Microcontroller
Selbststudium Semesterwoche 5

2. Bitmanipulationen

2.1 Schreiben Sie ein kleines Programm, welches das Carry-Bit mittels Bit-Maskierung im CCR löscht! (Kontrolle, indem Carry zuerst mit Monitor gesetzt wird, dh. Befehl „RS“!)

2.2 Schreiben Sie ein kleines Programm, welches das Overflow-Bit mittels Bit-Maskierung im CCR setzt!

2.3 Es gibt einige Befehle, welches das direkte Setzen (Sxx von SET!) bzw. Löschen (Cxx von CLEAR) zulassen. Wie heissen die Befehle, welche dasselbe wie unter Aufgabe *2.1* und *2.2* tun?

2.4 Ändern Sie die Programme unter 2.1. und 2.2 so ab, dass die gefundenen Befehle benützt werden.

```
Aufgabe21:          LDA     #$FF
                            STA             PTCDD

                            TPA
                            STA             PTCD

                            AND             %11111110
                            STA             PTCD

                            TAP
EndAufgabe21 BRA EndAufgabe21


Aufgabe22:          LDA             #$FF
                            STA             PTCDD

                            TPA
                            STA             PTCD

                            ORA             %10000000
                            STA             PTCD

                            TAP

EndAufgabe22 BRA EndAufgabe22
```

4. Tabellen

Stellvertretend für den Umgang mit Tabellen sollen Sie ein Programm erstellen, welches eine Quellentabelle bzw. -speicherbereich auf eine Zieltabelle bzw. -bereich kopiert und die Werte gleichzeitig um 3 erhöht.
Das Programm soll so ausgelegt sein, dass die Tabellenlänge und die Bereichsanfänge leicht variiert werden können (Infos stehen in Variablen), und dass eine Iteration mit Abfrage und Schleifenzähler verwendet wird.

Holen Sie die Daten an folgenden absoluten Stellen:

$0080: Bereichslänge (für dieses Beispiel z.B. =128d)
$0081: Schleifenzähler
$0082: Erste Quellenadresse (für dieses Beispiel z.B. =$0100)
$0084: Erste Zieladresse (für dieses Beispiel z.B. =$0280)

Für die Tabellenbereiche sind separate Sections zu definieren und das *.prm File entsprechend anzupassen.
Tipp: Erstellen Sie vorgängig ein Memory Map.

Den Programmablauf sollen Sie u.a. mit dem Monitor-TRACE-Command "**T**" beobachten.
Wie kann der zu kleine Offset-Bereich der indexierten Adressierung umgangen werden?

Erstellen Sie vorgängig zwingend ein Jackson Baumdiagramm! Realisation mittels einem Assembler-Template "HCS08_Assembler". Im Assemblerprogramm dürfen nur symbolische Namen (keine Zahlen) vorkommen.

Memory Mapping

```
$0080  TabLoopCounter
$0081  TabSrcPointer H
$0082  TabSrcPointer L     }→    Q1 + 3       $0100
                                 Q2 + 3       $0101        }→SrcTab (DS)
                                 Q3 + 3       $0102

$0083  TabDestPointer H
$0084  TabDestPointer L    }→    Q1 + 3       $02FF
                                 Q2 + 3       $0300        }→DestTab (DS)
                                 Q3 + 3       $0301
```

Main.asm

```
;--------------------------------------------------------------------
; 3.      STACK und VARIABLEN (Datenspeicher)
;--------------------------------------------------------------------
DATEN_STACK:              SECTION
;
TofStack:                 DS        StackSize-1 ;Stack in ASS in Source definiert!
                                    ;in C im *.prm File
BofStack:                 DS        1 ;Abschluss Stack

DATEN:                    SECTION
Variable1:                DS        1
Array1:                   DS        $20

DIRECT_PAGE_SECTION:      SECTION
TabLoopCounter:           DS          1
TabSrcPointer:            DS          2
TabDestPointer:           DS          2

SOURCE_SECTION:           SECTION
SrcTab:                   DS          50

DESC_SECTION:             SECTION
DestTab:                  DS          50
```

```
;-----------------------------------------------------------------------
; 2.EQUATIONS
;-----------------------------------------------------------------------
StackSize:          EQU          $60

;Konstanten-Werte (fuer IMM-Adressierung):
;-----------------------------------------
TabLength:          EQU          3


;***********************************************************************
;* Source Code                                                         *
;***********************************************************************

                    LDHX   #SrcTab
                    STHX   TabSrcPointer

                    LDHX   #DestTab
                    STHX   TabDestPointer

                    LDA    #TabLength
                    STA    TabLoopCounter

                    ; Tabelle abarbeiten...
CopyLoop:           BEQ    EndCopy

                    ; Byte laden
                    LDHX   TabSrcPointer
                    LDA    0,X
                    AIX    #1
                    STHX   TabSrcPointer

                    ; 3 addieren
                    ADD    #3

                    ; Byte speichern
                    LDHX   TabDestPointer
                    STA    0,X
                    AIX    #1
                    STHX   TabDestPointer

                    LDA    TabLoopCounter
                    DECA
                    STA    TabLoopCounter

                    BRA    CopyLoop

EndCopy:            NOP


EndLoop:            BRA    *
```

Project.prm

```
NAMES END /* Muss hier definiert sein.
             Kann fuer externe Objektdateien verwendet werden.*/

SEGMENTS
    Z_RAM                                =  READ_WRITE       0x0080 TO 0x00FF; //128 Bytes
    MY_RAM1                              =  READ_WRITE       0x0100 TO 0x0200;
    MY_RAM2                              =  READ_WRITE       0x02FF TO 0x03FF;
    MY_RAM3                              =  READ_WRITE       0x0400 TO 0x107F;
    MY_ROM                               =  READ_ONLY        0x182C TO 0xFFAF;
END

PLACEMENT
      DIRECT_PAGE_DATEN                    INTO Z_RAM;
      SOURCE_SECTION                       INTO MY_RAM1;
      DEST_SECTION                         INTO MY_RAM2;
       .data, Daten_STACK, DATEN           INTO MY_RAM3;
       .text, PROGRAMM, KONSTANTEN         INTO MY_ROM;
END

STACKSIZE 0x200                           // Stacksize 0x200 => 512 Bytes
```