

## Aufgabe 1: Technologien und Grundlagen

a) Gegeben sei folgende Wahrheitstabelle:

$a$	$b$	$c$	$d$	$f(a, b, c, d)$
0	0	0	0	1
0	0	0	1	1
0	0	1	0	0
0	0	1	1	1
0	1	0	0	1
0	1	0	1	1
0	1	1	0	0
0	1	1	1	1
1	0	0	0	1
1	0	0	1	0
1	0	1	0	1
1	0	1	1	1
1	1	0	0	0
1	1	0	1	0
1	1	1	0	1
1	1	1	1	1

Geben Sie  $f(a, b, c, d)$  in **konjunktiver kanonischer Normalform (KKN)** an.

Geben Sie nun **alle disjunktiven Minimalformen (DMF)** von  $f(a, b, c, d)$  an. Verwenden Sie für die Minimierung ein KV-Diagramm (ein Diagramm als Ersatz). Markieren Sie jeden zusammengefassten Term im KV-Diagramm und geben Sie letztlich die Minimalformen explizit unter Verwendung der Notation mit geschweiften Klammern an. Wie viele Minimalformen hat die Funktion?

	$a, b$			
	00	01	11	10
$c, d$	00			
	01			
	11			
	10			



	$a, b$			
	00	01	11	10
$c, d$	00			
	01			
	11			
	10			

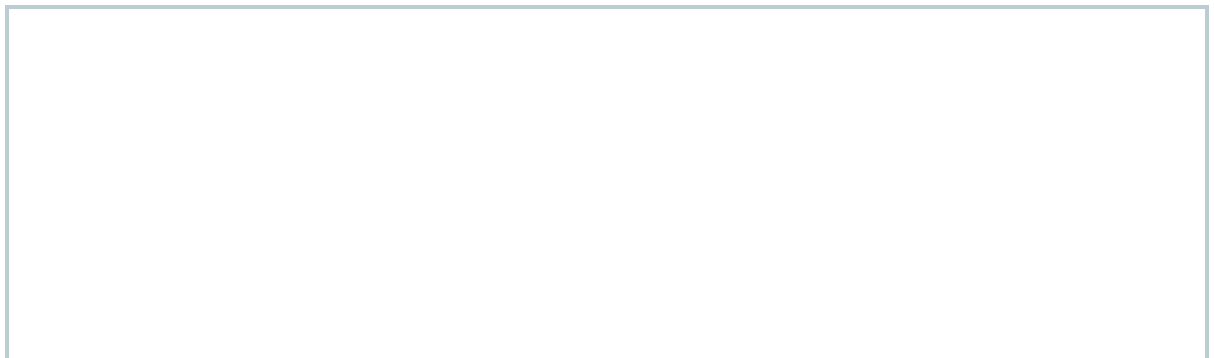
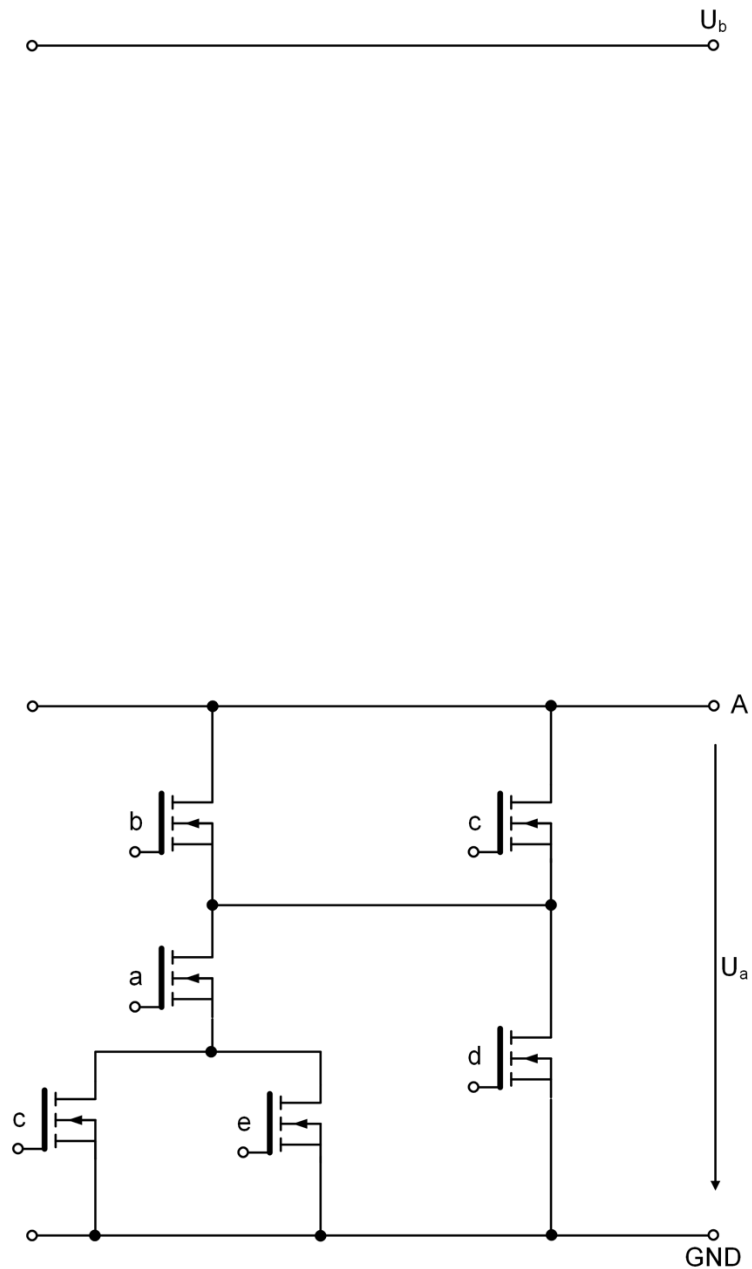
- b) Realisieren Sie basierend auf der Wahrheitstabelle  $f(a,b,c,d)$  mithilfe eines 4:1 MUX- Bausteins. Ferner stehen Ihnen zwei ODER-Gatter und beliebig viele Inverter zur Verfügung.

- c) Formen Sie  $h(a, b, c)$  so um, dass sich die Schaltfunktion ausschließlich aus NOR-Ausdrücken über jeweils zwei Termen zusammensetzt. Das Resultat darf keine Negationen in Form eines Negations-Strichs enthalten. Verwenden Sie für die finale Darstellung die Notation:  $(a \downarrow b)$ .

$$h(a, b, c) = ((\bar{a} \mid \bar{b}) + (c \downarrow b)) * \bar{a}$$

Zeichnen Sie den Schaltplan der Schaltfunktion  $h(a, b, c)$  unter Verwendung von Schaltsymbolen neuer DIN-Norm bestehend aus NOR-Gattern mit zwei Eingängen.

- d) Vervollständigen Sie die nachfolgend gegebene Grundsaltung um die Komplementärschaltung und geben Sie die durch diese Schaltung am Ausgang A realisierte Funktion  $g(a, b, c, d, e)$  explizit an. Verwenden Sie folgende Symbole für n-Kanal-  und p-Kanal-CMOS-Transistoren .



- e) Geben Sie eine VHDL-Beschreibung (Entity und Architecture) der Funktion  $h(a,b,c,d)$  an.

$$h(a,b,c,d) = (a + b) * (b + c) * d$$

$\Sigma_{A1} =$  \_\_\_\_\_ Punkte

## Aufgabe 2: Operations- und Steuerwerk

In dieser Aufgabe sollen ein Operations- und ein zugehöriges Steuerwerk auf Basis eines Schieberegisters entworfen werden, um den nachfolgenden, durch einen RT-Code gegebenen Algorithmus zu realisieren. Die Operationen aus Takt 4 und Takt 7 definieren die in a) zu realisierende Verdrahtung der ALU.

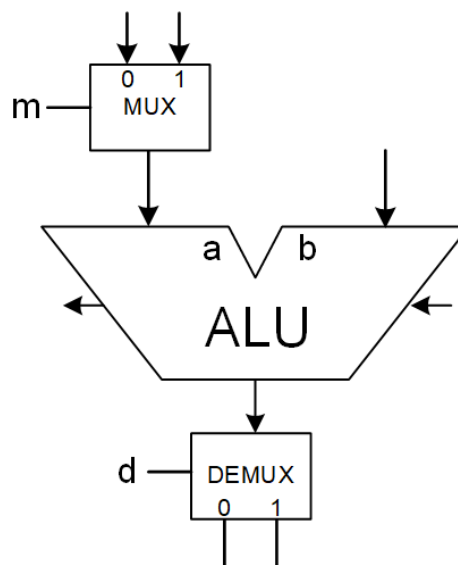
RT-Code	Takt	Kontrollsignale
declare register C, A(15:0), B(15:0), D(15:0), CNT(1:0)		
declare bus BUS(15:0)		
INIT:		
CNT <- 0, D <- 0;	#Takt0	_____
A <- BUS;	#Takt1	_____
LOOP:		
B <- 0;	#Takt2	_____
LOOP2:		
B(15:1) <- B(14:0), B(0) <- 0, CNT <- CNT + 1;	#Takt3	_____
<b>C &lt;- A(15), A(15:1) &lt;- A(14:0),     A(0) &lt;- 0;</b>	#Takt4	_____
if(C = 1) then B <- B + 1 fi;	#Takt5	_____
if(CNT <> 0) then goto LOOP2 fi;	#Takt6	_____
<b>D &lt;- D + B;</b>	#Takt7	_____
if (A = 0) then goto END else goto LOOP fi;	#Takt8	_____
END: goto END;	#Takt9	_____

- a) Ergänzen Sie das Blockschaltbild des Operationswerks so, dass es die nachfolgend und durch den RT-Code beschriebenen Komponenten und Funktionalitäten aufweist. Das Operationswerk besitzt die drei 16 Bit breiten Register A, B und D, sowie ein 2 Bit Zählregister CNT, welches mit dem Kontrollsignal (KS) *incCNT* inkrementiert und mit *clrCNT* auf Null gesetzt werden kann. Zudem gibt das Register mithilfe des Flags *zCNT* an, ob sein Inhalt gerade Null (*zCNT* = 1) ist oder nicht. Das Register D kann mithilfe des KS *clrD* auf Null zurückgesetzt werden. Das Register B kann über das KS *clrB* auf Null gesetzt werden, wohingegen *incB* den Inhalt inkrementiert und *lsB* den Inhalt um eine Stelle nach links shiftet und dabei eine Null nachzieht. Das Register A stellt das Kriterium *zA* bereit, welches angibt, ob der Inhalt von A Null ist (*zA* = 1) oder nicht. Über das KS *inA* soll der Inhalt des 16 Bit breiten Busses *BUS* in das Register A geladen werden können.

**Das Register A ist kein Schieberegister!**

Das KS *add* sorgt dafür, dass die ALU die beiden an den Eingängen a und b anliegenden Werte addiert, wohingegen das KS *sla* dafür sorgt, dass der am Eingang a anliegende Wert um ein Bit nach links unter Nachziehen einer Null geschiftet wird. Zudem stellt die ALU das Carry-Flag C bereit, welches angibt, ob bei der letzten Operation ein Überlauf aufgetreten ist oder nicht.

BUS



- b) Ergänzen Sie die obige Tabelle mit den zu den einzelnen RT-Operationen korrespondierenden Steuersignalen.
- c) Welchen Wert enthält das Register A am Ende des Algorithmus? Geben Sie explizit den Inhalt als Dezimalzahl an.

- d) Welchen Wert enthält das Register B am Ende des Algorithmus? Dieser Wert kann nur beschreibend angegeben werden.

- e) Welche Berechnung führt dieser Algorithmus durch? Geben Sie die semantische Bedeutung in einem Satz an. Hinweis: Das Register A erhält vom BUS einen 4-elementigen Vektor, wobei jedes Element einer positive vorzeichenlose 4 Bit Zahl entspricht. Element 0 steht in den unteren vier Bit von A, Element 1 in den nächsthöheren vier Bit bis hin zu Element 3 in den oberen vier Bit.

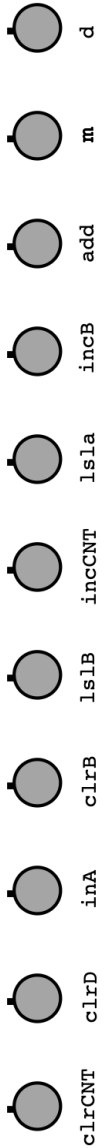
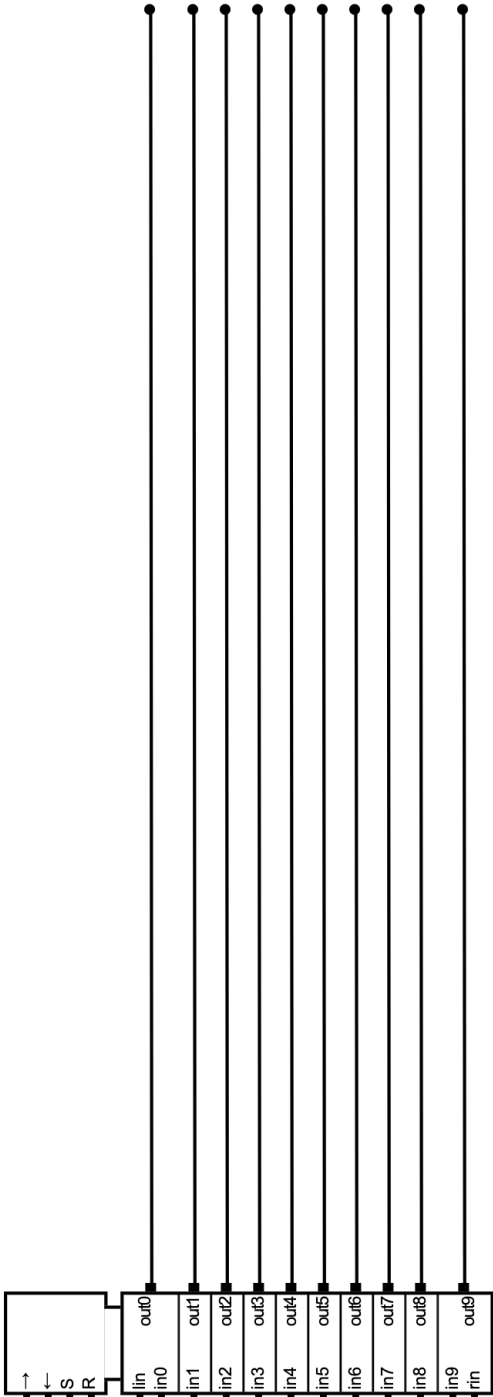
- f) Welche Berechnung würde der Algorithmus durchführen, wenn der Counter CNT eine Breite von 4 Bit bekommen würde? Geben Sie beschreibend den Inhalt von B und D nach Ablauf an.

- g) Entwerfen Sie ein Steuerwerk auf Basis eines Schieberegister, welches den Algorithmus auf dem Operationswerk realisiert, indem Sie den nachfolgenden Entwurf vervollständigen. Bedenken Sie, dass nicht alle Kriterien aus dem RT-Code eins zu eins vom Operationswerk bereitgestellt werden. Positionieren sie die Buttons für die Kriterien bei Bedarf nach Belieben.



Takt

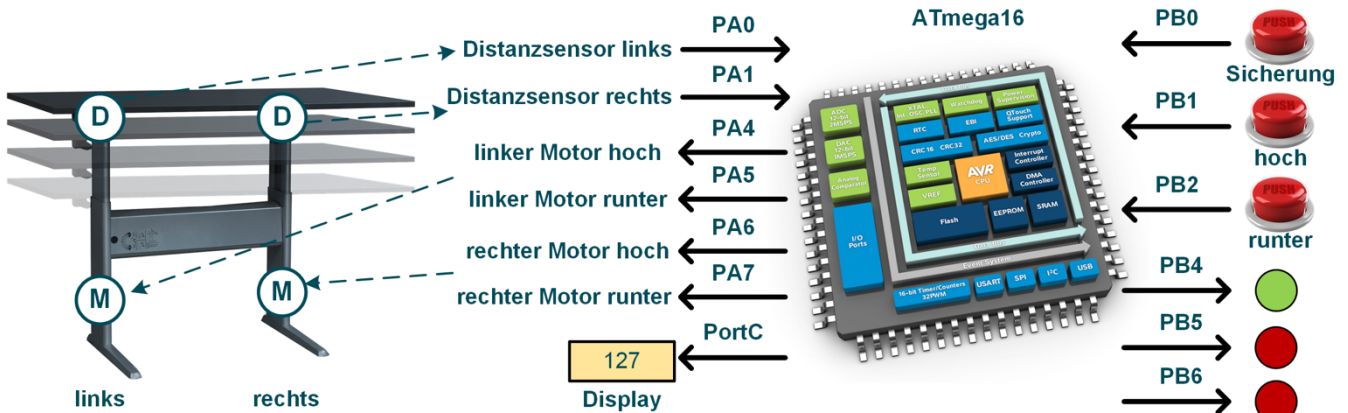
Reset



$\sum_{A2}$  = \_\_\_\_\_ Punkte

## Aufgabe 3: Assemblerprogrammierung

### Höhenverstellbarer Tisch



### Abbildung 1: Beschaltung des ATmega16

Nach einer sehr langen Home-Office-Phase sind die Rücken der Mitarbeiter des Instituts für Total Großartige Informatik (TGI) durch ihre fehlerhafte Haltung beim Sitzen am heimischen Schreibtisch ziemlich in Mitteldenschafter gezogen worden. Nun wurde von der Institutsleitung bestimmt, für jeden Mitarbeiter elektrisch höhenverstellbare Schreibtische zu beschaffen. Da auf Grund eines Softwarefehlers die Höhenverstellung des Schreibtisches nicht korrekt funktioniert und er automatisch nach einiger Zeit auf die Maximalhöhe ausfährt, soll im Rahmen eines Studierendenprojekts die auf einem ATmega16 basierende Steuerung umprogrammiert werden.

Für die Steuerung des Tisches ist an jeder Seite (links und rechts) ein Sensor zur Höhenmessung verbaut, der die aktuelle Höhe des Tisches als analoge Spannung an den Pins PA0 (links) und PA1 (rechts) bereitstellt. Der ADC gemessene Wert 0 entspräche dabei einer Höhe von 0 cm (nicht erreichbar, minimale Höhe sind 60cm). Mit einer 8 Bit Auflösung bei der AD-Wandlung kann die Höhe mit einer Genauigkeit von 0,5cm gemessen werden. Die aktuelle Höhe des Tisches gemessen auf der rechten Seite wird auf ein an PORTC angeschlossenes Display ausgegeben.

Die Motoren des Tisches (links und rechts) sind jeweils mit zwei Ausgängen an den ATmega angeschlossen und können initiiert durch die Ausgabe einer 1 an den entsprechenden Pins den Tisch nach oben (linker Motor PA4 und rechter Motor PA6) oder nach unten (linker Motor PA5 und rechter Motor PA7) fahren.

Als Eingabe für den Nutzer sind drei Taster vorhanden, die zum Einstellen der Tischhöhe genutzt werden. Der Taster an PB0 bildet eine Sicherung und muss immer gedrückt sein (PB0 = 1), solange der Tische verstellt werden soll. Dies Taster an PB1 (hoch) und PB2 (runter) dienen zur Höhenverstellung, wobei auch hier jeweils eine 1 am Pin einen gedrückten Taster indiziert.

Drei LEDs geben den Status des Systems an. Die grüne Systemanzeige-LED an PB4 leuchtet solange das System eingeschaltet ist. Die rote LED an PB5 wird eingeschaltet, sobald eine unterschiedliche Höhe an beiden Seiten des Tisches erkannt wird. Die rote LED an PB6 zeigt an, dass sich der Tisch allein verstellt hat, falls zum Beispiel die Motoren nachgeben.

Das Verhalten kann wie folgt skizziert werden. Ist das System eingeschaltet, so kann die Höhe des Tisches durch Betätigung der Sicherung in Kombination mit den Tastern für hoch oder runter verstellt werden. Die aktuell eingestellte Höhe wird gespeichert und auf dem Display dargestellt. Viermal in der Minute durchläuft das System eine Checkroutine, die überprüft, ob die gemessene Höhe noch der eingestellten Höhe entspricht und ob beide Sensoren dieselbe Höhe bestimmen. Sollte eines der beiden Dinge nicht stimmen, so wird der entsprechende Fehler durch eine LED angezeigt und das System schaltet die Motoren ab. Erst wenn die Fehler durch manuelles Eingreifen behoben wurden, kann der Tisch wieder verstellt werden.

## Aufgaben:

Die Bearbeitung der Aufgabe erfolgt in mehreren Unteraufgaben. Nutzen Sie die einzeln vorgegebenen Bereiche und schreiben Sie aussagekräftigen Assembler Code beziehungsweise kommentieren Sie gegebenenfalls den Code sinnvoll. Die Teilaufgaben sind sequentiell gestellt und sollen zusammen einen aneinanderkopierbaren Code ergeben. **Bedenken Sie in diesem Zusammenhang den eventuellen Wechsel zwischen Code- und Datensegment.**

- a) Was ist die maximal einstellbare Höhe des Tisches? Dabei kann angenommen werden, dass die Motoren den Tisch nicht höher fahren können, als die maximale mit den Distanzsensoren messbare Höhe. Begründen Sie Ihre Antwort.

- b) Sorgen Sie dafür, dass Ihnen die ATmega16-spezifischen Namensdefinitionen zur Verfügung stehen. Beginnen Sie anschließend die Programmierung mit den zusätzlichen Registernamensdefinitionen. Sorgen Sie dafür, dass das Register **R17** unter dem Namen **tmp1** und das Register **R18** unter dem Namen **tmp2** angesprochen werden können. Weitere von Ihnen eventuell genutzte Namen für einzelne Register müssen hier ebenfalls definiert werden.

- c) Reservieren Sie im RAM an der kleinstmöglichen Adresse unter dem Label **height** ein Byte, welches später die angefahrene Höhe des Tisches enthalten soll.

- d) Initialisieren Sie die Interrupt-Vektor-Tabelle, sodass nach einem **RESET** zum Label **INIT** und nach einem **Timer/Counter 1 B Compare Match Interrupt** zur **ISR\_CHECK** gesprungen wird. Sorgen Sie in diesem Zusammenhang dafür, dass Ihr weiterer Quellcode erst nach der Interrupt-Vektor-Tabelle abgelegt wird.

- e) Konfigurieren Sie unter dem Label **INIT** die Ein- und Ausgabeports und initialisieren Sie den Stackpointer. Beachten Sie eventuelle Initialwerte für die Ausgänge. Sie dürfen **nicht** davon ausgehen, dass die Register standardmäßig mit dem Wert Null initialisiert sind. Konfigurieren Sie sämtliche Eingänge für den Betrieb im Tri-State Modus, schalten Sie die Motoren ab und lassen Sie lediglich die grüne Systemanzeige-LED leuchten.

- f) Konfigurieren Sie den AD-Wandler, indem Sie ihn aktivieren, eine linksbündige Speicherung einstellen und den Prescaler auf 128 konfigurieren. Die Versorgungsspannung soll als Referenzspannung genutzt werden. Rufen Sie im Anschluss das später zu realisierende **read\_adc** Unterprogramm auf, welches im Register R16 den zu verwendenden Kanal erwartet und das 8 Bit Ergebnis der AD-Wandlung im selbigen Register zurückgibt. In diesem Fall soll die initiale Höhe mit Hilfe des rechten Sensors bestimmt und anschließend im RAM unter **height** abgelegt werden.

- g) Der Tisch verfügt über eine automatische Prüfroutine (**ISR\_CHECK**), die viermal in der Minute durchlaufen wird. Hierfür soll der **Timer/Counter 1B** in Verbindung mit seinem **Output-Compare-Match-Interrupt** genutzt werden. Es ist ein Prescaler von 256 zu verwenden. Konfigurieren Sie den Timer entsprechend. Die Angabe des Vergleichswertes genügt als Gleichung mit den konkreten Werten und muss nicht explizit ausgerechnet werden. Sie können das Zurücksetzen des Zählers hier konfigurieren oder später in der **ISR\_CHECK** realisieren. Beachten Sie dabei, dass **Timer/Counter 1B** genutzt wird. Aktivieren Sie die genutzten Interrupts lokal und global.

- h) Nachfolgend ist das Unterprogramm **READ\_ADC** teilweise gegeben. Ergänzen Sie den Code so, dass der in **R16** übergebene Kanal eingestellt wird, bevor die A/D-Wandlung beginnt und die oberen 8 Bit des Ergebnisses in **R16** zurückgegeben werden.

```
READ_ADC:
; TODO Einstellen des Kanals

sbi ADCSRA, ADSC
READ_ADC_LOOP:
sbic ADCSRA, ADSC
rjmp READ_ADC_LOOP

; TODO Ergebnis einlesen ...

ret
```

Was müsste geändert werden, wenn beim Polling als Indikator für die Fertigstellung der AD-Wandlung das Interrupt-Flag genutzt werden soll?

- i) Implementieren Sie die **ISR\_CHECK** Interrupt Service Routine, die das System auf zwei Fehlerzustände überprüft. Zu Beginn werden beide Distanz-/Höhenwerte des Tisches eingelesen und es erfolgt ein Vergleich. Sollten die Werte unterschiedlich sein, so ist ein Fehler aufgetreten und die rote LED an **PB5** soll eingeschaltet werden. Ist kein Fehler aufgetreten, so ist die LED auszuschalten. Im Anschluss erfolgt ein Vergleich des rechten Höhenwertes mit der unter **height** im RAM abgelegten, eingestellten Höhe. Sollten sich beide Werte unterscheiden, so ist der aufgetretene Fehler durch Einschalten der roten LED an **PB6** anzuzeigen, im anderen Fall ist die LED auszuschalten. Vermeiden Sie sämtliche Seiteneffekte.



- j) Realisieren Sie nun unter dem Label **MAIN** das Hauptverhalten des Systems. Das in einer **Endlosschleife** arbeitende Verhalten sei nachfolgend schrittweise skizziert
- Es erfolgt das Einlesen der Höhe des Tisches anhand des rechten Sensors und die Ausgabe in cm auf **PortC**
  - Anschließend erfolgt die Prüfung, ob ein Fehler aufgetreten ist oder nicht (PB5 & PB6). Sollte ein Fehler aufgetreten sein, so werden alle Motoren ausgeschaltet und wieder zu **MAIN** gesprungen, ansonsten wird der Programmablauf fortgesetzt.
  - Um die Motoren anzusteuern ist es erforderlich, dass die Sicherung (PB0) und der Taster für die entsprechende Richtung (hoch PB1, runter PB2) betätigt werden.
    - Prüfen Sie, ob die Sicherung gedrückt ist oder nicht, wenn ja geht der Programmablauf weiter, anderenfalls wird zu **MAIN** gesprungen.
    - Deaktivieren Sie die Interrupts global, da es während des Verstellens des Tisches zu kleinen Abweichungen in der Höhe kommen kann und das nicht als Fehler erkannt werden soll.
    - Prüfen Sie ob und falls ja, welcher Taster zum Verstellen des Tisches betätigt wird. Solange dieser gedrückt ist, sollen die entsprechenden Motoren angesteuert und der Tisch dadurch verstellt werden. (Beide Motoren sind dabei identisch zu steuern)
    - Anschließend werden alle Motoren deaktiviert und die aktuell eingestellte Höhe anhand des rechten Sensorwertes bestimmt und im **RAM** unter **height** hinterlegt.
    - Aktivieren Sie die Interrupts global und springen Sie zu **MAIN**.

Matrikelnummer: \_\_\_\_\_

Studiengang: \_\_\_\_\_

- k) Gegeben sei der nachfolgende Codeausschnitt zum Ablegen von Daten im ROM. Bitte geben Sie den resultierenden Inhalt des ROMs in Hexadezimaldarstellung an. Die Adressierung sei dabei mit Hilfe der Wortadressen vorzunehmen.

```
.cseg  
.org $101  
DATA: .db 12, 255, 32, 16, 5, 5
```

Adresse	Inhalt in Hex

Geben Sie eine Folge von Assemblerbefehlen für den ATmega an, die in einer Schleife die Daten aus dem ROM in den RAM unter dem Label **ARRAY** ablegen. Der entsprechende Speicherplatz ist bereits reserviert. **Es darf dabei kein Register separat als Zähler genutzt werden!**

$\Sigma_{A3} =$  \_\_\_\_\_ Punkte