Testprüfung in Mikrocomputertechnik 68HC11

Arbeitszeit: 90 min.
Hilfsmittel: beliebige eigene schriftliche
Das Aufgabenblatt bitte abgeben!

Name: _____

Studiengruppe: _____

Viel Erfolg !

Aufgabe	1.1	1.2	1.3	1.4	1.5	1.6	2.1	2.2	2.3	gesamt
Max. Punktezahl	4	2	2	2	2	5	17	45	11	90
Erreichte Punktezahl										

1. ***Allgemeine Fragen***
   → *Beantworten Sie die allgemeinen Fragen auf dem Aufgabenblatt*

1.1 Welchen Wert haben die Flags N, Z und C des Statusregisters (CC-Register) nach Ausführung der folgenden Befehle (0 oder 1 eintragen); Die Befehle werden nacheinander ausgeführt;
Carry ist anfangs 0 **(4)**

```
LDAB    #$81
CLRA
DECA
LSRB
```

N (Negativ)	Z (Zero)	C (Carry)

1.2 Was genau macht der Befehl **`TXS`** und welche **CPU-Register** (neben dem Register `PC`) werden verändert? **(2)**

1.3 Wie viele Mikrosekunden benötigt das nachfolgende Codefragment bis zum Erreichen von **`end_lp`**, wenn ein Systemzyklus 1 µs entspricht.

```
        LDAX    #4
lp      DEX
        BNE     lp

end_lp
```

(2)

1.4 Das **8-Bit-Kontrollregister** eines E/A-Bausteins befindet sich an der Adresse `$1800`. Es sollen nun in dem Bit 7 und Bit 6 zwingend gelöscht werden, die anderen Bits sollen ihre vorherigen Werte behalten.
Geben Sie nachfolgend für den 68HC11 das dafür notwendige Assembler-Codefragment an: **(2)**

1.5 Zwei E/A-Bausteine sollen eine Programmunterbrechung am 68HC11 auslösen können. Der Hardwareaufwand soll gering gehalten werden (d. h. keine Verwendung eines zusätzlichen Interruptcontrollers oder Multiplexers).

Skizzieren Sie die Beschaltung des IRQ-Eingangs der CPU mithilfe eines Blockschaltbilds. **(2)**

1.6 Gegeben ist folgendes 68HC11-Codefragment:

```
            ORG  $8000
name        'WT', 0

main        LDX  #name
            LDAB 0,X
```

a) Ergänzen Sie in der nachfolgenden Tabelle den Speicherauszug ab der Adresse $8000. Geben Sie dazu die Bytes als hexadezimale Zahl an: **(4)**

Adresse	$8000	$8001	$8002	$8003	$8004	$8005	$8006	$8007
Inhalt								

b) Wie lauten die Assemblerdirektiven, um sicherzustellen, dass nach dem Einschalten der Hardware der Programmablauf an der Marke **`main`** beginnt.
Achtung: Außer `name` und `main` stehen Ihnen keine symbolische Konstanten zur Verfügung **(1)**

_____

_____

2. *Programmierung*

→ Beantworten Sie diese Fragen auf dem karierten Bearbeitungsblatt

2.1 Entwerfen Sie ein Unterprogramm **LOWER**, das einen String, der mit dem Nullbyte abgeschlossen wurde, modifiziert in einen anderen Speicherbereich kopiert. Im Zielpuffer sollen alle Großbuchstaben des Strings in Kleinbuchstaben umgewandelt worden sein.

- Im Register X die Startadresse des zu kopierenden Strings
- Im Register Y die Startadresse des Zielpuffers

Der Rückgabewert befindet sich im Register B und gibt an wie viele Großbuchstaben in Kleinbuchstaben umgewandelt wurden. Der Zielstring ist ebenfalls mit dem Nullbyte abzuschließen.
Stellen Sie sicher, dass die Register X und Y am Ende des Unterprogramms wieder den übergebenen Werten entsprechen.

Aufrufbeispiel: Um einen String ab Adresse $8000 konvertiert in den Puffer ab Adresse $2000 abzulegen, wird im Hauptprogramm die folgende Befehlsfolge ausgeführt:

```
LDX    #$8000
LDY    #$2000
BSR    LOWER
```

Realisieren Sie das Unterprogramm **LOWER** in 68HC11-Assembler. **(17)**

Vorbemerkung: Die Vereinbarungen von Adressen (Code, Daten, I/O-Ports usw.) in den nun zu bearbeitenden Aufgabenstellungen entsprechen jenen des Praktikums. Sie können dieses als **gegeben** voraussetzen und müssen sie **nicht** im Code aufführen. Verwenden Sie jedoch aussagekräftige Namen für Variablen und Konstanten!

Adressen:

```
data          equ   $2000
stak          equ   $7FFF
prog          equ   $8000

porte         equ   $100A
pcdr          equ   $1003
pcdd          equ   $1007
pbdr          equ   $1004
```

2.2 Zur Temperaturüberwachung soll ein digitales Thermometer mithilfe eines 68HC11 entworfen werden. Am parallelen **Port E** ist ein Temperatursensor mit freilaufendem Analog-Digital-Wandler angeschlossen. Der Wandler liefert einen der Temperatur proportionalen 8-Bit-Wert. Der Temperaturbereich [0 °C .. 63,75 °C] wird auf den Wertebereich [0 .. 255] abgebildet. Die Auflösung beträgt 0,25 °C.

Die Temperatur soll in **bestimmten Zeitintervallen** vom Port gelesen werden und der **ganzzahlige Anteil** soll auf einer zweistelligen Siebensegmentanzeige als **Dezimalzahl** ausgegeben werden. Die Siebensegmentanzeige wird (wie im Praktikum) mithilfe zweier Decoder an **Port C** angeschlossen.
Port B wird später zur geeigneten Ausgabe der Viertelwerte verwendet.

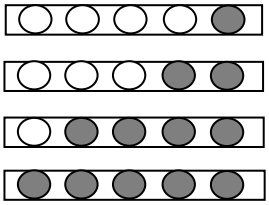
a) Schreiben Sie ein Unterprogramm **pInit** für die Initialisierung der verwendeten Ports.
   - Port E benötigt keine Initialisierung.
   - Die Richtung der Datensignale des Ports C sollen als Ausgang festgelegt werden.
   - Alle Datenbits der Ports B und C sollen auf 0 gesetzt werden. **(4)**

b) Für die Ausgabe eines zweistelligen Dezimalwertes ist das Unterprogramm **dezOut** zu schreiben, das den Wert auf **Port C** ausgibt.
   Der Dezimalwert liegt als sogenannte ungepackte BCD-Zahl vor, d. h. die Zehnerstelle befindet sich in der Byte-Variablen **zehner** und die Einerstelle in der Byte-Variablen **einer**. Es ist sichergestellt, dass die Variablenwerte zwischen [0 .. 9] liegen. **(11)**

c) Die Ausgabe der Viertelwerte erfolgt über 5 LEDs, die an **Port B** [Bit 4 .. Bit 0] angeschlossen sind.

Schreiben Sie dazu das Unterprogramm **`quartOut`**, das den übergebenen Wert im **Register A** geeignet umwandelt und auf den **Port B** ausgibt.
Im Register A sollen nur Bit 1 und Bit 0 interpretiert werden, stellen Sie aber sicher, dass der Wert in Register A am Ende des Unterprogramms wieder dem übergebenen Wert entspricht.
Verwenden Sie zur Umwandlung eine geeignete Wertetabelle.

Register A		Port B				
Bit 1	Bit 0	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	0	0	0	0	0	1
0	1	0	0	0	1	1
1	0	0	1	1	1	1
1	1	1	1	1	1	1



Realisieren Sie das Unterprogramm **`quartOut`** und geben Sie nachfolgend die Tabellendefinition an. **(16)**

d) Schreiben Sie das Hauptprogramm, das nach der Initialisierung (u. a. Aufruf von **`pInit`**) in einer Endlosschleife:
- Den aktuellen Sensorwert aus **Port E** liest,
- die Ausgabe der Viertelwerte mithilfe des Unterprogramms **`quartOut`** ausführt
- den Ganzzahlenanteil des Sensorwertes bildet,
  Hinweis: Teilen Sie den Wert durch 4
- das bereits realisierte Unterprogramm **`convert`** aufruft, um den binären Wert im Register A in eine ungepackte BCD-Zahl (in die Variablen **`zehner`** und **`einer`**) umzuwandeln,
- den Temperaturwert mithilfe des Unterprogramms **`dezOut`** ausgibt und das definierte Zeitintervall durch Aufruf des ebenfalls bereits realisierten Unterprogramms **`delay`** abwartet. **(14)**

2.3 Ein Mikrocontroller soll neben seinen anderen Aufgaben auch noch seine eigene Betriebszeit zählen. Dazu wird die entsprechenden Variable beim Hochlauf (Initialisierung) auf 0 gesetzt. Die Betriebszeit wird in Sekunden als 16-Bit-Wert im Speicher abgelegt.
Damit steht sie jederzeit für beliebige Zwecke zur Verfügung.

Ein vorhandener Sekundentakt löst Interrupts aus. Die Aufgabe der entsprechenden Interrupt-Serviceroutine besteht in der Aktualisierung der 16-Bit-Zeitvariablen. Nach einem halben Tag (43200 Sekunden) ist die Zeitvariable wieder auf 0 zu setzen. (Die Interruptquelle bzw. der Interrupt-Vektor sowie einen Quittierung des Interrupts spielen keine Rolle.)

Im Datenbereich ist die Zeitvariable wie folgt definiert:

```
time    ds.w        1
```

Schreiben Sie nur die Interrupt-Serviceroutine. **(11)**