

Aufgabe 1: Technologien und Grundlagen

a) Gegeben sei folgende Wahrheitstabelle:

<i>a</i>	<i>b</i>	<i>c</i>	<i>d</i>	<i>f(a, b, c, d)</i>
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	0
0	1	0	0	0
0	1	0	1	0
0	1	1	0	1
0	1	1	1	1
1	0	0	0	0
1	0	0	1	1
1	0	1	0	0
1	0	1	1	1
1	1	0	0	1
1	1	0	1	1
1	1	1	0	1
1	1	1	1	1

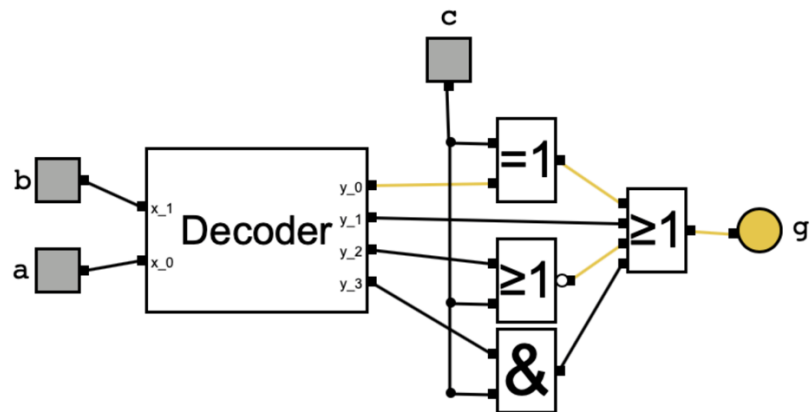
Geben Sie $f(a, b, c, d)$ in einer **konjunktiven Normalform (KNF)** an.

Ist die von Ihnen angegebene Funktion ebenfalls eine **konjunktive Minimalform (KMF)**? Begründen Sie Ihre Antwort.

- b) Geben Sie eine VHDL-Beschreibung (Entity und Architecture) der Funktion $f_{\text{DMF}}(a, b, c, d)$ an.

$$f_{\text{DMF}}(a, b, c, d) = ab + ad + bc$$



- c) Geben Sie die durch die nachfolgende Schaltung realisierte Funktion an. Ihnen stehen als Verknüpfungen UND und ODER zur Verfügung. Die Negation einzelner Variablen ist erlaubt.

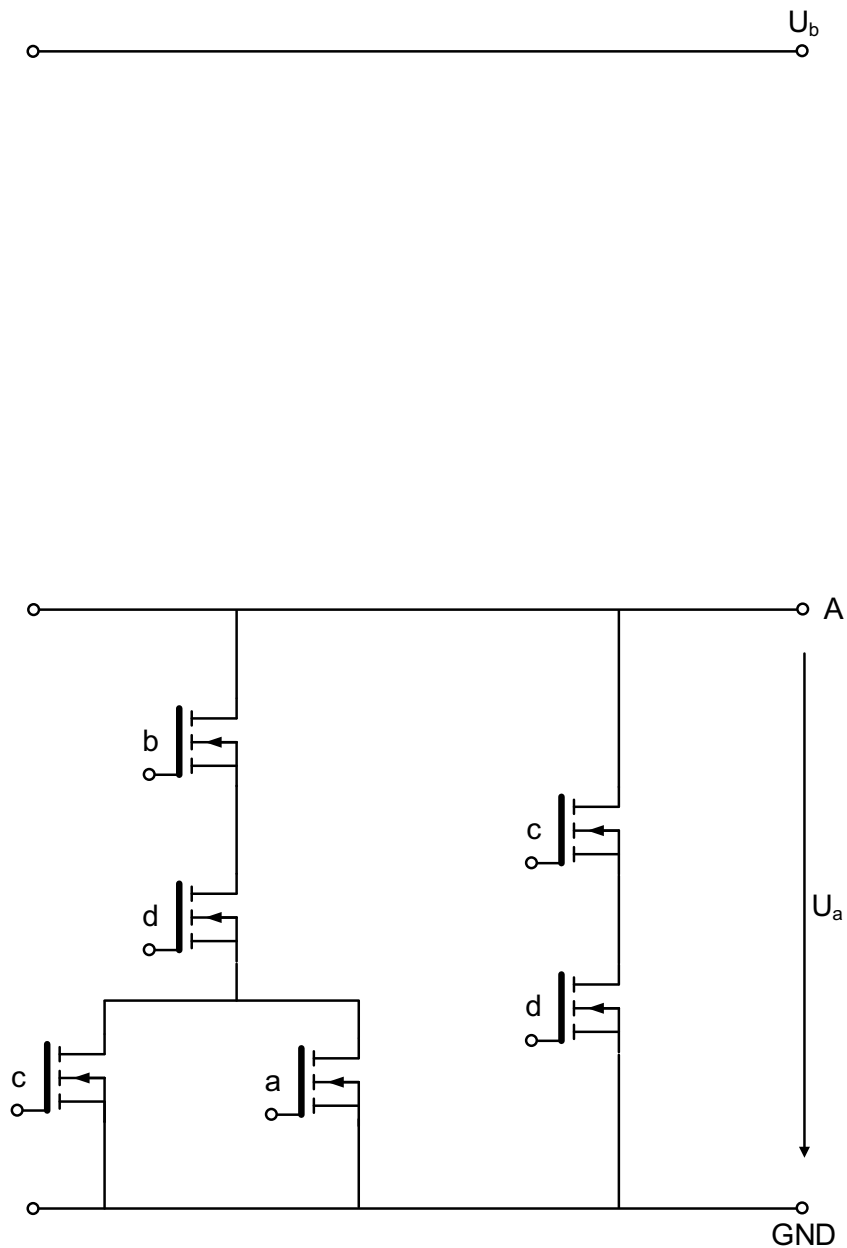


- d) Formen Sie $h(a, b, c)$ so um, dass sich die Schaltfunktion ausschließlich aus NOR-Ausdrücken über jeweils zwei Termen zusammensetzt. Das Resultat darf keine Negationen in Form eines Negationsstrichs enthalten. Verwenden Sie für die finale Darstellung die Notation: $(a \downarrow b)$.

$$h(a, b, c) = c \mid (a + b)$$

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

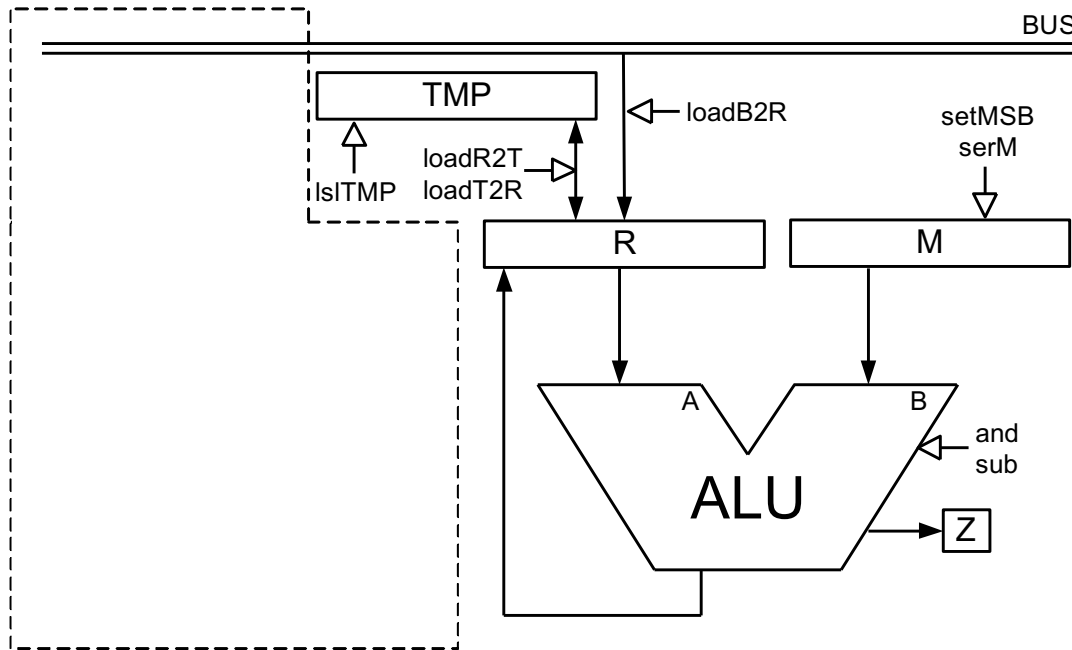
- e) Vervollständigen Sie die nachfolgend gegebene Schaltung um die Komplementärschaltung und geben Sie die durch diese Schaltung am Ausgang A realisierte Funktion $f(a, b, c, d)$ explizit an. Verwenden Sie folgende Symbole für n-Kanal-  und p-Kanal-CMOS-Transistoren .



$\Sigma_{A1} = \underline{\hspace{2cm}}$ Punkte

Aufgabe 2: Operations- und Steuerwerk

In dieser Aufgabe sollen ein Operations- und ein zugehöriges Steuerwerk in Form einer Zählersteuerung entworfen werden, um den Nachfolgenden durch einen lückenhaften Pseudocode und eine Abfolge von Steuersignalen gegeben Algorithmus zu realisieren.



Gegeben sei das obige Blockschaltbild eines Operationswerks, welches die nachfolgend beschriebenen Komponenten und Funktionalitäten aufweist. Sämtliche Register, der Bus sowie die Ein- und Ausgänge der ALU sind 16 Bit breit. Das Steuersignal *loadB2R* erlaubt es, den auf dem *BUS* anliegenden Wert in das Register *R* zu laden. Über die Steuersignale *loadR2T* und *loadT2R* ist es möglich, den Wert aus dem Register *R* in das Register *TMP* zu kopieren (*loadR2T*) und umgekehrt (*loadT2R*). Ferner ermöglicht das Steuersignal *lsTMP* den logischen Linksshift des Inhalts vom Register *TMP* mit dem Nachziehen einer Eins. Das Register *M* verfügt über das Steuersignal *setMSB* zum Setzen des MSB des Registers (8000_{16} bzw. 32768_{10}) und Löschen der restlichen Bits des Registers. Ferner erlaubt das Signal *serM* das Setzen des Registerwertes auf $FFFF_{16}$ (65535_{10}). Für die Berechnung besitzt das Operationswerk eine *ALU* mit den zwei Eingängen *A* und *B*. Der Eingang *A* ist mit dem Register *R*, welches gleichzeitig das Ergebnisregister der *ALU* darstellt, verbunden, wohingegen das Register *M* als Eingangsregister *B* dient. Mit Hilfe des Steuersignals *sub* kann die *ALU* die Subtraktion $A - B$ ausführen. Das Steuersignal *and* ermöglicht das Verunden der Eingangswerte *A* und *B*. Als Kriterium stellt die *ALU* das *Z*-Flag zur Verfügung, welches den Wert Eins annimmt, sobald das Ergebnis der letzten Operation Null war, anderenfalls hat es den Wert Null.

- Erweitern Sie das Operationswerk im gekennzeichneten Bereich (gestrichelte Linie) um das Register *CNT*, dessen Wert über das Steuersignal *loadB2CNT* vom Bus eingelesen und mit *decCNT* dekrementiert werden kann. Das Steuersignal *loadC2I* soll es erlauben, den Wert von *CNT* in das Register *IDX* zu kopieren, welches Sie ebenfalls ergänzen müssen.

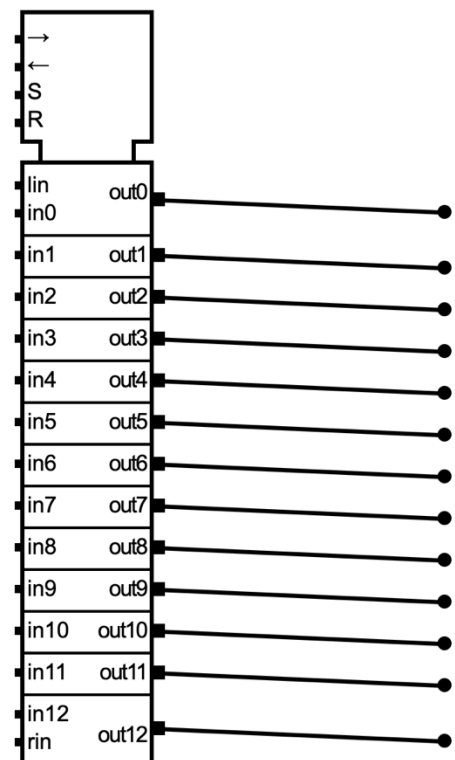
- b) Ergänzen Sie neben dem nachfolgend gegebenen Pseudocode die zu den RT-Operationen korrespondierenden Steuersignale.

RT-Code	Takt	Steuersignale
declare bus BUS(15:0)		
declare register R(15:0), CNT(15:0), M(15:0), TMP(15:0), IDX(15:0)		
INIT:		
R <- BUS;	#Takt0	_____
TMP <- R, CNT <- BUS;	#Takt1	_____
SAVE:		
IDX <- CNT;	#Takt2	_____
TEST:		
M <- 65535, R <- TMP;	#Takt3	_____
R <- R - M;	#Takt4	_____
if (Z = 1) then		
goto END		
fi;	#Takt5	_____
R <- TMP, M <- 32768;	#Takt6	_____
R <- R and M, CNT <- CNT - 1;	#Takt7	_____
TMP(15:1) <- TMP(14:0),		
TMP(0) <- 1,		
if (Z = 0) then		
goto CHECK		
else		
goto TEST		
fi;	#Takt8	_____
CHECK:		
R <- TMP;	#Takt9	_____
R <- R and M;	#Takt10	_____
if (Z = 1) then		
TMP(15:1) <- TMP(14:0),		
TMP(0) <- 1, CNT <- CNT - 1,		
goto SAVE		
else		
goto TEST		
fi;	#Takt11	_____
END:		
goto END;	#Takt12	_____

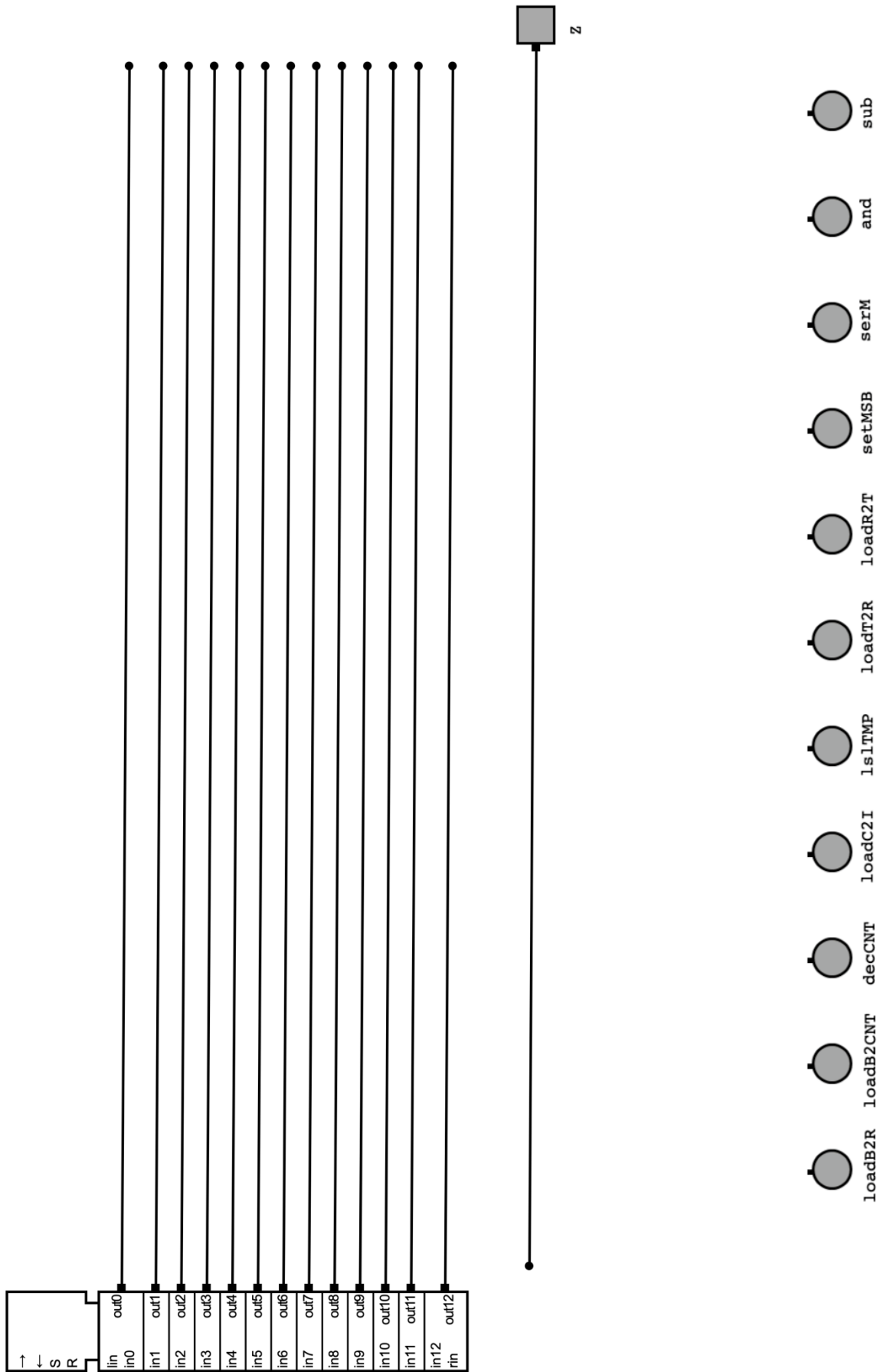
- c) Welchen Wert enthält das Register TMP am Ende des Algorithmus?

- d) Welche Berechnung führt dieser Algorithmus durch, wenn in R ein beliebiger Wert und in CNT zu Beginn der Wert 16 hineingeladen wird? Geben Sie die semantische Bedeutung in einem Satz an.

- e) Entwerfen Sie ein Steuerwerk auf Basis eines Schieberegisters, welches den Algorithmus auf dem Operationswerk realisiert, indem Sie den Entwurf auf der nächsten Seite vervollständigen. Die Sprünge müssen hier nicht realisiert werden.
- f) Ergänzen Sie die nachfolgende Schaltung so, dass sie das zeitliche Verhalten des Steuerwerks inklusive der Sprünge aus den Takten fünf, acht und zwölf realisiert. Änderungen des Inhalts des Registers erfolgen **nur** bei einem positiven Taktsignal (Schalter). Ferner soll es möglich sein, dass das Signal START in Kombination mit dem Taktsignal zur Initialisierung des Registers mit dem Wert 1 führt.



Z



$\Sigma_{A2} = \underline{\hspace{2cm}}$ Punkte

Aufgabe 3: Assemblerprogrammierung

Sicherheitstür

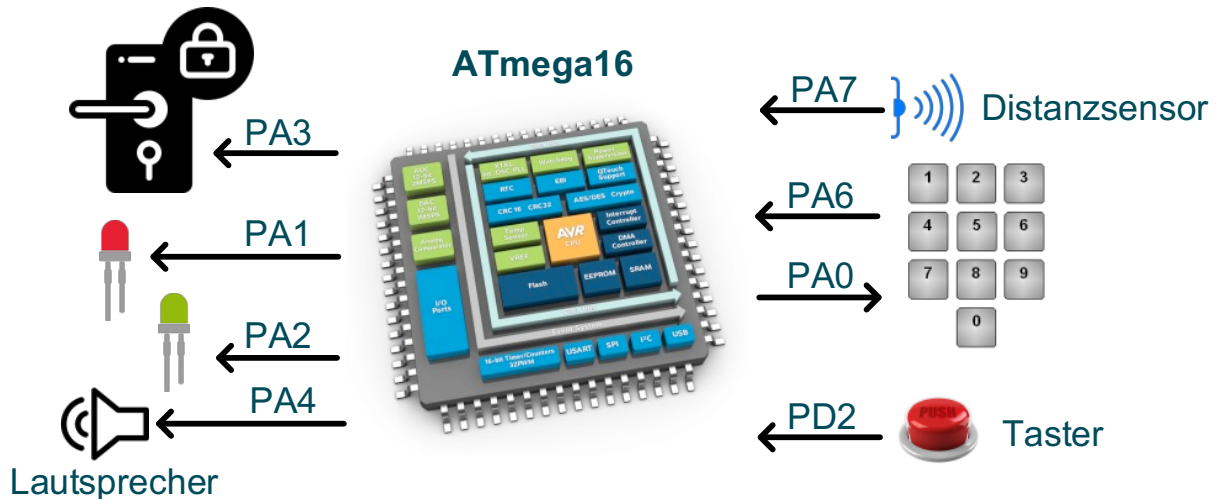


Abbildung 1: Beschaltung des ATmega16 für die Sicherheitstür

Ziel dieser Aufgabe ist die Implementierung einer Sicherheitstür mit Zugangskontrolle in Assembler auf dem aus der Übung bekannten Mikrokontroller ATmega16. Dieser ist entsprechend Abbildung 1 mit einem Distanzsensormodul, einem Tastenfeld, mit LEDs (rot PA1 und grün PA2), einem Taster sowie dem Schließmechanismus der Tür verbunden.

Das Verhalten des Systems kann wie folgt skizziert werden. Die Tür befindet sich im Bereitschaftsmodus, in dem das Tastenfeld deaktiviert, die Tür verschlossen und die grüne LED ausgeschaltet ist. Die rote LED leuchtet und mit Hilfe des Distanzsensors wird auf die Annäherung einer Person an das Tastenfeld der Tür gewartet.

Befindet sich jemand vor der Tür, so wird das System aktiviert, indem das Tastenfeld eingeschaltet wird (PA0 = 1) und es kann mit der Eingabe des achtstelligen Pin-Codes begonnen werden. Nach der Eingabe erfolgt der Vergleich mit dem Code der Tür. Stimmen beide nicht überein, beginnt der Ablauf von vorn. Ist der eingegebene Code korrekt, so wird die grüne LED ein- und die rote LED ausgeschaltet. Im Anschluss muss der Taster gedrückt werden, bevor für fünf Sekunden das Schloss entriegelt (PA3 = 1) wird und für diesen Zeitraum ein Piep-Ton (PA4 = 1) erklingt. Im Anschluss beginnt das System von vorn.

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. Die Teilaufgaben werden unabhängig voneinander bearbeitet und müssen nicht aneinanderkopierbar sein. Es kann unter anderem vorkommen, dass in einer Teilaufgabe Direktiven genutzt werden, die sich auf den RAM oder den ROM beziehen, was jeweils erkennbar sein muss.

- a) Ergänzen Sie den nachfolgenden Assemblercode so, dass das Ablegen des Codes (CODE) und der Tabelle mit den Schwellenwerten der einzelnen Tasten (KEYTABLE) direkt im Anschluss an die Interrupt Vektor Tabelle erfolgt. Der CODE besteht aus vier Bytes, die jeweils zwei Ziffern des Codes (oberes und unteres Nibble) enthalten. Das Tastenfeld generiert ein analoges Signal am Eingang PA0. Die A/D-Wandlung liefert je nach gedrückter Taste einen Wert w innerhalb der angegebenen Grenzen aus der Tabelle KEYTABLE und es gilt z.B. für die Taste 1: $36 \leq w \leq 44$.

; Ablegen direkt nach der Interrupt Vektor Tabelle

CODE:

```
.DB 0x22, 0x11
.DB 0x20, 0x19
```

KEYTABLE:

```
.DB 28, 33 ; '0'
.DB 36, 44 ; '1'
.DB 49, 58 ; '2'
.DB 68, 80 ; '3'
.DB 83, 96 ; '4'
.DB 105, 118 ; '5'
.DB 121, 134 ; '6'
.DB 139, 152 ; '7'
.DB 161, 174 ; '8'
.DB 202, 211 ; '9'
.DB 255, 0 ; ENDE-KeineTaste
```

In welchem Speicher werden die Daten durch den vorherigen Assemblercode abgelegt?

Wie lautet der hinterlegte Code des Systems und an welcher Speicheradresse ist er abgelegt?

- b) Beginnen Sie die Programmierung mit den einzelnen Registernamensdefinitionen. Sorgen Sie dafür, dass das Register **R16** unter dem Namen **Tmp**, das Register **R17** unter dem Namen **qSec** und das Register **R18** unter dem Namen **Open** angesprochen werden können. Definieren Sie ferner die Konstante **maxDist** mit dem Wert 150, die die maximale Distanz des Benutzers vom System angibt.

- c) Initialisieren Sie die Interrupt-Vektor-Tabelle, sodass nach einem **RESET** zum Label **INIT**, nach dem **External Interrupt Request 0** zur **ISR_INT0** und nach einem **Timer/Counter 0 Overflow Interrupt** zur **ISR_TIMER0** gesprungen wird.

- d) Für das Zwischenspeichern der Eingabe werden acht Bytes im RAM ab Adresse \$100 benötigt, die unter dem Label **INPUT** zugreifbar sein sollen.

Unter welcher Adresse (explizit angeben) wird der letzte Wert der Eingabe abgelegt?

- 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. Beachten Sie zudem, wodurch ein abgeschaltetes System von außen erkennbar ist.

- f) Konfigurieren Sie den **externen** Interrupt **INT0** so, dass er bei einer durch den an **PD2** angeschlossenen Taster hervorgerufenen **fallenden Flanke** ausgelöst wird.

- g) Die Messung der Zeitdauer erfolgt mit Hilfe des **Timer/Counter 0**. Konfigurieren Sie ihn mit einem **Prescaler** von **1024**. Sorgen Sie weiterhin dafür, dass beim Überlauf des Zählregisters ein Interrupt ausgelöst wird. Aktivieren Sie den Interrupt des Timers und zudem alle Interrupts **global**.

Wie häufig tritt bei diesen Einstellungen ein Überlauf des Zählregisters auf? Geben Sie den gerundeten Wert in Überläufe pro Sekunde an.

h) Implementieren Sie die **ISR_INT0** Interrupt Service Routine, die den Wert des Registers **Open** invertiert.

- i) Schreiben Sie nun das Unterprogramm **GET_KEY**, welches mithilfe des gegebenen Unterprogramms **READ_ADC** eine A/D-Wandlung vornimmt und anhand des Ergebnisses der Wandlung und der im **ROM** unter **KEYTABLE** abgelegten Schwellenwerte die gedrückte Taste bestimmt. Sollte keine Taste gedrückt worden sein, so ist der im Register **tmp** zu hinterlegende Rückgabewert 255, anderenfalls soll der Rückgabewert dem Zahlenwert der Taste entsprechen. Das Unterprogramm **READ_ADC** erwartet im Register **tmp** den für die A/D-Wandlung zu verwendenden Kanal und liefert das Ergebnis der Wandlung ebenfalls im Register **tmp** zurück.

- j) Realisieren Sie nun unter dem Label **MAIN** das oben beschriebene, in einer Endlosschleife arbeitende Hauptverhalten des Systems. Nachfolgende Hinweise dienen als Anhaltspunkte und können (müssen aber nicht) bei Bedarf berücksichtigt werden.

Das Verhalten kann schrittweise wie folgt skizziert werden:

- Das System wird in den Ruhemodus versetzt (alles außer der roten LED abschalten).
- In einer Schleife wiederholtes Auslesen des Distanzwertes mit Hilfe des **READ_ADC** Unterprogramms bis die gegebene Schwelle **maxDist** unterschritten wird.
- Aktivieren des Tastenfeldes (**PA0** = 1).
- Sequentielles Einlesen der acht vom Benutzer einzugebenden Ziffern innerhalb einer Schleife und Ablegen dieser im **RAM** unter dem Label **INPUT**. Hierzu soll das Unterprogramm **GET_KEY** verwendet werden. Es ist darauf zu achten, dass nach jedem Eingeben einer Ziffer die entsprechende Taste wieder losgelassen werden muss, bevor die nächste Ziffer eingegeben werden kann.
- Prüfung der Eingabe auf Übereinstimmung mit dem unter dem Label **CODE** abgelegten Systemcode.
- Keine Übereinstimmung führt sofort zum Übergang des Systems in den Ruhemodus (Anfang).
- Eine Übereinstimmung führt zum Einschalten der grünen und zum Abschalten der roten LED.
- Nun wird solange gewartet, bis der Taster betätigt wurde (über Register **Open** realisierbar).
- Im Anschluss erfolgt das Öffnen des Schlosses (**PA3** = 1) sowie das Einschalten eines Tons (**PA4** = 1) für fünf Sekunden. Beachten Sie, dass die **ISR_TIMER0** das Register **qSec** bei jedem Aufruf inkrementiert.
- Unabhängig davon, ob die Tür geöffnet und der Raum betreten wurde, erfolgt nach Ablauf der 5 Sekunden das Zurückversetzen des Systems in den Ruhemodus.

$\Sigma_{A3} =$ _____ Punkte