

Aufgabe 1: Technologien und Grundlagen

a) Gegeben sei folgende Wahrheitstabelle:

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

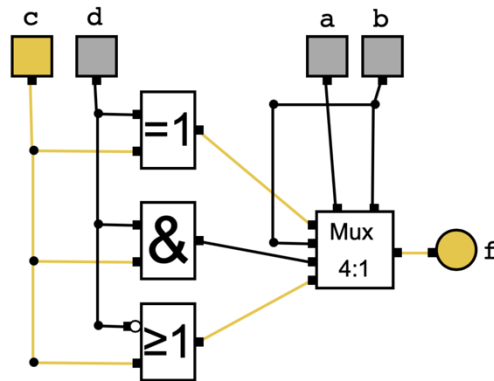
Geben Sie $f(a, b, c, d)$ in **konjunktiver kanonischer Normalform (KKN)** an.

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) Geben Sie die durch die nachfolgende Schaltung realisierte Funktion an. Ihnen stehen als Verknüpfungen UND, ODER sowie die Negation zur Verfügung.

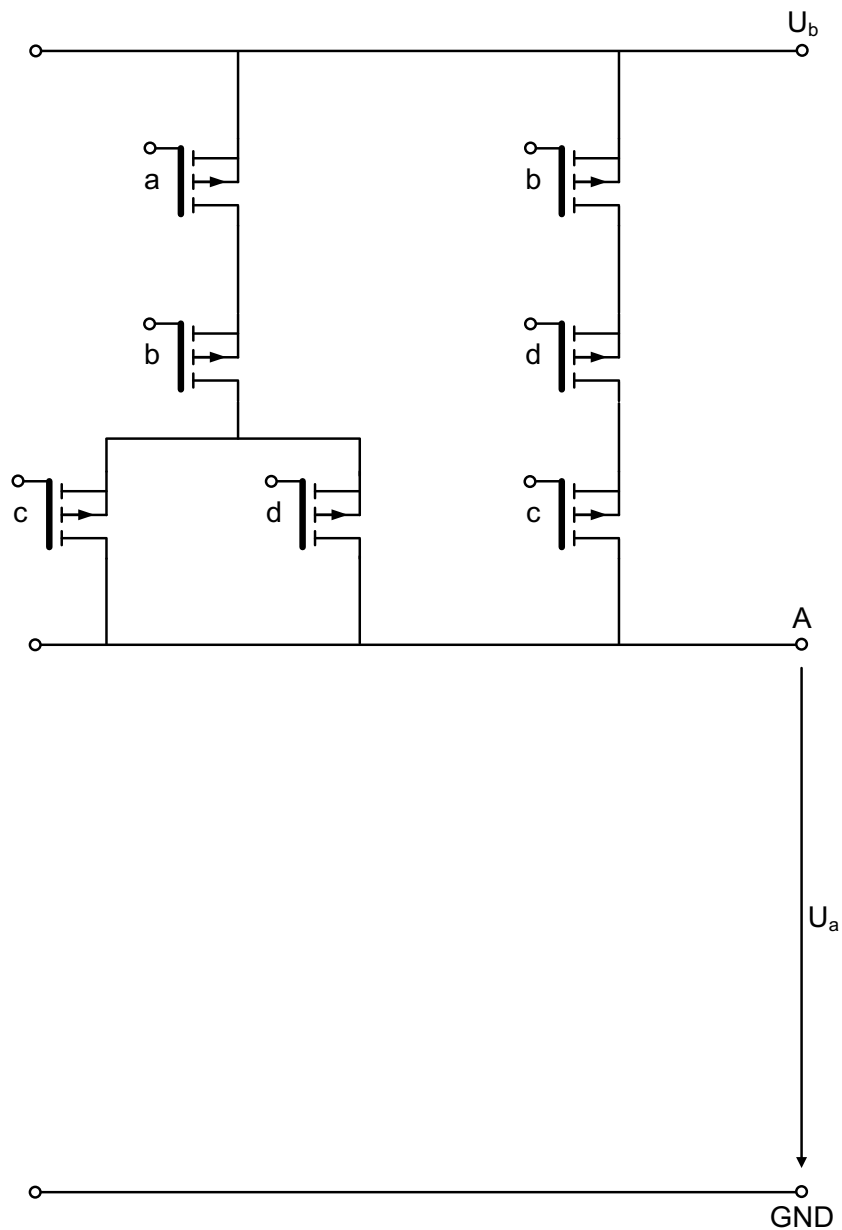


- c) Formen Sie $h(a, b, c)$ so um, dass sich die Schaltfunktion ausschließlich aus NAND-Ausdrücken über jeweils zwei Termen zusammensetzt. Das Resultat darf keine Negationen in Form eines Negationsstrichs enthalten. Verwenden Sie für die finale Darstellung die Notation: $(a \mid b)$.

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

Zeichnen Sie den Schaltplan der Schaltungsfunktion $h(a, b, c)$ unter Verwendung von Schaltsymbolen neuer DIN-Norm bestehend aus NAND-Gattern mit zwei Eingängen.

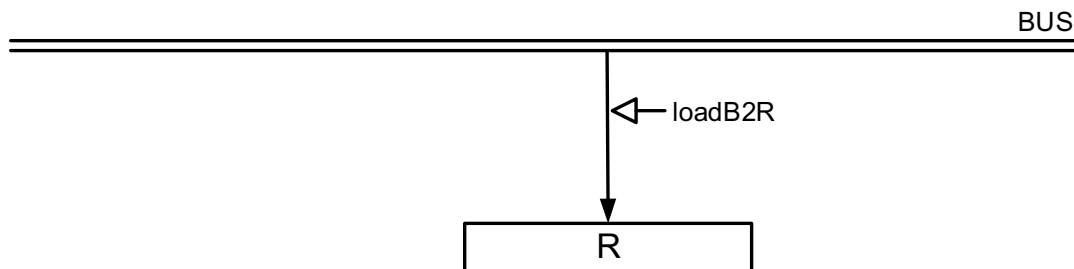
- d) Vervollständigen Sie die nachfolgend gegebene (Komplementär-)Schaltung um die Grundschal-
tung und geben Sie die durch diese Schaltung am Ausgang A realisierte Funktion $f(a, b, c, d)$ expli-
zit an. Verwenden Sie folgende Symbole für n-Kanal-  und p-Kanal-CMOS-Transistoren .



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



- a) Ergänzen Sie das obige Blockschaltbild des Operationswerks so, dass es die nachfolgend beschriebenen Komponenten und Funktionalitäten aufweist. Die dargestellten Komponenten erlauben es mit Hilfe des Steuersignals *loadB2R*, den auf dem *BUS* anliegenden Wert in das Register *R* zu laden. Sämtliche Register, der Bus sowie die Ein- und Ausgänge der *ALU* sind 16 Bit breit. Weiterhin soll das Operationswerk das Register *CNT* besitzen, dessen Wert über das Steuersignal *clrCNT* gelöscht und mit *incCNT* inkrementiert werden kann. Über die Steuersignale *loadR2T* und *loadT2R* soll es möglich sein, den Wert aus dem Register *R* in das Register *TMP* zu kopieren (*loadR2T*) und umgekehrt (*loadT2R*). Ferner ermöglicht das Steuersignal *lshTMP* den logischen Rechtsshift des Inhalts vom Register *TMP* mit dem Nachziehen einer Eins. Das Register *M* verfüge über das Steuersignal *clrM* zum Löschen des Inhalts sowie das Steuersignal *incM* zum Inkrementieren des Registerwertes. Ferner erlaube das Signal *serM* das Setzen des Registerwertes auf 0xFFFF (65535). Für die Berechnung besitzt das Operationswerk eine *ALU* mit den zwei Eingängen *A* und *B*. Der Eingang *A* sei 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.

Hinweis: Das Setzen des Z-Flags muss im Rahmen dieser Aufgabe nicht betrachtet werden.

- b) Ergänzen Sie den nachfolgend gegebenen Pseudocode unter Zuhilfenahme der zu den einzelnen Takten (Zeilen) korrespondierenden Steuersignale.

RT-Code	Takt	Steuersignale
declare bus BUS(15:0) declare register R(15:0), CNT(4:0), M(15:0), TMP(15:0)		
INIT:		
R <- BUS, CNT <- 0;	#Takt0	loadB2R, clrCNT
_____;	#Takt1	loadR2T
TEST:		
_____;	#Takt2	serM, loadT2R
_____;	#Takt3	sub, clrM
if (Z = 1) then goto END fi;	#Takt4	
_____;	#Takt5	loadT2R, incM;
_____;	#Takt6	and;
if (Z = 0) then _____		lsrTMP
goto CHECK		
else _____		lsrTMP
goto TEST		
fi;	#Takt7	
CHECK:		
_____;	#Takt8	loadT2R
_____;	#Takt9	and
if (Z = 1) then _____,		lsrTMP
_____		incCNT
goto TEST		
else goto TEST		
fi;	#Takt10	
END: goto END;	#Takt11	

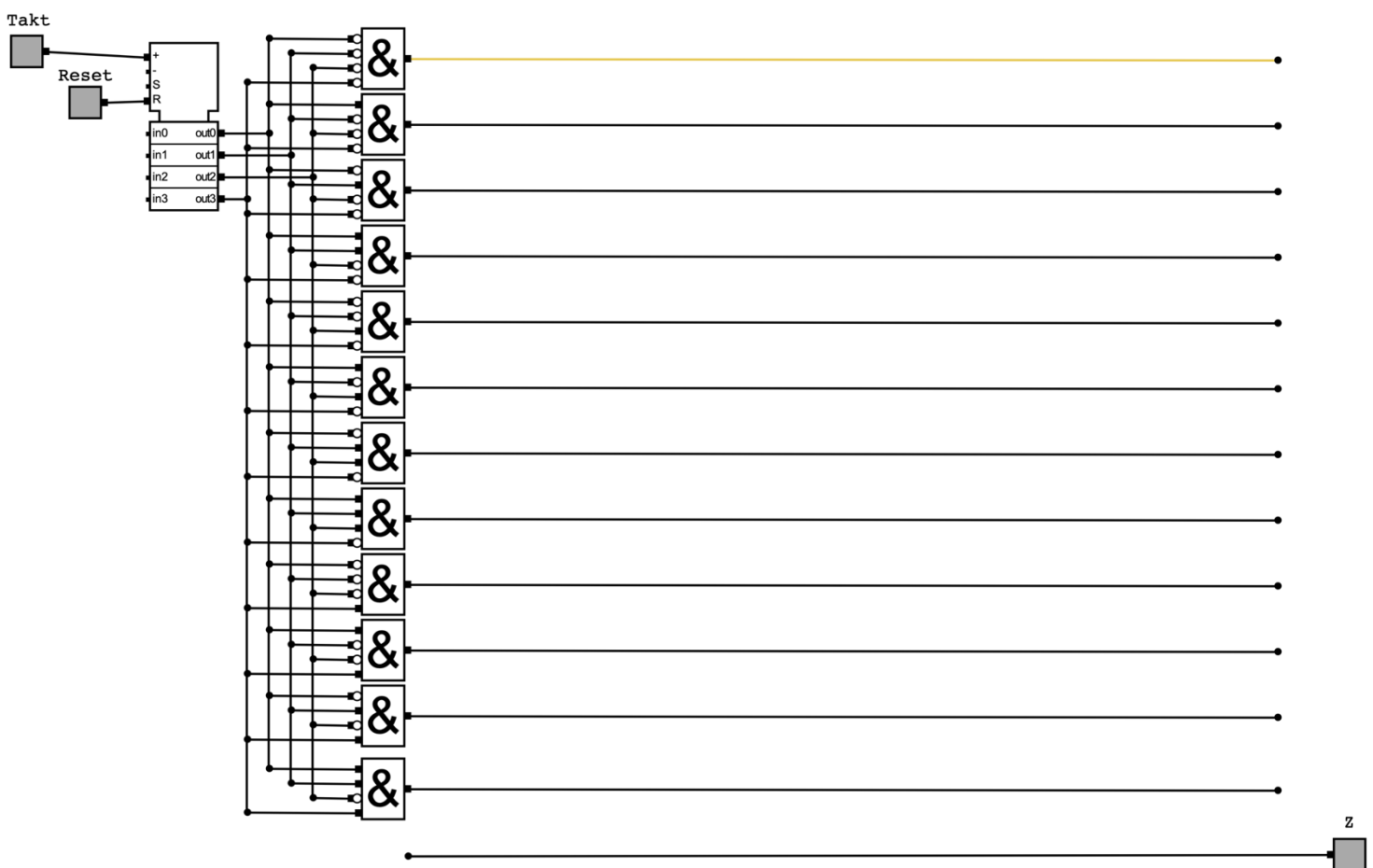
Matrikelnummer: _____

Studiengang: _____

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

- d) Welche Berechnung führt dieser Algorithmus durch? Geben Sie die semantische Bedeutung in einem Satz an.

- e) Entwerfen Sie ein Steuerwerk auf Basis eines Binärzählers, welches den Algorithmus auf dem Operationswerk realisiert, indem Sie den nachfolgenden Entwurf vervollständigen. Die Sprünge selbst müssen hier nicht realisiert werden.



☐ clrCNT
 ☐ incCNT
 ☐ lsrTMP
 ☐ loadB2R
 ☐ loadT2R
 ☐ loadR2T
 ☐ clrM
 ☐ incM
 ☐ serM
 ☐ and
 ☐ sub

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

Aufgabe 3: Assemblerprogrammierung

Sprinkleranlage für den Garten

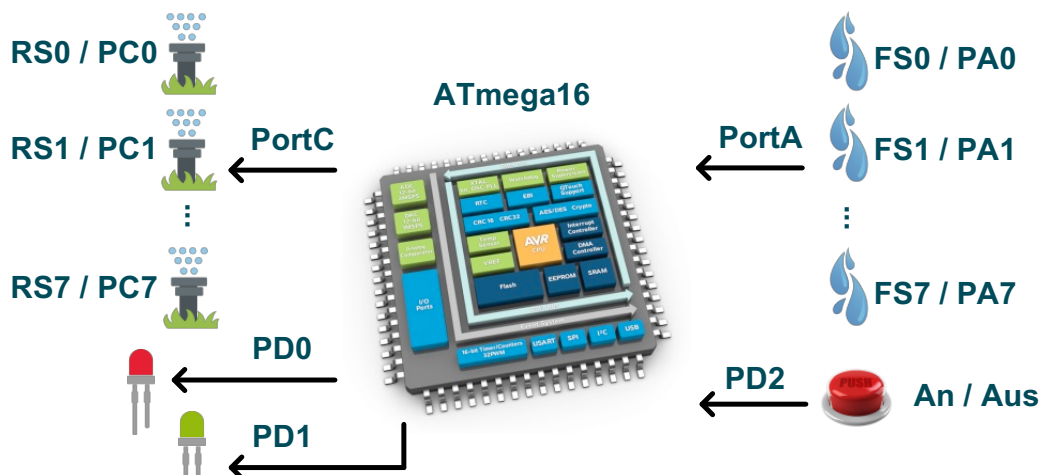


Abbildung 1: Beschaltung des ATmega16 für die Sprinkleranlage

Ziel dieser Aufgabe ist die Implementierung einer Sprinkleranlage für den Garten in Assembler auf dem aus der Übung bekannten Mikrokontroller ATmega16. Dieser ist mit acht Feuchtigkeitssensoren, acht Sprinklern, zwei LEDs (rot – PD0 und grün – PD1) sowie einem Taster verschaltet, siehe Abbildung 1.

Das Verhalten der Anlage kann wie folgt skizziert werden. Ist die Anlage abgeschaltet, so leuchtet die rote LED. Durch das Betätigen des Tasters, kann die Anlage ein- bzw. abgeschaltet werden. Eine eingeschaltete Anlage wird durch eine dauerhaft leuchtende grüne LED angezeigt (die rote LED ist aus). Die Anlage prüft im eingeschalteten Zustand jede Minute die Feuchtigkeitssensoren sequentiell mit Hilfe eines Vergleichs mit den zu den Sensoren gehörenden Schwellenwerten. Werden diese unterschritten, wird der zu einem Sensoren korrespondierende Sprinkler für die angegebene Dauer eingeschaltet und anschließend wieder ausgeschaltet. Die Schwellenwerte und Dauern können individuell eingestellt werden (Ablegen im ROM) und erlauben das Halten unterschiedlicher Feuchtigkeit je nach Bepflanzung des Bereichs in dem der jeweilige Sprinkler steht.

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 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.

- Ergänzen Sie den nachfolgenden Assemblercode so, dass das Ablegen der Daten direkt im Anschluss an die Interrupt Vektor Tabelle erfolgt. Bei den Daten handelt es sich um die Schwellenwerte (SW) und Zeitdauern (ZD) der acht Sprinkler bzw. Feuchtigkeitssensoren beginnend bei Sprinkler und Sensor Null. Auf die Daten kann unter dem Label **ThresDura** zugegriffen werden.

; Ablegen direkt nach der Interrupt Vektor Tabelle

```
ThresDura:      ;SW  ZD
                .db 124, 30
                .db 100, 60
                .db 200, 10
                .db 30,  45
                .db 90,  30
                .db 180, 20
                .db 150, 30
                .db 70,  60
```

In welchem Speicher werden die Daten durch den vorherigen Assemblercode 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 **Tmp2** und das Register **R18** unter dem Namen **OnOff** angesprochen werden können. Ferner soll das Register **R25** als Übergabeparameter unter dem Namen **Counter** genutzt werden. Weitere von Ihnen eventuell genutzte Namen für einzelne Register müssen hier ebenfalls definiert werden. Der aktuelle Zustand der Sprinkler wird in **R19** unter dem Namen **Sprinkler** hinterlegt (Bit 0 korrespondiert zu Sprinkler 0 und eine 1 bedeutet, der Sprinkler ist aktiviert).

- 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 1 Compare Match Interrupt** zur **ISR_TIMER1** gesprungen wird.

- d) Für die Messungen der Bewässerungsdauern werden acht Bytes im RAM ab Adresse \$60 benötigt, die unter dem Label **TIMES** zugreifbar sein sollen. Ferner ist jeweils ein Byte unter dem Namen **SECONDS** (zum Messen der verstrichenen Sekunden) und **MINUTES** (zum Messen der verstrichenen Minuten) zu reservieren.

Unter welcher Adresse (explizit angeben) ist der Wert hinter dem Label **MINUTES** 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. Initialisieren Sie das Register **OnOff** sowie die Werte im RAM unter den Labeln **SECONDS** und **MINUTES** mit dem Wert Null. 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 **steigenden Flanke** ausgelöst wird.

- g) Die Messung der Zeitdauer erfolgt mit Hilfe des **Timer/Counter 1**. Konfigurieren Sie ihn so, dass er **einmal pro Sekunde** einen Compare Match Interrupt auslöst. Verwenden Sie den **CTC-Modus** und nutzen Sie einen **Prescaler** von **64**. Berechnen Sie den entsprechenden Vergleichswert. Aktivieren Sie den Interrupt des Timers und zudem alle Interrupts **global**.

- h) Implementieren Sie die Interrupt-Service-Routine **ISR_TIMER1** die den unter dem Label **SECONDS** im RAM abgelegten Wert inkrementiert und aktualisiert, sobald das System eingeschaltet ist.

- i) Implementieren Sie die **ISR_INT0** Interrupt Service Routine, die überprüft, ob das System ein- (Wert von **OnOff** = 0xFF) oder ausgeschaltet (Wert von **OnOff** = 0) ist und entsprechend zwischen den Zuständen wechselt. Bedenken Sie, dass zusätzlich zum Ändern des Wertes von **OnOff**, die LEDs beschaltet werden müssen. Ferner muss ein Ausschalten des Systems zum Abschalten der Sprinkler führen.

j) Die Initialisierung des A/D-Wandlers sei nachfolgend gegeben.

```
ldi Tmp, (1 << REFS0) | (1 << ADLAR)
out ADMUX, Tmp

ldi Tmp, (1 << ADEN) | (1 << ADPS1)
out ADCSRA, Tmp
```

Welche Referenzspannung und welchen Prescaler nutzt der A/D-Wandler? Was können Sie über die Art der Speicherung des Ergebnisses sagen?

k) Nachfolgend ist das Unterprogramm **READ_ADC** teilweise gegeben. Ergänzen Sie den Code so, dass der in **R25 (Counter)** übergebene Kanal eingestellt wird, bevor die A/D-Wandlung beginnt.

```
READ_ADC:
; TODO Einstellen des Kanals

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

in R25, ADCH
ret
```

Wird für die A/D-Wandlung ein Interrupt mit oder ohne SLEEP Modus genutzt? Begründen Sie Ihre Antwort und benennen Sie gegebenenfalls die Methodik mit der hier gearbeitet wird.

- l) Realisieren Sie nun unter dem Label **MAIN** das Hauptverhalten des Systems. Das in einer Endlosschleife arbeitende Verhalten sei nachfolgend schrittweise skizziert
- In einer Schleife darauf warten, bis das System eingeschaltet wird und 60 Sekunden vergangen sind
 - Inkrementieren des unter dem Label **MINUTES** im RAM abgelegten Wertes und Zurücksetzen des Wertes unter dem Label **SECONDS**
 - Sequentielles Prüfen der Sprinkler in einer **Schleife**
 - Ist der Sprinkler aktiv, muss überprüft werden, ob seine Bewässerungsdauer erreicht ist oder nicht (**TIMES** des Sprinklers im RAM = **MINUTES**) und er gegebenenfalls deaktiviert werden muss (Register **Sprinkler** und **PORTC**)
 - Ist er inaktiv, muss der korrespondierende Feuchtigkeitssensor ausgelesen werden (**READ_ADC** mit entsprechendem Kanal) und mit der Schwelle verglichen werden (**ThresDura**). Wurde die Schwelle unterschritten, muss der Sprinkler aktiviert werden (Register **Sprinkler** und **PORTC**) und seine Bewässerungsdauer (**TIMES** des Sprinklers im RAM = **MINUTES** + Dauer des Sprinklers aus ROM) aktualisiert werden.

$\Sigma_{A3} =$ _____ Punkte