```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:         *      16.01.2025                       */
/*****************************************************************/

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

void uartInit(unsigned int baudRate, unsigned int dataBits, unsigned int stopBits, unsigned int
paritySelect, unsigned int parityEnable) {
    unsigned int uartConfig = 0;
    unsigned int Frequenzteiler;
    uartConfig = (paritySelect << 1) + parityEnable;
    uartConfig = (uartConfig << 1) + stopBits;
    uartConfig = (uartConfig << 2) + dataBits;

    PINSEL0 |= UART_PINSEL_CONFIG;  // P0.8 = TxD1, P0.9 = RxD1 für UART1 aktivieren

    Frequenzteiler = PCLOCK / (16 * baudRate);  // Baudratenteiler berechnen
                                                // DLAB : einstellung BAUD
    U1LCR = DLAB_BIT | uartConfig;              // DLAB-Bit setzen, 8 Datenbits, 1 Stoppbit
    U1DLL = Frequenzteiler % 256;               // Niedriges Byte des FrequenzTeilers
    U1DLM = Frequenzteiler / 256;               // Hohes Byte des FrequenzTeilers
    U1LCR = uartConfig;                         // DLAB-Bit löschen
    U1FCR = UART_FIFO_ENABLE;                   // FIFO aktivieren und zurücksetzen
}

// Zeichen über UART senden
void uartSendChar(char data) {
    while ((U1LSR & UART_READY_BIT) == 0);      // Warte, bis das Transmit-Register bereit ist
    U1THR = data;                               // char send, sobald beschrieben -> senden
}

// String über UART senden
void uartSendString(char* str) {
    int i = 0;
    while (str[i] != '\0') {                    // Null Terminierung
        uartSendChar(str[i]);                   // Zeichenweise senden
        i++;
    }
  }

// Zeichen über UART empfangen
char uartReadChar(void) {
    while (!(U1LSR & UART_RX_READY));           // Warte auf empfangenes Zeichen
    return U1RBR;                               // Zeichen zurückgeben
}                                               // DR Bit - Bit 0 in LSR

// Hex Zeichen in numerischen Wert umwandeln
unsigned int hexCharToValue(char c) {
    if (c >= '0' && c <= '9') {
        return c - '0';                         // '0' = 48 , '9' = 57
    } else if (c >= 'A' && c <= 'F') {          // 'A' = 65 , 'F' = 70
        return c - 'A' + 10;                    // 'c' = 97 , 'f' = 102
    } else if (c >= 'a' && c <= 'f') {
        return c - 'a' + 10;
    }
    return INVALID_HEX_VALUE;                   // Ungültiges Zeichen
}
```

```
 72
 73  // Hexadezimale Adresse von Benutzer lesen
 74  void uartReadHexInput(char* inputBuffer, unsigned long* address) {
 75      char receivedChar;
 76      unsigned int hexValue;
 77      int i=0;                                        // Anzahl der empf. zeichen
 78      *address = 0;                                   // Adresse init -> um aufzubauen
 79
 80      while (i < HEX_DIGIT_LIMIT) {
 81          receivedChar = uartReadChar();              // Zeichen empfangen
 82          uartSendChar(receivedChar);                 // Echo des Zeichens -> bestätigung
 83
 84          if (receivedChar == '\r') {                 // Eingabe beenden
 85              break;
 86          }
 87                                                      // Umwandlung des Zeichens in Hex-Wert
 88          hexValue = hexCharToValue(receivedChar);
 89          if (hexValue != INVALID_HEX_VALUE) {
 90                                                      // ADDR UPDATE & Zeichen in Puffer schreiben
 91              *address = (*address << 4) + hexValue;
 92              inputBuffer[i++] = receivedChar;
 93              } else {
 94               uartSendString(" Ungueltiges Zeichen! Bitte nur Hex-Zeichen verwenden (0-9, a-f, A-F)."
    CR_LF);
 95              }
 96      }
 97          inputBuffer[i] = '\0';                      // Nullterminierung für String
 98                                                      // Zusätzliche Zeichen nach Eingabe verwerfen
 99      while (receivedChar != '\r') {
100          receivedChar = uartReadChar();
101          uartSendChar(receivedChar);
102          if (receivedChar != '\r') {
103              uartSendString(" Ungueltiges Zeichen nach der Eingabe! Bitte mit Enter abschliessen." CR_LF);
104          }
105      }
106
107  }
108
109  // Speicherinhalt als Hexadezimalwerte ausgeben - Jedes Byte als Zwei Zeichen ausgeben
110  void memoryDumpHex(unsigned long address, unsigned int length) {
111      char* ptr = (char*) address;                    // Speicheradresse auf char Zeiger umwandeln,
112      int i;                                          // um Byteweise zugreifen zu können
113      for (i = 0; i < length; i++) {
114          unsigned char value = ptr[i];               // aktuelles Byte auslesen
115
116          // Upper-Byte Hälfte und Lower-ByteHälfte extrahieren
117          char upperHalf = (value >> 4) & 0x0F ;    // um 4 shiften und mask. zum extrahieren
118          char lowerHalf = value & 0x0F;              // Bit-maske
119
120          uartSendChar(upperHalf < 10 ? '0' + upperHalf : 'A' + (upperHalf - 10));
121                                                      // lowerHalf analog zu upperHalf
122          uartSendChar(lowerHalf < 10 ? '0' + lowerHalf : 'A' + (lowerHalf - 10));
123
124          uartSendChar(' ');                          // Leerzeichen für visuelles
125      }
126                                                      // Sende eines Returns und Line-Feed
127      uartSendString(CR_LF);
128  }
129
130  int main(void) {
131      char inputBuffer[HEX_DIGIT_LIMIT + 1];  // Platz für die eingegebene Adresse (8 Zeichen +
    Nullterminierung)
132      unsigned long address;
133
134      uartInit(BAUDRATE, 3, 1, 1, 1);  // UART initialisieren
135
136      uartSendString("0123456789");
137      uartSendString(CR_LF);
138
139      while (1) {
140          uartReadHexInput(inputBuffer, &address);
141          uartSendString(CR_LF);
```

```
142
143             uartSendString(inputBuffer);  // Original-Eingabe anzeigen
144             uartSendString(": ");
145
146             memoryDumpHex(address, 16);  // Speicherinhalt anzeigen
147             uartSendString(CR_LF"\n");
148         }
149     }
150
```