```
;*********************************************************
;  Hochschule für Technik und Wirtschaft Dresden(FH)    *
;  FB ET,       Praktikum Mikrocontrollertechnik        *
;---------------------------------------------------------*
;Projekt:  Microcontrollersteuerung                      *
;Modul:    prakt.a51                                     *
;System:   SAB80C517A ( MCB517AC/KEIL )                  *
;Aufgabe:  Rahmenprogramm zu allen Praktikumsaufgaben    *
;          In diesen 8051-Programmrumpf sind an den      *
;          geeigneten Stellen die notwendigen Programm-  *
;          zeilen einzufügen.                            *
;---------------------------------------------------------*
;Autor: J. Huhle HTWD                                     *
;       huhle@et.htw-dresden.de         Tel.: HA 2542    *
;Datum: 08.05.2002                                       *
;---------------------------------------------------------*
;bearbeitet von:                                          *
;*********************************************************

$include(reg517a.inc)     ;Register_Symboldefinitionen

;Definitionen für verschiebare (relocatable)...
mainprog  segment  code   ; ... Programm-
const     segment  code   ; Konstanten-
main_data segment  data   ; Daten-
;main_bits segment  bit    ; Bitsegmente

;***  glob. Nutzervereinbarungen und -definitionen  ***
;Konstanten
  t0rel equ     0-50000 ; T0_Reloadwert
  mask1 equ     0efh    ; Init.wert für MASKE
  user_psw  equ 8       ; Name für eine Registerbank_Adresse

;Variable in der Reg_Bank 1  (Beispiele)
        dseg at 8               ;nutzt RG1 R0.. max. 8 Byte
MASKE:   ds     1       ;ein Byte für MASKE res. (entspricht R0)
POINTER: ds   1 ;ein Byte für POINTER res. (entspricht R1)

;Variable im frei nutzbarem Reg_bereich (ein Beispiel)
        rseg main_data
A_normiert:     ds      2               ; lo/hi Byte für Normierten Analogwert
A_Mittelwert:   ds      2               ; lo/hi Byte des Mittelwertes, gerundet
LCD_Daten:      ds      8               ; Char's fürs Display
RP_Daten:       ds      32              ; lo/hi Byte Ringpuffer
RP_Hilfswert:   ds      1               ; aktuelle Position im Ringpuffer
T0_Hilfswert:   ds      1               ; Erweiterung Timer0
        iseg  at 0e0h
  usersp:  ds   20h     ; Reservierung für Stack ab E0H

;---------------------------------------------------------------------
;------- Interrupteinsprungadressen ------------------------------
        cseg    at 0
        JMP     start                   ; Hauptprogramm

        cseg at 0bh
        JMP     t0_isr                  ; Einsprung der T0-OV-ISR

        cseg at 23h
        ds      3                       ; Einsprünge der UART-ISR

        cseg at 83h                     ; ser.Kanal 1
        ds      3                       ; reserviere 3 Byte für Debugger-ISR
```

```
;------------------------------------------------------------------
;------- Programmstart ab PC=0100 -----------------------------------
        rseg mainprog
        org     0100h
start:
        ; Initialisierungen ------------------------------
        CLR     EAL                     ; alle INT's sperren
        MOV     sp,#usersp-1            ; Stackpointer festlegen
        MOV     psw,#0
        ; On-Chip Peripherie initialisieren --------------
        CALL    OnChipInit
        SETB    eal                     ; INT freigeben
        ; Port P4 vorladen -------------------------------
        MOV     P4, #01h
        ; Display initialisiern und Texte ausgeben --------
        ;CALL   LCD_INIT


        ; Hauptprogrammschleife --------------------------
main:

        JMP     main

;------------------------------------------------------------------
;-------- T0_ISR   Basiszeitgeber -----------------------------------
t0_isr:
        ; Timer Reload & Register sichern -----------------
        MOV     th0,#high t0rel         ;T0_reload
        MOV     tl0,#low  t0rel
        PUSH    psw
        PUSH    acc
        MOV     psw,#user_psw
        ; Lauflicht an P4 --------------------------------
        MOV     A, P4
        RL      A                       ;LED-Kette inkrementieren
        MOV     P4, A
        ; Timererweiterung holen -------------------------
        MOV     A, T0_Hilfswert ; Wert holen
        INC     A                       ; incrementieren
        ANL     A, #03d                 ; auf 0..3 begrenzen
        MOV     T0_Hilfswert, A         ; sichern
        ; UP-Aufrufe -------------------------------------
        CJNE    A, #00, L1
        CALL    AN_FETCH
        SJMP    t0isren
L1:
        CJNE    A, #01, L2
        CALL    AN_AVERAGE
        SJMP    t0isren
L2:
        CJNE    A, #02, L3
        CALL    AN_TO_STRING
        SJMP    t0isren
L3:
        CALL    STR_TO_LCD
        ; fertig -----------------------------------------
t0isren:POP     acc
        POP     psw
        RETI

;------------------------------------------------------------------
;-------- OnChip Init -----------------------------------------------
```

```
OnChipInit:
        ; Timer0-Initialisierung -------------------------
        MOV     th0,#high t0rel
        MOV     tl0,#low  t0rel
        MOV     tmod,#1                 ; 16-bit_timer
        SETB    et0                     ; INT_Freigabe
        SETB    tr0                     ; T0-start
     ;u.U. ser.Schnittstelle initialisieren
        ;siehe conea0.inc
        ; INT-Prioritäten festlegen ----------------------
        MOV     ip0,#00000000b  ;
        MOV     ip1,#00000010b  ;T0 auf Stufe2
        RET                     ;Ende des UP

;---------------------------------------------------------------------
;-------- Einfügen von Treibern --------------------------------------
;$include ( conea0.inc )
$include ( analog.inc )

;---------------------------------------------------------------------
;-------- Standarttexte ----------------------------------------------
        rseg const      ;z.B. für LCD-Texte
LCD_Text1:
        db      'LCD-OK',0
LCD_Text2:
        db      '0,000 Volt',0

        end                     ;Programmende
```

```
;------------------------------------------------- M.Lipinsky ----
; ------ Analogwert holen & Normieren -------------------------------
AN_FETCH:
        PUSH    ACC
        PUSH    PSW
        ; Analogwert holen ------------------------------
        MOV     A, P1
        ;MOV    A, #01d
        ;CALL   ANIN
        ; Normieren --------------------------------------
        MOV     B, #10d                 ; Normierung auf 5V
        MUL     AB                      ; und Upsamplen mit 2
        ; im RAM ablegen ---------------------------------
        MOV     r0, #A_normiert
        MOV     @r0, A
        INC     r0
        MOV     @r0, B
        ; fertig -----------------------------------------
        POP     PSW
        POP     ACC
        RET

;------------------------------------------------- M.Lipinsky ----
; ------ Analogwert Mittelwert bilden -------------------------------
AN_AVERAGE:
        PUSH    ACC
        PUSH    PSW
        ; aktuelle Position im RP ermitteln --------------
        MOV     A, RP_Hilfswert
        ADD     A, #2                   ; nächste Position
        ANL     A, #1Fh                 ; auf 0..31 begrenzen
        MOV     RP_Hilfswert, A
        ; Pointer bilden ---------------------------------
        ADD     A, #RP_Daten            ; Start RP
        MOV     r0,A                    ; akt Pos addieren
        ; Wert A_Normiert im RP eintragen -----------------
        MOV     @r0, A_Normiert         ; kopiere LowByte
        INC     r0
        MOV     @r0, A_Normiert+1       ; kopiere HighByte
        ; Summenschleife vorbereiten ----------------------
        MOV     r0, #16d                ; Laufvariable
        MOV     r1, #RP_Daten           ; Pointer
        MOV     r2, #00                 ; LowByte
        MOV     r3, #00                 ; HighByte
        ; Summenbildung -----------------------------------
SummenSchleife:
        CLR     C
        MOV     A, @r1                  ; LowByte
        ADD     A, r2
        MOV     r2, A
        INC     r1
        MOV     A, @r1                  ; HighByte
        ADDC    A, r3
        MOV     r3, A
        INC     r1
        DJNZ    r0, SummenSchleife
        ; Division (durch 32) ----------------------------
        MOV     r0, #05d
Teilen:
        CLR     C                       ; HighByte
        MOV     A, r3
        RRC     A
```

```
        MOV     r3, A
        MOV     A, r2                   ; LowByte
        RRC     A
        MOV     r2, A
        DJNZ    r0, Teilen
        ; Rundung ---------------------------------------
        MOV     A, #00d                 ; LowByte
        ADDC    A, r2
        MOV     r2, A
        MOV     A, #00d                 ; HighByte
        ADDC    A, r3
        MOV     r3, A
        ; im RAM ablegen ---------------------------------
        MOV     r0, #A_Mittelwert+1
        MOV     @r0, A
        DEC     r0
        MOV     A, r2
        MOV     @r0, A
        ; fertig -----------------------------------------
        POP     PSW
        POP     ACC
        RET

;------------------------------------------------------ M.Lipinsky ----
; ------ Analogwert in String umwandeln -------------------------------
AN_TO_STRING:
        PUSH    ACC
        PUSH    PSW
        ; anzuzeigenden MW holen -------------------------
        MOV     r0, #A_Mittelwert
        MOV     A, @r0                  ; LowByte
        MOV     r6, A
        INC     r0                      ; HighByte
        MOV     A, @r0
        MOV     r7, A
        ; Zähler vorladen --------------------------------
        MOV     r4, #00d                ; Ziffernzähler
        MOV     r3, #01d                ; Stellenzähler
        MOV     r2, #00d                ; Ausgabezähler
        MOV     r1, #LCD_Daten          ; Pointer auf Puffer
        ; akt. Stellenwertigkeit in Zehnerpotenz ----------
StellenSchleife:
        DEC     r3                      ; Stellenzähler dekr
        ; Unterdrückung führender Nullen -----------------
normaleStelle:
        CJNE    r7, #00d, toString      ; HighByte=0 und
        CJNE    r4, #00d, toString      ; Ziffernzähler=0
        ; dann Leerzeichen ausgeben
        MOV     @r1, #32d
        AJMP weiter
        ; als String ausgeben ----------------------------
toString:
        MOV     A, 30h                  ; Offset lt. ASCII-Tabelle
        ADD     A, r7                   ; auszugebende Ziffer addieren
        MOV     @r1, A                  ; im Puffer ablegen
        INC     r4                      ; Ziffernzähler erhöhen
weiter:
        INC     r2                      ; Ausgabezähler und
        INC     r1                      ; Pointer erhöhen
        ; Punkt notwendig? -------------------------------
        CJNE    r3, #00d, naechsteStelle
        CJNE    r4, #01d, naechsteStelle
```

```
        MOV     @r1, #46                ; Dezimalpunkt
        INC     r2                      ; Ausgabezähler und
        INC     r1                      ; Pointer erhöhen
        ; nächste Ziffer errechnen ------------------------
naechsteStelle:
        MOV     A, r6                   ; LowByte mal 10
        MOV     B, #10d
        MUL     AB
        MOV     r7, B
        MOV     r6, A
        ; alles ausgegeben? -------------------------------
        CJNE    r3, #(0-3), StellenSchleife
        ; 'Milli' ausgeben? -------------------------------
        CJNE    r2, #04, toStringFertig
        MOV     @r1, #'m'
toStringFertig:
        ; fertig -------------------------------------------
        POP     PSW
        POP     ACC
        RET

;------------------------------------------------------ M.Lipinsky ----
; ------ String ausgeben ----------------------------------------------
STR_TO_LCD:
        PUSH    ACC
        PUSH    PSW
        ; auf Zeile0, Spalte 2 schalten -------------------
        MOV     A, #194
        ; CALL  LCD_CONTROL
        ; Ausgabe des Strings ------------------------------
        MOV     r0, #04                 ; Zähler
        MOV     r1, #LCD_Daten          ; Pointer auf Puffer
AusgabeSchleife:
        MOV     A, @r1                  ; Char holen
        ;CALL   LCD_PUTCHAR             ; und ausgeben
        INC     r1                      ; nächstes Zeichen
        DJNZ    r0, AusgabeSchleife
        ; fertig -------------------------------------------
        POP     PSW
        POP     ACC
        RET
```