzeige initial löschen
}

// BCD-Anzeige mit Wert aktualisieren
void updateBCD(unsigned int value) {
    if (value > BCD_MAX) value = BCD_MAX;           // Wert begrenzen

    IOCLR0 = SEGMENT_MASK;                          // Alte Anzeige löschen
    IOSET0 = BCD_CODES[value];                      // Neuen Wert setzen
}

```

```
 73  // LED-Muster aktualisieren
 74  void updateLEDs(unsigned int pattern) {
 75      IOCLR1 = LED_MASK;                                  // Alle LEDs ausschalten
 76      IOSET1 = (pattern << 16);                           // Neues Muster auf P1.16-P1.23 setzen
 77  }
 78
 79  // Eingabewert von Schaltern einlesen
 80  unsigned int readInputBCD(void) {
 81      return (IOPIN0 >> 10) & 0xF;                        // Bits 10-13 ausmaskieren
 82  }
 83
 84  // Schalter-Status einlesen
 85  void readSwitchState(void) {
 86      g_switchState = (IOPIN0 & SWITCH_MASK) >> 16;  // Bits 16-17 ausmaskieren
 87  }
 88
 89  // Timer-Interrupt-Service-Routine
 90  void T0isr(void) __irq {
 91                                                          // LED-Muster bei aktivem Schalter aktualisieren
 92      if (g_switchState & 0x2) {
 93          g_ledPattern = (g_ledPattern << 1) + 1;     // Neues Bit hinzufügen
 94
 95                                                          // Bei Überlauf zurücksetzen
 96          if (g_ledPattern > LED_MAX) {
 97              g_ledPattern = 0;
 98          }
 99
100          updateLEDs(g_ledPattern);
101      } else {
102          updateLEDs(0);                                  // Alle LEDs ausschalten
103      }
104
105                                                          // Interrupt-Flag zurücksetzen
106      T0IR = 0x01;
107      VICVectAddr = 0x00;
108  }
109
110  // Timer initialisieren
111  void initTimer(void) {
112  // Timer-Konfiguration
113      T0PR = 12500 - 1;                                   // Prescaler für 1 kHz
114      T0MR0 = 500;                                        // Match-Wert für 0.5 Sekunden
115      T0MCR = 0x03;                                       // Interrupt und Reset bei Match
116      T0TCR = 0x01;                                       // Timer starten
117
118   // Interrupt-Konfiguration
119      VICVectAddr0 = (unsigned long)T0isr;                // ISR-Adresse setzen
120      VICVectCntl0 = 0x24;                                // Kanal 4 aktivieren
121      VICIntEnable = 0x10;                                // Timer 0 Interrupt aktivieren
122  }
123
124  int main(void) {
125      unsigned int bcdInput = 0;
126
127  // Hardware initialisieren
128      initBCD();
129      initLED();
130      initTimer();
131
132      while (1) {
133  // Schalter-Status aktualisieren
134          readSwitchState();
135
136  // BCD-Anzeige aktualisieren
137          if (g_switchState & 0x1) {
138              bcdInput = readInputBCD();
139              updateBCD(bcdInput);
140          } else {
141              IOCLR0 = SEGMENT_MASK;                  // BCD-Anzeige löschen
142          }
143      }
144  }
```