

```

1  /*****
2  /* Hochschule fuer Technik und Wirtschaft
3  /* Fakultät fuer Ingenieurwissenschaften
4  /* Labor fuer Eingebettete Systeme
5  /* Mikroprozessortechnik
6  *****/
7  /*
8  /* C_Uebung.C:
9  /* Programmrumpf fuer C-Programme mit dem Keil
10 /* Entwicklungsprogramm uVision fuer ARM-Mikrocontroller
11 /*
12 *****/
13 /* Name / Matrikel-Nr.: *      Valentin Straßer
14 /*                      *      Michal Roziel
15 *****/
16 /* Abgabedatum:      *      16.01.2025
17 *****/
18
19 #include <LPC21xx.H> // LPC21xx Mikrocontroller Definitionen
20 #include "C_Uebung.H"
21
22 void uartInit(unsigned int baudRate, unsigned int dataBits, unsigned int stopBits, unsigned int
paritySelect, unsigned int parityEnable) {
23     unsigned int uartConfig = 0;
24     unsigned int Frequenzteiler;
25     uartConfig = (paritySelect << 1) + parityEnable;
26     uartConfig = (uartConfig << 1) + stopBits;
27     uartConfig = (uartConfig << 2) + dataBits;
28
29     PINSEL0 |= UART_PINSEL_CONFIG; // P0.8 = TxD1, P0.9 = RxD1 für UART1 aktivieren
30
31     Frequenzteiler = PCLOCK / (16 * baudRate); // Baudratenteiler berechnen
32                                           // DLAB : einstellung BAUD
33     U1LCR = DLAB_BIT | uartConfig; // DLAB-Bit setzen, 8 Datenbits, 1 Stoppbit
34     U1DLL = Frequenzteiler % 256; // Niedriges Byte des FrequenzTeilers
35     U1DLM = Frequenzteiler / 256; // Hohes Byte des FrequenzTeilers
36     U1LCR = uartConfig; // DLAB-Bit löschen
37     U1FCR = UART_FIFO_ENABLE; // FIFO aktivieren und zurücksetzen
38 }
39
40 // Zeichen über UART senden
41 void uartSendChar(char data) {
42     while ((U1LSR & UART_READY_BIT) == 0); // Warte, bis das Transmit-Register bereit ist
43     U1THR = data; // char send, sobald beschrieben -> senden
44 }
45
46 // String über UART senden
47 void uartSendString(char* str) {
48     int i = 0;
49     while (str[i] != '\0') { // Null Terminierung
50         uartSendChar(str[i]); // Zeichenweise senden
51         i++;
52     }
53 }
54
55 // Zeichen über UART empfangen
56 char uartReadChar(void) {
57     while (!(U1LSR & UART_RX_READY)); // Warte auf empfangenes Zeichen
58     return U1RBR; // Zeichen zurückgeben
59 } // DR Bit - Bit 0 in LSR
60
61 // Hex Zeichen in numerischen Wert umwandeln
62 unsigned int hexCharToValue(char c) {
63     if (c >= '0' && c <= '9') {
64         return c - '0'; // '0' = 48 , '9' = 57
65     } else if (c >= 'A' && c <= 'F') { // 'A' = 65 , 'F' = 70
66         return c - 'A' + 10; // 'c' = 97 , 'f' = 102
67     } else if (c >= 'a' && c <= 'f') {
68         return c - 'a' + 10;
69     }
70     return INVALID_HEX_VALUE; // Ungültiges Zeichen
71 }

```

```

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(" Ungültiges Zeichen! Bitte nur Hex-Zeichen verwenden (0-9, a-f, A-F).");
95             CR_LF);
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(" Ungültiges 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 +
132     // Nullterminierung)
133     unsigned long address;
134
135     uartInit(BAUDRATE, 3, 1, 1, 1); // UART initialisieren
136
137     uartSendString("0123456789");
138     uartSendString(CR_LF);
139
140     while (1) {
141         uartReadHexInput(inputBuffer, &address);
142         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
```