```
;*****************************************************************
MULT24x24:
   NOLIST
; Multiplkation 24Bit x 24Bit Produkt 48Bit
;Source: www.piclist.com/techref/microchip/math/mul/24x24b-tk.htm
;  CBLOCK
;     ACb5,ACb4,ACb3    ;ACb2:0 Multiplikant
;     ACb2.ACb1,ACb0    ;ACb5:0 Produkt
;     BCb2,BCb1,BCb0    ;BCb2:0 Multiplikator
;     cntBit1,cntBit0   ;Hilfsvariable
;  ENDC
;
;  ACb2:0   Multiplikant
;  BCb2:0   Multiplikator
;  ACb5:0   Produkt
;
;--MULT24x24_Test
;   ; preload values to test
;   MOVLW    0xAB    ;Multiplikant 0xABCDEF = 11.259.375
;   MOVWF    ACb2
;   MOVLW    0xCD
;   MOVWF    ACb1
;   MOVLW    0xEF
;   MOVWF    ACb0
;   ;
;   MOVLW    0x98    ;Multiplikator 0x987654 = 9.991.764
;   MOVWF    BCb2
;   MOVLW    0x76
;   MOVWF    BCb1
;   MOVLW    0x54
;   MOVWF    BCb0   ;ACb5:0 = 0x6651AF33BC6C=112.501.017.787.500
;--Test Ende
   LIST
   CLRF     ACb5           ;clear destination
   CLRF     ACb5+1         ;addres ACb5 + 1
   CLRF     ACb5+2
   MOVLW    D'24'
   MOVWF    cntBit1        ;number of bits
   RRF      ACb5+3,F       ;shift out to carry
   RRF      ACb5+4,F       ;next multiplier bit
   RRF      ACb5+5,F
ADD_LOOP_24x24
   BTFSS    STATUS,C       ;if carry is set we must add multipland
                           ;to the product
   GOTO     SKIP_LOOP_24x24;nope, skip this bit

   MOVF     BCb2+2,W       ;get LSB of multiplicand
   ADDWF    ACb5+2,F       ;add it to the lsb of the product
   MOVF     BCb2+1,W       ;middle byte
   BTFSC    STATUS,C       ;check carry for overflow
   INCFSZ   BCb2+1,W       ;if carry set we add one to the source
   ADDWF    ACb5+1,F       ;and add it  (if not zero, in
            ;that case mulitpland=0xff->0x00)
   MOVF     BCb2,W         ;MSB byte
   BTFSC    STATUS,C       ;check carry
   INCFSZ   BCb2,W
   ADDWF    ACb5,F         ;handle overflow
   ;
SKIP_LOOP_24x24
   ;note: carry contains most significant bit of addition here
   ;shift in carry and shift out
   ;next multiplier bit, starting from less significant bit
   RRF      ACb5,F
   RRF      ACb5+1,F
   RRF      ACb5+2,F
   RRF      ACb5+3,F
```

```
   RRF      ACb5+4,F
   RRF      ACb5+5,F
   DECFSZ   cntBit1,F
   GOTO     ADD_LOOP_24x24
   RETURN
;*****************************************************************
SUB16_16:
   movf     BCb0,W
   subwf    ACb0,f
   movf     BCb1,W
   btfss    STATUS,C
   incfsz   BCb1,W
   subwf    ACb1,f
   btfsc    STATUS,C          ;C=1 Ergebnis = 0 oder Positiv
   GOTO     sub16_ende
   movf     BCb1,w            ;STATUS,C=0
   movwf    ACb1
   movf     BCb0,w
   movwf    ACb0
 sub16_ende:
   RETURN
;*****************************************************************
;*****************************************************************
DIV_48by24:
   NOLIST
; Source: http://www.piclist.com/techref/microchip/math/div/48by24ng.htm
; Die Variablen wurden von littel endian zu big endian umbenannt
; Die Prüfung ergab richtige Ergebnisse. Ottmar
;
; Routine to divide a 48 bit number with a 24 bit number
; result upto 48 bits !
;
; Formula:      ACb5:0 = ACb5:0 / BCb2:0
;
;   CBLOCK
;      ACb5,ACb4,ACb3   ;Dividend und Quotient
;      ACb2,ACb1,ACb0
;      BCb2,BCb1,BCb0   ;Divisor
;      Temp2,Temp1,Temp0,Temp
;      cntBit
;   ENDC
;
; Format:       big endian, Dividen5, BCb3 = msb
;                           Dividen0, BCb0 = lsb
; Ram used:     ACb 6 bytes ( 48 bits )
;               BCb 3 bytes ( 24 bits )
;               Temp 3 bytes
;               cntBit 1 byte for loop counting ( and temp. saving of carry
)
;
; BCb is preserved
; ACb holds result
; Returns with zero in W if failed ( division with zero )
; Else returns with one in w
;
; Based on pseudo-code from Nikolai Golovchenko [golovchenko@mail.ru]
; max time in loop: 30 cycles
;-----------------------------------------------------------------------
   LIST

        MOVF    BCb2,W    ; Test for zero division
        IORWF   BCb1,W
        IORWF   BCb0,W
        BTFSC   STATUS,Z
        RETLW   0x00          ; divisor = zero, not possible to calculate
return with zero in w
```

```
        ; prepare used variables
        CLRF    Temp2
        CLRF    Temp1
        CLRF    Temp0

        clrf    Temp;;;;;

        MOVLW   D'48'           ; initialize bit counter
        MOVWF   cntBit

DIVIDE_LOOP_48by24
        RLF     ACb0,F
        RLF     ACb1,F
        RLF     ACb2,F
        RLF     ACb3,F
        RLF     ACb4,F
        RLF     ACb5,F
        ; shift in highest bit from dividend through carry in temp
        RLF     Temp0,F
        RLF     Temp1,F
        RLF     Temp2,F

        rlf     Temp, f;;;;;

        MOVF    BCb0,W     ; get LSB of divisor
        btfsc   Temp2, 7
        goto    Div48by24_add

        ; subtract 24 bit divisor from 24 bit temp
        SUBWF   Temp0,F        ; subtract

        MOVF    BCb1,W     ; get middle byte
        SKPC                    ;  if overflow ( from prev. subtraction )
        INCFSZ  BCb1,W     ; incresase source
        SUBWF   Temp1,F        ; and subtract from dest.

        MOVF    BCb2,W       ; get top byte
        SKPC                     ;  if overflow ( from prev.subtraction )
        INCFSZ  BCb2,W       ; increase source
        SUBWF   Temp2,F          ; and subtract from dest.

        movlw 1
        skpc
        subwf   Temp, f;;;;;
        GOTO    DIVIDE_SKIP_48by24 ; carry was set, subtraction ok,
continue with next bit

Div48by24_add
        ; result of subtraction was negative restore temp
        ADDWF   Temp0,F        ; add it to the lsb of temp

        MOVF    BCb1,W     ; middle byte
        BTFSC   STATUS,C        ; check carry for overflow from previous
addition
        INCFSZ  BCb1,W     ; if carry set we add 1 to the source
        ADDWF   Temp1,F        ; and add it if not zero in that case
Product+Multipland=Product

        MOVF    BCb2,W       ; MSB byte
        BTFSC   STATUS,C        ; check carry for overflow from previous
addition
        INCFSZ  BCb2,W
        ADDWF   Temp2,F          ; handle overflow

        movlw 1
```

```
        skpnc
        addwf   Temp, f;;;;;

DIVIDE_SKIP_48by24
        DECFSZ  cntBit,F      ; decrement loop counter
        GOTO    DIVIDE_LOOP_48by24      ; another run
        ; finally shift in the last carry
        RLF     ACb0,F
        RLF     ACb1,F
        RLF     ACb2,F
        RLF     ACb3,F
        RLF     ACb4,F
        RLF     ACb5,F
        RETLW   0x01    ; done
;*****************************************************************
```