

```

1  ;*****
2  ;* htw saar - Fakultaet fuer Ingenieurwissenschaften *
3  ;* Labor fuer Eingebettete Systeme *
4  ;* Mikroprozessortechnik *
5  ;*****
6  ;* Assembler_Startup.S: *
7  ;* Programmruempf fuer Assembler-Programme mit dem Keil *
8  ;* Entwicklungsprogramm uVision fuer ARM-Mikrocontroller *
9  ;*****
10 ;* Aufgabe-Nr.: * 2.4 *
11 ;* * *
12 ;*****
13 ;* Gruppen-Nr.: * Donnerstagsgruppe 1 (14:15-15:45) *
14 ;* * *
15 ;*****
16 ;* Name / Matrikel-Nr.: * Finn Ottinger 5014272 *
17 ;* * Maximilian Kany 5016118 *
18 ;* * *
19 ;*****
20 ;* Abgabedatum: * 18.12.2025 *
21 ;* * *
22 ;*****
23
24 ;*****
25 ; Symboldefinitionen *
26 ;*****
27 RAM_Size EQU 0x0800 ; z. B. 2 KB Datenspeicher
28
29 ;*****
30 ;* Daten-Bereich bzw. Daten-Speicher *
31 ;*****
32 AREA Daten, DATA, READWRITE
33 Datenanfang
34 ALIGN
35
36 Top_Stack EQU Datenanfang + RAM_Size
37 Datenende EQU Top_Stack
38 Ergebnis_BCD DCD 0 ; Zum Speichern des BCD-Ergebnisses
39
40 ;*****
41 ;* Programm-Bereich bzw. Programm-Speicher *
42 ;*****
43 AREA Programm, CODE, READONLY ; Setzt den Startpunkt
44 ARM
45
46 Reset_Handler
47 MSR CPSR_c, #0x10 ; User Mode aktivieren
48
49 ;*****
50 ;* Hauptprogramm *
51 ;*****
52 HauptProgramm
53 ; --- Test 1: uItoBCD-Aufruf (Aufgabe 2.3)
54 LDR SP, =Top_Stack
55 LDR R0, =TestString
56 BL Berechnung
57
58 endlos
59 B endlos ; Endlosschleife
60
61 ;*****
62 ;* Unterprogramm: uItoBCD (Unsigned Integer to Packed BCD) *
63 ;* Eingang: R0 = Unsigned Integer (32-Bit) *
64 ;* Ausgang: R0 = Gepackte BCD-Zahl (32-Bit) *
65 ;* Verwendet: R1-R5, LR *
66 ;*****
67 Berechnung
68 ; R0_in = Adresse String
69 STMFD sp!, {LR}
70
71 ; 1. X bestimmen (AtoI)
72 BL AtoI ; R0 = X (signed int)
73
74 ; 2. Y berechnen (Formel: Y = f(X))
75 BL Formel ; R0 = Y (signed int)
76
77 ; 3. Y nach BCD konvertieren (uItoBCD)

```

```

78      ; Achtung: Y muss positiv sein, da uItoBCD unsigned ist. (Unsere Formel liefert Y >= 0)
79      BL      uItoBCD      ; R0 = Y_BCD
80
81      LDMFD    sp!, {LR}
82      BX      LR
83
84  Formel
85      STMFD    SP!, {LR}      ; Speichert Register LR auf dem Stack (Sicherung des Kontexts)
86
87      ; Konstante Laden
88      MOV      R1, R0      ; Lädt den Wert von der Adresse in R0 (X_Wert) in Register R1
89      LDR      R2, =KONSTANTE_9      ; Lädt die Adresse der Konstante 0x38E38E39 in R2
90
91      MUL      R0, R1, R1      ; Berechnet R1 * R1 (Quadrat von X_Wert) und speichert das Ergebnis
in R0
92
93      UMULL    R3, R4, R0, R2      ; Führt eine 64-Bit-Multiplikation von R0 und R1 durch, Ergebnis in
R3 (niederwertig) und R4 (höherwertig)
94      MOV      R0, R4, LSR #1      ; Schiebt das höherwertige Ergebnis (R4) um 1 Bit nach rechts
(Division durch 2)
95
96      MOV      R0, R0, LSL #2      ; Schiebt das Ergebnis um 2 Bits nach links (Multiplikation mit 4)
97
98      LDMFD    SP!, {LR}      ; Stellt die gesicherten Register LR vom Stack wieder her
99      BX      LR      ; Springt zurück zur Aufrufadresse (Rückkehr aus dem Unterprogramm)
100
101  AtoI
102      ; R4 (Ergebnis), R2 (Vorzeichen), R1 (temporär), LR sichern
103      STMFD    sp!, {LR}
104      ; Input R0: Adresse String
105      ; Output R0: Ergebnis
106
107      MOV      R4, #0      ; R4 = Ergebnis (Akku) = 0
108      MOV      R2, #1      ; R2 = Vorzeichen = +1
109
110      ; 1. Vorzeichen prüfen (Zeichen an R0 laden in R3)
111      LDRB     R3, [R0], #1
112
113      CMP      R3, #ASCII_MINUS
114      MOVEQ    R2, #-1      ; R2 = -1 (negativ)
115      CMPNE    R3, #ASCII_PLUS
116      LDRBEQ   R3, [R0], #1      ; Zeichen laden, Pointer R0 erhöhen
117
118  parse_digits
119      SUBS     R3, R3, #ASCII_ZERO ; ASCII nach Zahl (z.B. '5' -> 5)
120
121      ; R4 = R4 * 10 + R3 (Multiplikation mit 10)
122      ADDPL    R4, R4, R4, LSL #2      ; R4 = R4 * 5
123      LSLPL    R4, R4, #1      ; R4 = R4 * 10
124      ADDPL    R4, R4, R3      ; R4 = R4 + Ziffer
125
126      LDRBPL   R3, [R0], #1      ; Zeichen laden, Pointer R0 erhöhen
127      BPL     parse_digits      ; Nächste Ziffer
128
129      ; 3. Vorzeichen anwenden
130  apply_sign
131      CMP      R2, #1
132      RSBNE    R4, R4, #0      ; Wenn R2 != 1 (negativ), dann R4 = 0 - R4 (Negieren)
133
134      MOV      R0, R4      ; Ergebnis in R0 für den Rücksprung
135
136      LDMFD    sp!, {LR}      ; R1-R4 wiederherstellen
137      BX      LR
138
139  uItoBCD
140      ; Sichert alle verwendeten Register (inkl. LR) und verwendet STMFD
141      STMFD    sp!, {R1-R7, LR}
142
143      MOV      R1, #0      ; R1 = BCD-Ergebnis = 0
144      MOV      R3, #0      ; R3 = Bit-Verschiebung (shift_count = 0)
145      LDR      R5, =INV_10_C      ; R5 = 0xCCCCCCCC (Kehrwert für Division)
146
147  LOOP_START
148      ; 1. Division: R4 (Quotient) = R0 / 10
149      ; R4 enthält den oberen 32-Bit-Teil des 64-Bit-Ergebnisses (Quotient)
150      UMULL    R6, R4, R0, R5      ; R6:RdLo, R4:RdHi = R0 * R5
151      MOVS     R4, R4, LSR #3      ; R4 = R4 / 8 (um den Faktor 8 zu kompensieren)

```

```
152
153     ; 2. Rest (Ziffer) berechnen: Rest = Dividend - Quotient * 10
154     MOV     R6, #10                ; R6 = 10
155     MUL     R6, R4, R6              ; R6 = Quotient * 10
156
157     SUB     R2, R0, R6              ; R2 = Dezimalziffer (Rest) = R0 - (Quotient * 10)
158
159     ; 3. Nächster Dividend setzen
160     MOV     R0, R4                  ; R0 = Quotient (für den nächsten Schleifendurchlauf)
161
162     ; 4. BCD-Verpackung
163     LSL     R2, R2, R3              ; Schiebe die Ziffer (R2) an die richtige BCD-Position
164     ORR     R1, R1, R2              ; Füge die Ziffer in das BCD-Ergebnis (R1) ein
165
166     ADD     R3, R3, #4              ; Erhöhe den Shift-Wert um 4 Bit
167
168     BNE     LOOP_START
169
170     CONVERSION_DONE
171     MOV     R0, R1                  ; Finales Ergebnis in R0 kopieren
172
173     LDMFD   sp!, {R1-R7, LR}       ; Register wiederherstellen
174     BX      LR
175
176     ;*****
177     ;* Konstanten im CODE-Bereich                                     *
178     ;*****
179     INV_10_C    EQU 0xCCCCCCD
180
181     ASCII_ZERO  EQU '0'
182     ASCII_PLUS  EQU '+'
183     ASCII_MINUS EQU '-'
184
185     KONSTANTE_9 EQU 0x38E38E39      ; Definiert die DCD-Konstante 0x38E28E39 (z. B. für
Farbdemodulation)
186
187     TestString  DCB "+100"
188
189     Ergebnis    DCD 0
190
191     ;*****
192     ;* Ende der Programm-Quelle                                     *
193     ;*****
194     END
195
```