Lehrstuhl für Automatisierung und Informationssysteme
Prof. Dr.-Ing. B. Vogel-Heuser

Vorname:	
Nachname:	
Matrikelnummer:	

Prüfung – Informationstechnik

Wintersemester 2018 / 2019

08.03.2019

Bitte legen Sie Ihren Lichtbildausweis bereit.

Sie haben für die Bearbeitung der Klausur 120 Minuten Zeit.

Diese Prüfung enthält 33 nummerierte Seiten inkl. Deckblatt.

Bitte prüfen Sie die Vollständigkeit Ihres Exemplars!

Bitte nicht mit rot oder grün schreibenden Stiften oder Bleistift ausfüllen!

Aufgabe	Erreichte Punkte
1	
2	
3	
4	
5	
6	
7	
8	
∑GL	
9	
10	
11	
12	
13	
∑BS	
14	
15	
16	
17	
18	
∑MSE	
19	
20	
21	
22	
23	
∑C	
∑	

Aufgabe GL: Grundlagen

Aufgabe GL:
49 Punkte

1. Umrechnung zwischen Zahlensystemen

Nennen Sie die Basen der beiden Zahlensysteme, in die sich Zahlen des 9er-Systems besonders einfach umrechnen lassen.

3	27

2. IEEE 754 Gleitkommazahlen

Rechnen Sie die Dezimalzahl $(-11{,}75)_{10}$ in eine Gleitkommazahl (angelehnt an die IEEE 754 Darstellung) mit folgender Formatierung um:

V e (4 Bit) M (5 Bit)

Hinweis: Ergebnisse und Nebenrechnungen außerhalb der dafür vorgesehenen Textblöcke werden nicht bewertet.

Vorzeichen

V= 1

Mantisse (Binärzahl und Normalisiert)

M_2= $(1011{,}11)_2 = (1{,}01111 * 2^3)_2$

Exponent

E = 3

Bias und biased Exponent

B= $2^{(x-1)}-1 = 7$ **e =** $B + E = (10)_{10} = (1010)_2$

Vollständige Gleitkommazahl (10 bit)

1 1010 01111

Kann die gegebene Dezimalzahl bei gegebener Genauigkeit als Binärzahl exakt dargestellt werden?

X	
Ja	Nein

3. Quer- und Längsparität

Folgende Nachricht wurde mittels gerader Parität gegen Übertragungsfehler geschützt. Finden und korrigieren Sie den/die minimalen Übertragungsfehler.

1	0	1	**0**
0	1	1	**0** ~~**1**~~
1	1	1	**1**
0	**0**	**1**	**1**

4. Hamming-Distanz

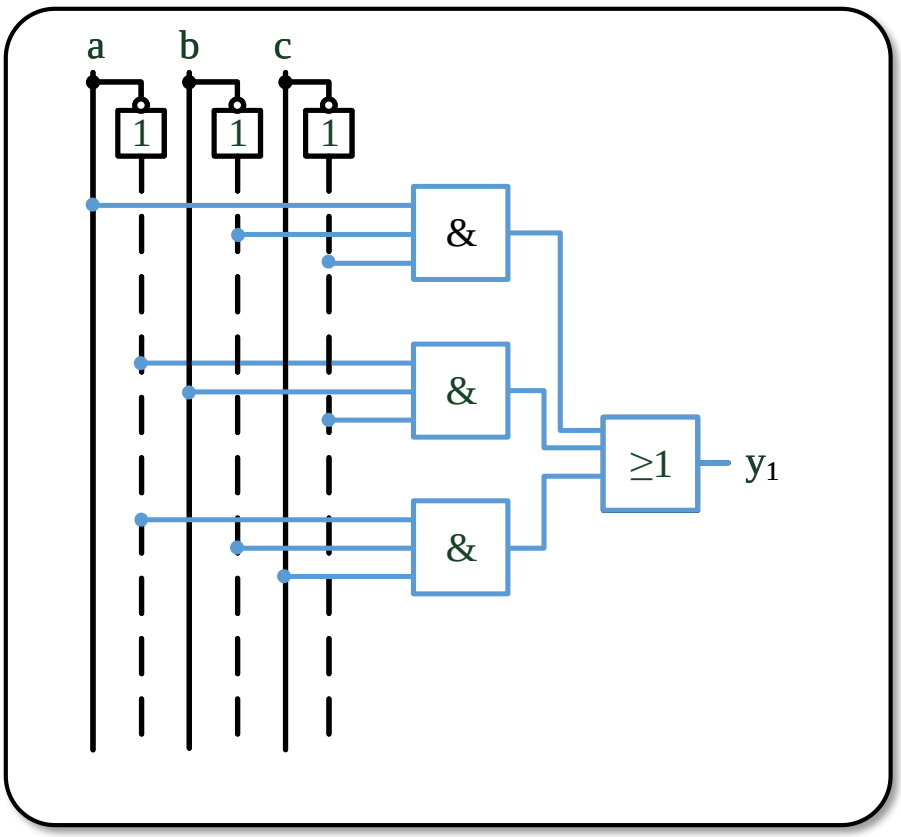
Gegeben sei eine Hamming-Distanz von 4. Wie viele Bitfehler können damit behoben werden? Wie lautet die Formel zur Bestimmung der Anzahl erkennbarer Bitfehler

Erkennungsformel: **h = e + 1**

Beheben: **h = 2f + 1 → f = 1**

5. Logische Schaltungen und Schaltbilder

Gegeben sei nebenstehende Wahrheitstabelle. Erstellen Sie das zugehörige Schaltbild der DNF.



a	b	c	y_1
1	1	1	0
1	1	0	0
1	0	1	0
1	0	0	1
0	1	1	0
0	1	0	1
0	0	1	1
0	0	0	0

Welche Schaltung ist hier dargestellt? Bitte kreuzen Sie den richtigen Fachbegriff an.

(x) XOR-Gatter für 3 Eingänge

( ) Dreikanal Demultiplexer

( ) Zweikanal-Multiplexer mit Selektionseingang „b“

( ) 1 Bit Synchron Zähler

6. Normalformen und Minimierung

a)Welche Terme sollten für die nebenstehende Wahrheitstabelle zusammengefasst werden, um eine Schaltung mit möglichst wenigen Schaltgliedern zu erhalten?

(X) Minterme

( ) Maxterme

a	b	c	d	y_1
0	0	0	0	0
0	0	0	1	X
0	0	1	0	X
0	0	1	1	1
0	1	0	0	1
0	1	0	1	0
0	1	1	0	0
0	1	1	1	0
1	0	0	0	X
1	0	0	1	1
1	0	1	0	0
1	0	1	1	X
1	1	0	0	X
1	1	0	1	0
1	1	1	0	X
1	1	1	1	X

b) KV-Diagramme

Ermitteln Sie die minimierte DNF für die nebenstehende Wahrheitstabelle mittels eines KV-Diagramms.

Hinweis 1: Das Einzeichnen der Schleifen in das KV-Diagramm ist als Lösung ausreichend, die minimierte Formel muss nicht abgelesen werden.

Hinweis 2: Das zweite abgebildete KV-Diagramm dient als Ersatz, falls Sie sich verzeichnen. Kennzeichnen Sie durch Ankreuzen im Feld „dieses KV-Diagramm werten", welches KV-Diagramm bewertet werden soll.

Hinweis 3: „X“ entspricht „don‘t care“-Einträgen

[X] Dieses KV-Diagramm werten

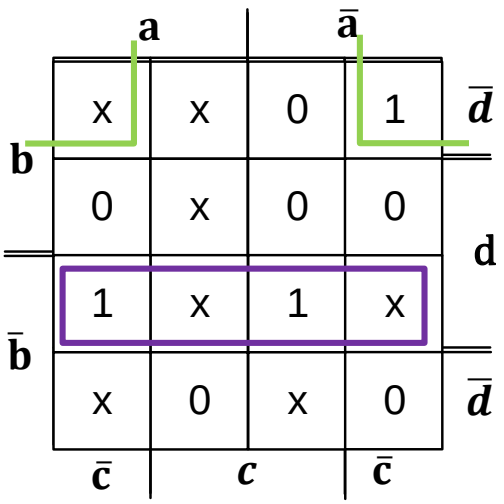


Bild G-6.1: *KV-Diagramm 1*

[ ] Dieses KV-Diagramm werten

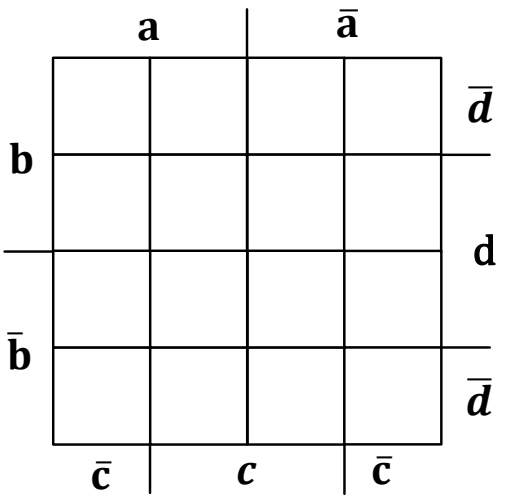


Bild G-6.2: *KV-Diagramm 2*

7. Flip-Flops

Gegeben ist die folgende Master-Slave-Flip-Flop-Schaltung (MS-FF).

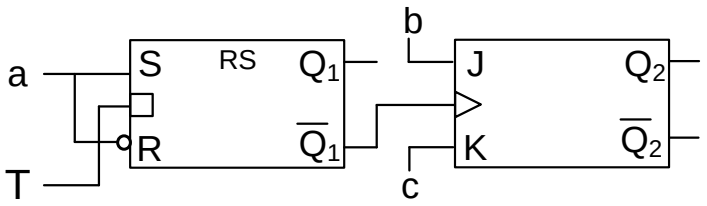
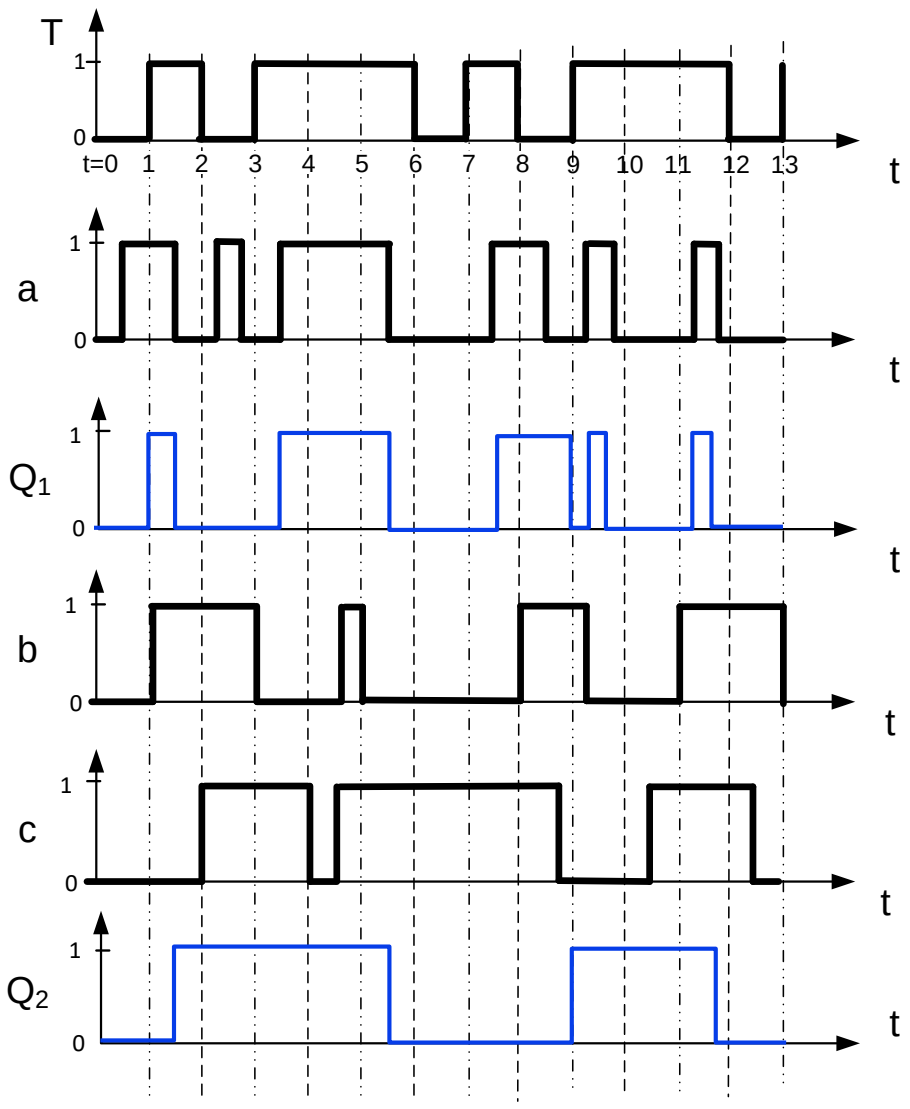


Bild G-7.1: *MS-FF*

Bei t = 0 sind die Flip-Flops in folgendem Zustand: $Q_1 = Q_2 = 0$.

Analysieren Sie die Schaltung für den Bereich $t = [0; 13[$, indem Sie für die Eingangssignale a, b, c und T die zeitlichen Verläufe für Q_1 und Q_2 in die vorgegebenen Koordinatensysteme eintragen.

Hinweis: Signallaufzeiten können bei der Analyse vernachlässigt werden.



8. MMIX-Rechner

Gegeben sei der nachfolgende Algorithmus sowie ein Ausschnitt der MMIX-Code-Tabelle (Bild G-8.1), eines Register- (Bild G-8.2) sowie eines Datenspeichers (Bild G-8.3):

	0x_0	0x_1	…	0x_4	0x_5	…
	0x_8	0x_9		0x_C	0x_D	
…	…	…	…	…	…	…
0x1_	FMUL	FCMPE	…	FDIV	FSQRT	…
	MUL	MUL I		DIV	DIV I	
0x2_	ADD	ADD I	…	SUB	SUB I	…
	2ADDU	2ADDU I		8ADDU	8ADDU I	
…	…	…	…	…	…	…
0x8_	LDB	LDB I	…	LDW	LDW I	…
	LDT	LDT I		LDO	LDO I	
0x9_	LDSF	LDSF I	…	CSWAP	CSWAP I	…
	LDVTS	LDVTS I		PREGO	PREGO I	
0xA_	STB	STB I	…	STW	STW I	…
	STT	STT I		STO	STO I	
…	…	…	…	…	…	…
0xE_	SETH	SETMH	…	INCH	INCMH	…
	ORH	ORMH		ANDNH	ANDNMH	

Bild G-8.1: *MMIX-Code-Tabelle*

Algorithmus: $\frac{(a-b)^2+c}{255}$

Registerspeicher		
Adresse	Wert vor Befehlsausführung	Kommentar
…	…	…
$0x86	0x00 00 00 00 00 00 62 0F	Nicht Veränderbar
$0x87	0x00 00 00 00 00 00 00 06	Variable a
$0x88	0x00 00 00 00 00 00 00 0B	Variable b
$0x89	0x00 00 00 00 00 00 00 01	Variable c
$0x8A	0x00 00 00 00 00 00 62 08	Zwischenergebnis
$0x8B	0x00 00 00 00 00 00 01 00	Nicht Veränderbar
…	…	

Bild G-8.2: *Registerspeicher*

a) Im Registerspeicher eines MMIX-Rechners befinden sich zu Beginn die in Bild G-8.2 gegebenen Werte. In der Spalte *Kommentar* wurde angegeben, welche Daten diese enthalten und wofür die einzelnen Zellen benutzt werden müssen. Setzen Sie zuerst den Inhalt der vorgegebenen Registerspeicherzelle *Variable c* im aktuellen Zustand auf den Dezimalwert 1024. Führen Sie anschließend den gegebenen Algorithmus aus. Übersetzen Sie diese Operationen in Assembler-Code mit insgesamt maximal 5 Anweisungen. Verwenden Sie dazu lediglich die in Bild G-8.1 umrahmten Befehlsbereiche. Speichern Sie die Zwischenergebnisse nach jedem Befehl des Algorithmus in der Registerzelle mit dem Kommentar *Zwischenergebnis*.

1	SETL $0x89 0x04 00
2	SUB $0x8A $0x87 $0x88
3	MUL $0x8A $0x8A $0x8A
4	ADD $0x8A $0x8A $0x89
5	DIVI $0x8A $0x8A 0xFF

oder MULI $0x89 $0x8B 0x04

b) Nehmen Sie an, der Inhalt der Registerspeicherzelle *Zwischenergebnis* sei nach Ausführung des Algorithmus 0x 01 23 45 67 89 AB CD EF. Speichern Sie diesen als Tetra im Datenspeicher ab Adresse 0x0 … 62 0C. Wie lautet der vollständige Assembler-Befehl zum Speichern? Welche Werte befinden sich nach Ausführung des Speicherbefehls im Datenspeicher?

Befehl:

STTI $0x8A $0x86 0x00

Datenspeicher	
Adresse	Wert
…	…
0x00 00 00 00 00 00 62 07	
0x00 00 00 00 00 00 62 08	**00**
0x00 00 00 00 00 00 62 09	**00**
0x00 00 00 00 00 00 62 0A	**00**
0x00 00 00 00 00 00 62 0B	**00**
0x00 00 00 00 00 00 62 0C	**89**
0x00 00 00 00 00 00 62 0D	**AB**
0x00 00 00 00 00 00 62 0E	**CD**
0x00 00 00 00 00 00 62 0F	**EF**
0x00 00 00 00 00 00 62 10	
0x00 00 00 00 00 00 62 11	
…	…

Bild G-8.3: *Datenspeicher*

Aufgabe BS: Betriebssysteme

Aufgabe BS:
51 Punkte

9. Speicherreservierung

a) Gegeben sei der folgende Speicherzustand eines Stapelspeichers. Vervollständigen Sie den Ablauf mittels der Methode Least Frequently Used (LFU). Gleichstand sollen Sie mit der Methode Last In First Out (LIFO) auflösen.

Hinweis: Nur Antworten innerhalb der Lösungskästen werden gewertet!

Zeit		0	1	2	3	4
Referenz		-	1	0	2	1
Seite 1		3	3	3	3	3
Seite 2		2	1	0	0	0
Seite 3		4	4	4	2	1
Seite 4		5	5	5	5	5
Zähler	0	3	2	3	3	3
	1	1	1	1	1	2
	2	1	0	0	1	1
	3	3	2	2	2	2
	4	2	1	1	1	1
	5	4	3	3	3	3

b) Beurteilen Sie die nachfolgenden Aussagen zum Thema Speicherreservierung und Adressumformung auf Ihre Korrektheit.

	wahr	**falsch**
Der logische Adressraum kann größer sein als der physische	(X)	( )
Segmentierung kann nur zusammen mit Seitenwechsel-verfahren angewendet werden	( )	(X)
Das Verfahren der Segmentierung unterbindet die Entstehung unbenutzbar kleiner Teile des Speichers	( )	(X)

10. Asynchrones Scheduling, präemptiv, Round Robin (RR)

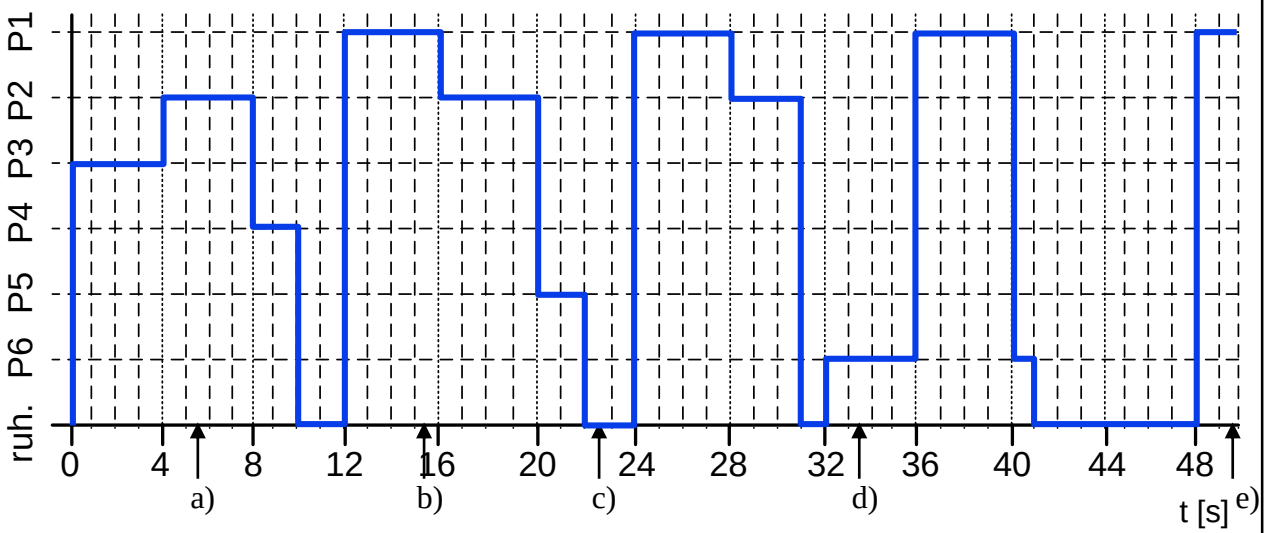
Gegeben seien die folgenden sechs Prozesse (Bild BS-10.1), welche jeweils ab dem Zeitpunkt *Start* eingeplant werden sollen. Zur Abarbeitung eines Tasks wird die Rechenzeitspanne *Dauer* benötigt. Periodische Tasks werden mit der Häufigkeit *Frequenz* erneut aufgerufen. Erstellen Sie im untenstehenden Diagramm das präemptive Scheduling nach dem Schema Round-Robin für den Zeitraum $t = [0; 50[$ s für einen Einkernprozessor. Treffen innerhalb eines Zeitschlitzes mehrere Tasks ein, beachten Sie die *Prioritäten* und anschließend FIFO. Ein Zeitschlitz hat eine Größe von vier Sekunden. Tragen Sie die jeweils zu den gekennzeichneten Zeitpunkten aktiven Tasks ein.

Hinweis: Nur Antworten innerhalb des Lösungskastens werden gewertet!

	Priorität	Start	Dauer	Frequenz
P1	1 (hoch)	6 s	4 s	13 s
P2	2	1 s	11 s	Einmalig
P3	3	0 s	4 s	einmalig

	Priorität	Start	Dauer	Frequenz
P4	4	1 s	2 s	einmalig
P5	5	9 s	2 s	einmalig
P6	6 (niedrig)	23 s	5 s	einmalig

Bild BS-10.1: *Taskspezifikation*



a) P1 ( ), P2 (x), P3 ( ), P4 ( ), P5 ( ), P6 ( ), ruhend ( )

b) P1 (x), P2 ( ), P3 ( ), P4 ( ), P5 ( ), P6 ( ), ruhend ( )

c) P1 ( ), P2 ( ), P3 ( ), P4 ( ), P5 ( ), P6 ( ), ruhend (x)

d) P1 ( ), P2 ( ), P3 ( ), P4 ( ), P5 ( ), P6 (x), ruhend ( )

e) P1 (x), P2 ( ), P3 ( ), P4 ( ), P5 ( ), P6 ( ), ruhend ( )

11. Semaphoren

a) Gegeben seien die folgenden vier Tasks T1 bis T4 mit absteigender Priorität sowie die dazugehörigen Semaphoren S1 bis S4 (Bild BS-11.1). Die Startwerte der Semaphoren entnehmen Sie der Antworttabelle. Tragen Sie in der ersten Spalte der Antworttabelle den aktuell laufenden Task ein sowie im Rest der Zeile die Werte der Semaphoren nach Ausführung des jeweiligen Tasks. Bei Schleifen dürfen Sie vor der ersten Iteration abbrechen.

T1	T2	T3	T4
P(S1)	P(S2)	P(S3)	P(S4)
		P(S3)	P(S4)
…	…	…	…
V(S3)			V(S2)
V(S4)	V(S1)	V(S4)	V(S3)

Bild BS-11.1: *Semaphorenzuweisung*

Task	S1	S2	S3	S4
-	1	0	1	0
T1	**0**	**0**	**2**	**1**
T3	**0**	**0**	**0**	**2**
T4	**0**	**1**	**1**	**0**
T2	**1**	**0**	**1**	**0**
T1	**0**	**0**	**2**	**1**
T3	**0**	**0**	**0**	**2**
T4	**0**	**1**	**1**	**0**
T2	**1**	**0**	**1**	**0**

b) Wie lauten in PEARL die Befehle zur Anforderung und zur Freigabe der Semaphore S2?

Anforderung: **REQUEST S2;**

Freigabe: **RELEASE S2;**

12. Echtzeitbetriebssysteme

Im Folgenden (Bild BS-12.1) ist ein unvollständiges „erweitertes Taskzustandsdiagramm von RTOS-UH“ gegeben.

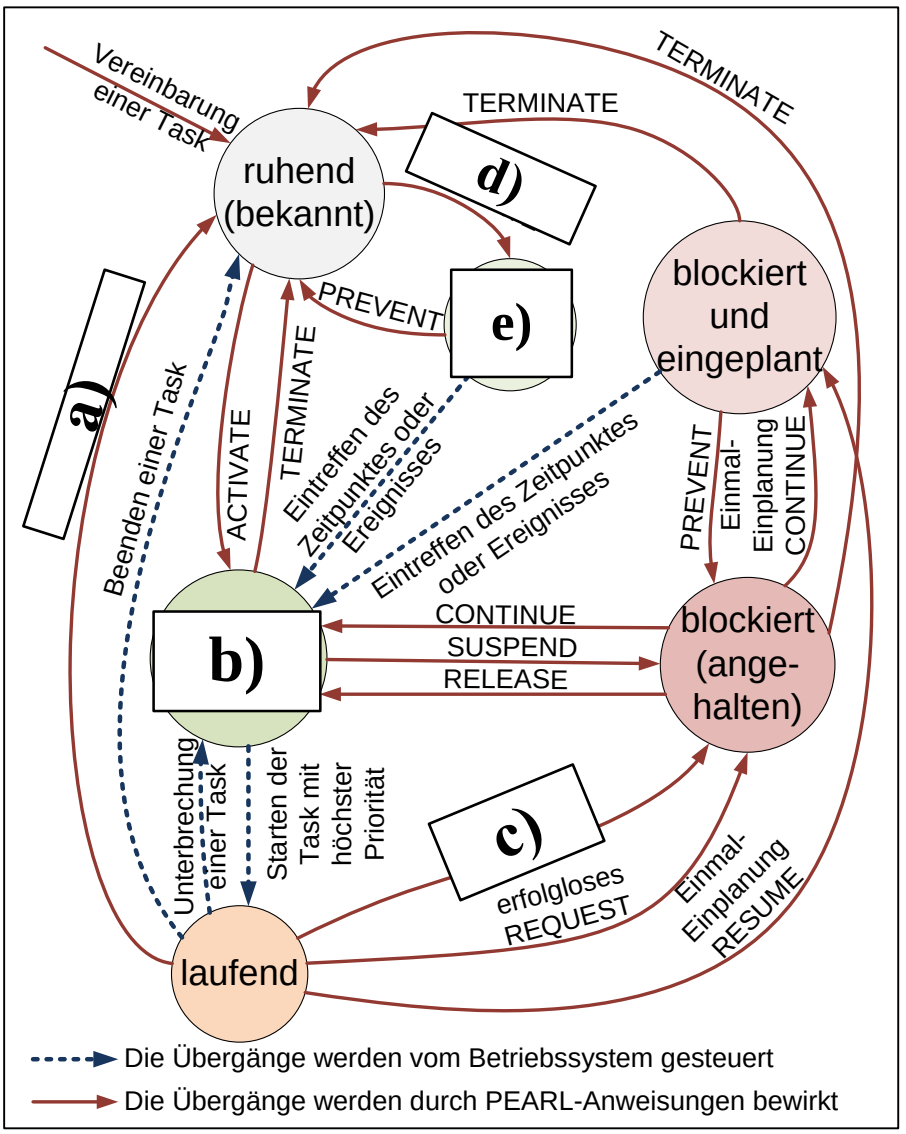


Bild BS-12.1: *Erweitertes Taskzustandsdiagramm von RTOS-UH*

Bezeichnen Sie die im Diagramm mit Buchstaben markierten Lücken:

a) TERMINATE

b) bereit oder lauffähig

c) SUSPEND

d) Einplanung oder ACTIVATE

e) Eingeplant

13. IEC 61131-3: Funktionsbausteinsprache (FBS)

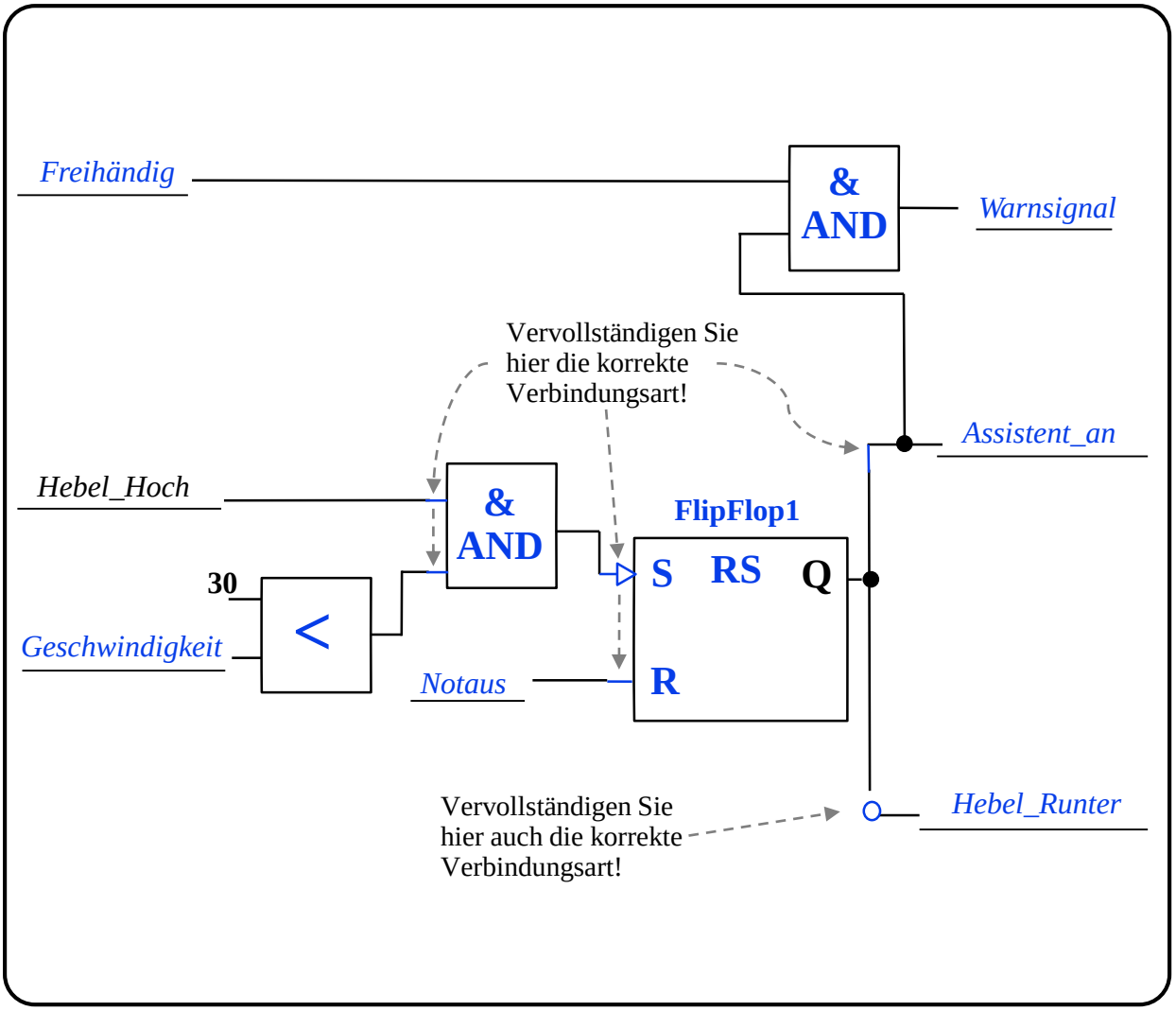
In einem Kraftfahrzeug unterstützt das teilautonome Fahrerassistenzsystem den Fahrer in bestimmten Fahrsituationen. Um Missbrauch des Systems zu vermeiden, soll eine Freihanderkennung des Fahrers in die Steuerung des Fahrassistenten integriert und mit der FBS programmiert werden.

Betätigt der Fahrer den Steuerungshebel nach oben, liegt am Eingang *Hebel_Hoch* einmalig der Wert 1 an. Das Signal zum Starten der Fahrerassistenz Steuerung (*Assistent_an*) soll dann auf 1 wechseln, sofern die *Geschwindigkeit* einen Wert größer als 30 liefert.

Ob der Fahrer freihändig fährt, detektiert ein Lenkradsensor. Dabei ist das Signal *Freihändig* wahr, wenn für länger als 3 Sekunden keine Hand am Lenker anliegt. Fährt der Fahrer freihändig und der Assistent ist an, soll ein akustisches Warnsignal ausgelöst werden, indem das Signal *Warnsignal* von 0 auf 1 gesetzt wird.

Fährt der Fahrer für 20 Sekunden ohne Hand am Lenker, sendet der Sensor am *Notaus* Signal eine 1, wodurch *Assistent_an* zurückgesetzt wird. Ist der Assistent aus, soll an *Hebel_Runter* eine 1 anliegen, was den Steuerungshebel nach unten bewegt.

Hinweis : Signalverzögerungen im System sind zu vernachlässigen.



Aufgabe MSE: Modellierung und Softwareentwicklung

Aufgabe MSE:
44 Punkte

14. Automaten

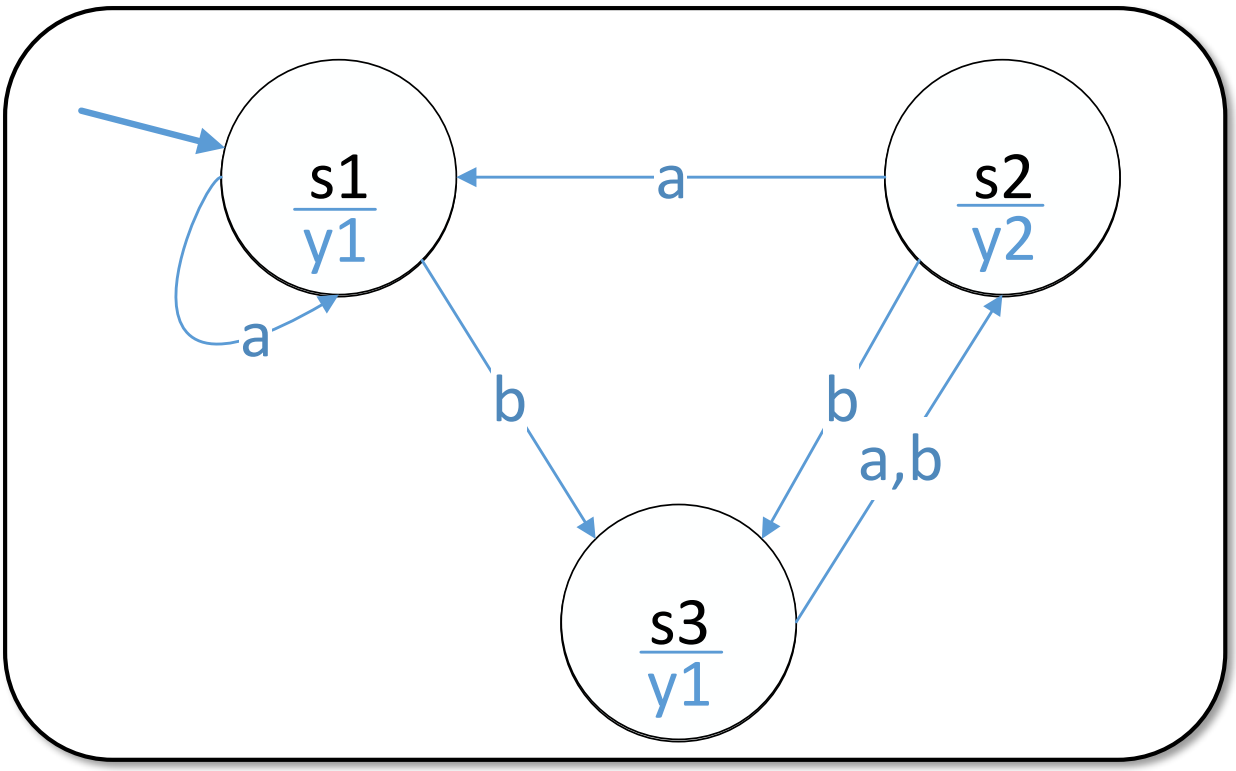
T	s1,y1	s2, y2	s3, y1
a	s1	s1	s2
b	s3	s3	s2

Bild MSE-14.1: *Übergangstabelle*

a) Ist die Übergangstabelle in Bild MSE-14.1 für einen Mealy- oder einen Moore-Automaten?

Mealy [ ] Moore [X]

b) Überführen Sie die Übergangstabelle in ihren zugehörigen Automat. Nehmen Sie an, dass s1 der Startzustand ist.



c) Welche drei Eigenschaften muss ein Problem aus der Wirklichkeit haben, um durch einen Automaten modellierbar zu sein?

- **Diskrete Ein-/Ausgabe**
- **Endliche Anzahl von Zuständen**
- **Gedächtnislosigkeit**

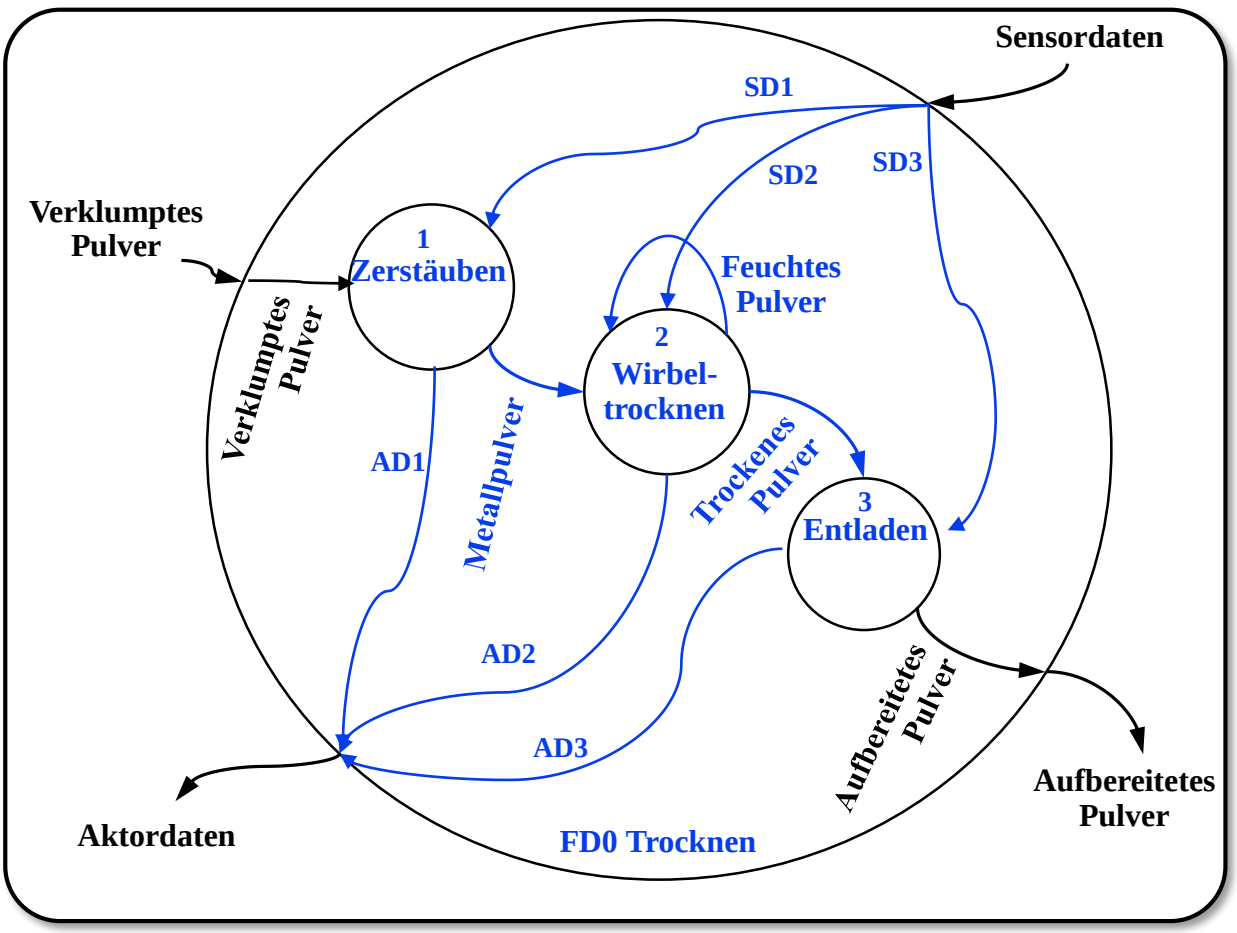
15. SA/RT: Flussdiagramm

Für die folgenden Teilaufgaben soll ein Prozess zum *Trocknen* von Metallpulver betrachtet werden, der aus drei Prozessschritten besteht.

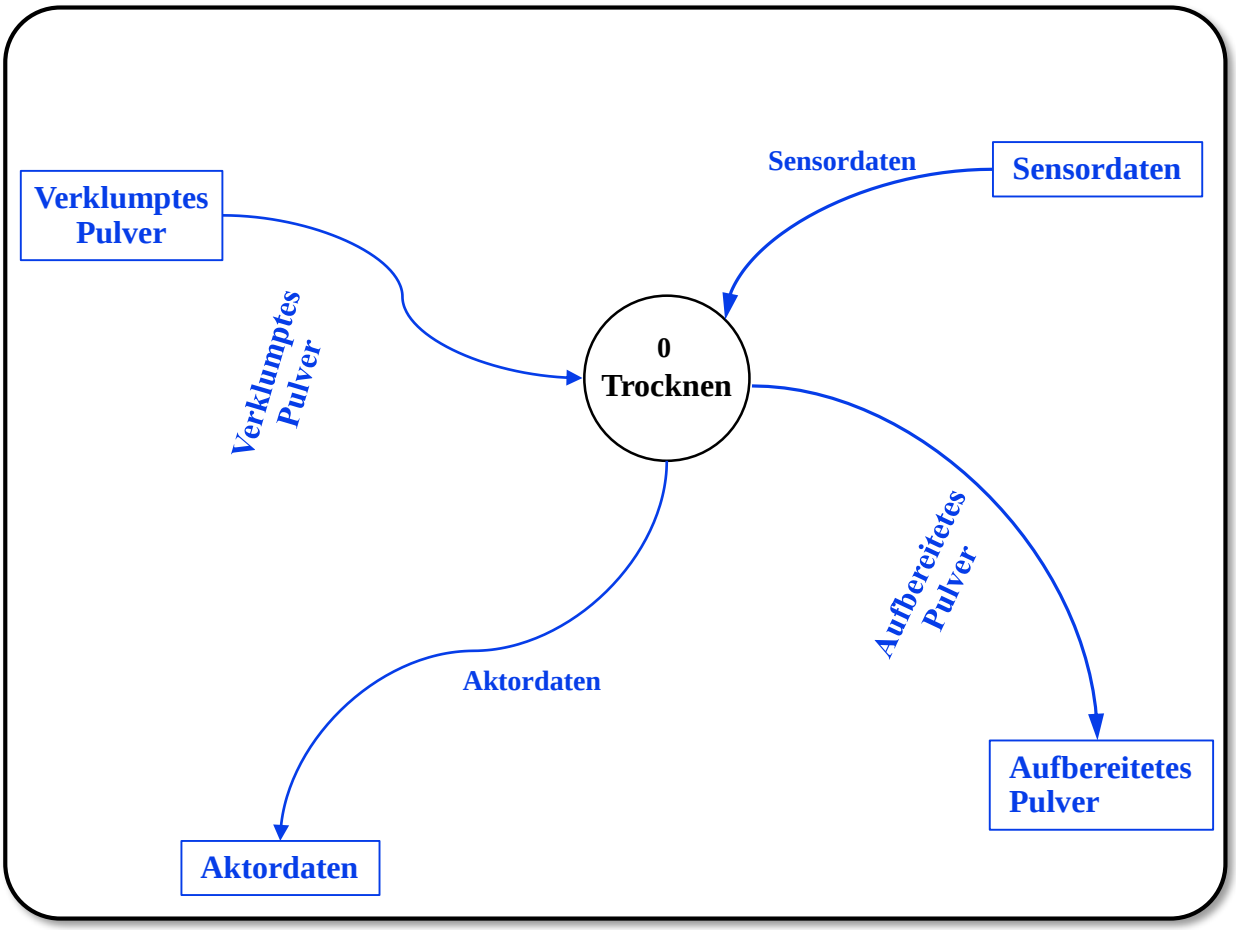
Im Prozess *Zerstäuben* (Prozess 1) wird *verklumptes Pulver* fein zerstäubt. Im Prozess *Wirbeltrocknen* (Prozess 2) wird das *Metallpulver* mittels Heißluft verwirbelt und somit Feuchtigkeit entzogen. Anschließend wird *trockenes Pulver* an den nächsten Prozess weitergeleitet, während *feuchtes Pulver* wieder in den Prozess 2 zurückgeführt wird. Im Prozess *Entladen* (Prozess 3) wird schließlich das elektrostatisch geladene Pulver entladen und als *aufbereitetes Pulver* dem Gesamtprozess entnommen.

a) Modellieren Sie den Prozess *Trocknen* (Prozess 0) mittels Strukturierter Analyse / Real-Time (SA/RT) in einem Flussdiagramm (FD0). Identifizieren Sie hierzu alle Subprozesse sowie Datenflüsse und tragen Sie diese mit Bezeichnung ein. Beachten Sie, dass Sensor- und Aktordaten eines Prozesses mit SD*Prozessnummer* und AD*Prozessnummer* (z.B. SD1 oder AD3) zusammengefasst werden. Beachten Sie außerdem folgende Informationen:

Flüsse Material: Metallpulver, trockenes Pulver, feuchtes Pulver	**Flüsse Sensordaten:** je nach Prozess: SD*Prozessnummer*, also z.B. SD1 bei „Zerstäuben“. **Flüsse Aktordaten:** Wie Sensordaten nur AD*Prozessnummer*



b) Erstellen Sie für den zuvor beschriebenen Prozess *Trocknen* das zugehörige Kontextdiagramm.



16. SA/RT: Antwortzeitspezifikation, Timing-Diagramm

Nachfolgend soll der zeitliche Ablauf des Trocknungsvorgangs in Form eines Timing-Diagramms erstellt werden, um sicherzustellen, dass der Prozess korrekt durchgeführt wird.

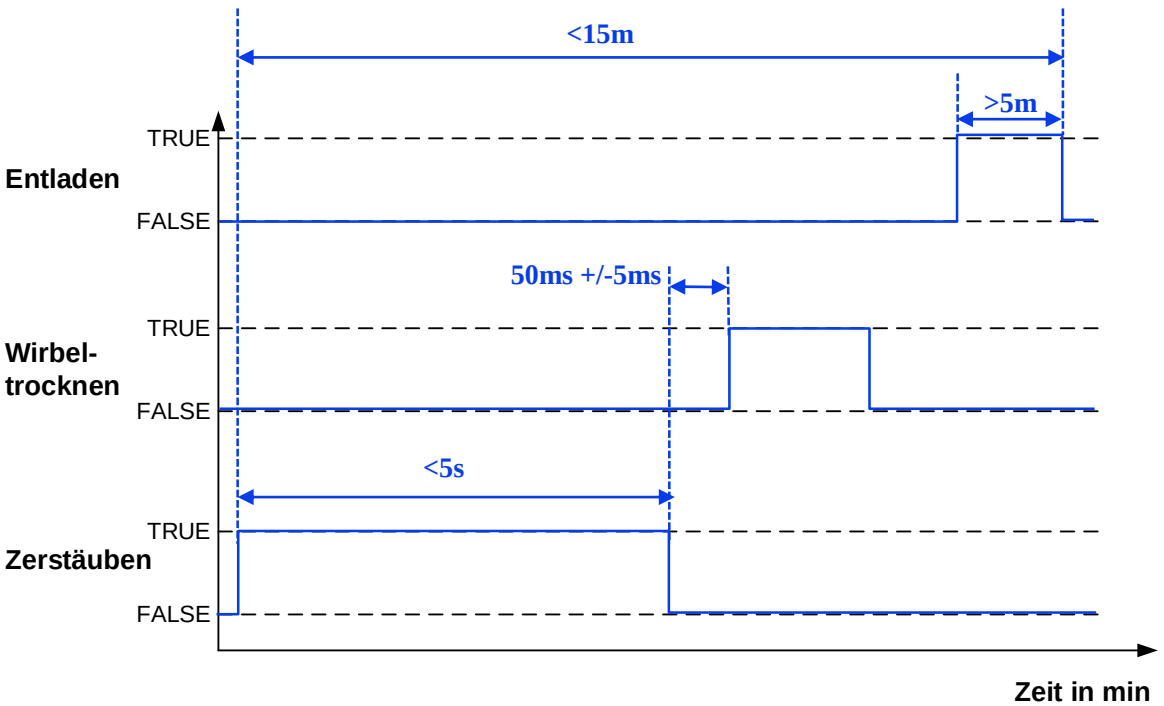
Die Zustände der Prozesse können jeweils *TRUE* (z.B. Zerstäuben aktiv) oder *FALSE* (z.B. Zerstäuben beendet) sein.

Ergänzen Sie das Timing-Diagramm gemäß folgender Angaben (Werteverläufe und Zeitangaben):

- Das Zerstäuben einer Pulvercharge (*Zerstäuben=TRUE*) muss nach spätestens 5 Sekunden abgeschlossen sein
- Der anschließende Wechsel in den Modus Wirbeltrocknen (*Zerstäuben=FALSE, Wirbeltrocknen=TRUE*) muss in exakt 50 Millisekunden mit einer Toleranz von 5 Millisekunden erfolgen
- Der abschließende Entladungsprozess (*Entladen=TRUE*) muss mindestens für 5 Minuten durchgeführt werden
- Der gesamte Trocknungsprozess darf nicht mehr als 15 Minuten benötigen

Hinweis: Eine maßstabsgetreue Darstellung der Zeiten ist nicht notwendig.

Jeweils <= bzw. >= statt < bzw. > auch ok



17. Bussysteme

Beurteilen Sie die Behauptungen auf ihre Richtigkeit hin.

Hinweis: Nur Antworten innerhalb der Lösungskästen werden gewertet!

	wahr	*falsch*
Bei MOST handelt es sich um ein dezentral gesteuertes Buszugriffsverfahren	**( )**	**(x)**
Token Ring eignet sich zum priorisierten Versand wichtiger Nachrichten	**( )**	**(x)**
Ethernet setzt CSMA/CA als Buszugriffsverfahren ein	**( )**	**(x)**
Bei EtherCAT handelt es sich um eine Master-Slave Architektur	**(x)**	**( )**

18. Unified Modeling Language

a) Das objektorientierte Paradigma lässt sich anhand von sieben Grundkonzepten beschreiben. Nennen Sie drei dieser Grundkonzepte.

Relation, Verfeinerung, Vererbung, Abstraktion, Polymorphie, Klassenbildung, Kapselung

b) Nennen Sie je zwei Beispiele für Verhaltensdiagramme und Strukturdiagramme in der UML.

Verhaltensdiagramme:

- **Zustandsdiagramm**
- **Aktivitätsdiagramm**

Strukturdiagramme:

- **Klassendiagramm**
- **Kompositionsstrukturdiagramm**

Aufgabe C: C-Programmierung

Aufgabe 19:
11 Punkte

19. C: Datentypen, Operatoren und Boolsche Algebra

a) Datentypen und Operatoren

Welche Ausgaben erzeugen die `printf`-Anweisungen in den folgenden Codefragmenten? Wählen Sie die korrekte Antwort (nur Einfachantwort möglich), bzw. füllen Sie die Lücken.

```
float x = 6.1;
int y = 2;
float f = x / y;
printf("%.2f", f);

( ) 3.050000
( ) 3.00
(x) 3.05
( ) 3
```

```
int x = 2;
int z = 65;
//'A' entspricht ASCII-Code 65
char d = 'A';

printf("%.1f", d / (float) x);
→ Ausgabe: 32.5

printf("%c", z);
→ Ausgabe: A
```

Definieren Sie die Datentypen der folgenden Variablen so, dass **so wenig Speicher wie möglich** benötigt wird, aber der notwendige Wertebereich abgedeckt wird. Wählen Sie den korrekten Datentyp in Abbildung C-19.1 (nur Einfachantwort möglich).

Variable `hoehe`, Höhe eines Baums (≤ 100)
Variable `name`, Name eines Baums (Maximale Länge 15 Zeichen)
Variable `datum`, Datum der Höhenmessung, codiert in der Form MMYY (MM = Monat, YY = Jahr)
Variable `holzeinschlag`, Holzeinschlag in Festmetern (Gleitkommazahl, < 2000)

```
hoehe          ( )short   ( )float      ( )int      (x)char
name           ( )char    (x)char[16]   ( )string   ( )char[15]
datum          ( )float   (x)short      ( )long     ( )char
holzeinschlag  ( )short   ( )double     (x)float    ( )int
```

Abbildung C-19.1: Antwortalternativen (nur Einfachantwort möglich).

b) Ein- und Ausgabe

Das folgende Codefragment soll die zu einer Uhrzeit gefällten Bäume einlesen (Eingabeformat: XXXX.YY, XXXX = Uhrzeit, YY = Gefällte Bäume). Wählen Sie die korrekte Alternative in Abbildung C-19.2 (nur Einfachantwort möglich), bzw. füllen Sie die Lücke. Alle Variablen sind bereits deklariert.

```
(1)_____("(2)_____", x, y);
```

```
(1) ( ) fprintf
    ( ) print
    (x) scanf
    ( ) printf

(2) %i.%i
```

Abbildung C-19.2: Antwortalternativen für Aufgabe 19 b) (nur Einfachantwort möglich).

c) Boolsche Algebra

Bestimmen Sie das Ergebnis der Ausdrücke in Abbildung C-19.3 im Dezimalsystem. Gegeben sind die folgenden Variablen:

```
int a = 1;
int b = 2;
short c = 8;
int *d = &b;
```

Nach jedem Ausdruck werden die Variablen auf die oben genannten Werte zurückgesetzt.

c.1)	`((a | c) >> *d) + c`	**10**
c.2)	`(c << a) ^ (c / 2)`	**20**

Abbildung C-19.3: Ausdrücke für Aufgabe 19 c)

d) Boolsche Ausdrücke

Schreiben Sie jeweils einen boolschen Ausdruck, der die Aussagen der textuellen Beschreibungen wiedergibt.
Die Variablen `i`, j und k sind bereits definiert.

d.1) `j` ist kleiner als `i` und die Summe aus `j` und `k` ist gerade

```
j < i && ((j + k) % 2) == 0
```

d.2) `k` ist mindestens doppelt so groß wie `i` und `j` ist ganzzahliges Vielfaches von 3

```
(k >= 2 * i) && (j % 3 == 0)
```

Aufgabe 20:
12 Punkte

20. Kontrollstrukturen

Sie sollen ein Programm erstellen, mit dem ein Sägewerk die angelieferten Stämme verwalten kann. Das Programm soll außerdem berechnen, ob das Sägewerk zu den aktuellen Marktpreisen und Kosten einen Gewinn erzielen kann. Die Stämme werden zunächst vermessen und auf Basis des Volumens in Festmetern in Pakete aufgeteilt.

Bitte wählen Sie in den folgenden Aufgaben die korrekte Antwortmöglichkeit (nur Einfachantwort möglich) oder füllen Sie die Lücken.

Sie erstellen das Gesamtprogramm in drei Abschnitten (Teilaufgaben a), b) und c)). Beachten Sie, dass Code, der in einer früheren Teilaufgabe gegeben oder erstellt wurde, auch in den folgenden Teilaufgaben gilt.

a) Sie erstellen zunächst das Codegerüst. Binden Sie die Bibliotheken zur Ein- und Ausgabe und zur Verwendung mathematischer Funktionen (z.B. `pow` zur Berechnung von Potenzen) ein. Definieren Sie außerdem die Konstante `PI` mit einer Präprozessordirektive mit dem Wert `3.1416`.

```
#include _____(1)_____
#include _____(2)_____
 ____(3)____

int main()
{
    //Wird in Teilaufgabe b) implementiert.
    //Wird in Teilaufgabe c) implementiert.
    return 0;
}
```

Abbildung C-20.1: Codefragment für Aufgabe 20 a)

Lücke Nr. 1: `stdio.h`

Lücke Nr. 2: `math.h`

Lücke Nr. 3: `#define PI 3.1416`

b) Holz wird in Festmetern verkauft. Festmeter berechnen Sie mit der Huberschen Formel:

$$V = \frac{\pi}{4} * d^2 * L * 10^{-4},$$

$V = Festmeter,\ d = Durchmesser,\ L = Länge.$

Initialisieren Sie das Array `iBaeume` mit den Daten aus Abbildung C-20.2. Iterieren Sie über alle Werte und summieren Sie die Festmeter. Sobald Sie ein Paket mit 200 Festmetern erreicht haben, erfolgt eine Ausgabe und ein Zähler wird erhöht.
Der Code aus Teilaufgabe a) gilt weiterhin.

	Durchmesser	Länge
1	123	5
2	333	20
3	555	50
4	100	3

***Abbildung C-20.2*:** Maße der angebotenen Stämme

```
float fVolumen = 0;
int iDurchmesser = 0;
int iLaenge = 1, iGesamthoehe = 0, iPakete = 0, i = 0;

int iBaeume[4][2] = _____(4)_____;

for (_____(5)_____)
{
    fVolumen +=
   (PI / 4) * _____(6)_____ * iBaeume[i][1] * _____(7)_____;
    iGesamthoehe += iBaeume[i][0];

    if (fVolumen > 200)
    {
        printf("%i Baeume, %.2f Festmeter Holz\n", i, fVolumen);
        iPakete++;
        _____(8)_____;
    }
}
```

Abbildung C-20.3: Codefragment für Aufgabe 20 b)

Lücke Nr. 4: `{{123, 5}, {333, 20}, {555, 50}, {100, 3}}`

Lücke Nr. 5

```
( ) i = 1; i < 4; i + 1          ( ) i = 1; i < 4; i++
( ) i = 0; i > 4; i--            (x) i = 0; i < 4; i++
( ) i = 0; i < 2; i++            ( ) i = 0; i <= 4; i++
```

Lücke Nr. 6: `pow(iBaeume[i][0], 2)`

Lücke Nr. 7: `pow(10,-4)`

Lücke Nr. 8: `fVolumen = 0`

c) Sie möchten den Ertrag auswerten, den Sie mit dem gelieferten Holz erzielen können. Da Sie in Paketen von 200 Festmetern verkaufen, berechnen Sie den Ertrag auf Basis der Pakete, die Sie in Teilaufgabe b) gezählt haben. Geben Sie den Gewinn als Gleitkomazahl mit 2 Nachkommastellen an.

Beachten Sie die Datentypen und Startwerte der in Abbildung C-20.1 und C.-20.3 deklarierten Variablen. Der Code aus Teilaufgabe a) und b) gilt weiterhin.

```
float fPreisProFestmeter = 12;
float fKostenProFestmeter = 10.12;
int iFestmeter = _____(9)_____;

float fKosten = iFestmeter * fKostenProFestmeter;
float fPreis = iFestmeter * fPreisProFestmeter;

if (_____(10)_____)
{
    printf("Gewinn: __(11)_\n", fPreis - fKosten);
} else
{
    printf("Es wird kein Gewinn erzielt.\n");
}
```

Abbildung C-20.4: Codefragment für Aufgabe 20 c)

Lücke Nr. 9: iPakete * 200

Lücke Nr. 10: fPreis > fKosten

Lücke Nr. 11:

( ) %i	( ) %c	( ) &f
( ) %lf	(x) %.2f	( ) &d

21. Objektorientierte Programmierung

Aufgabe 21:
23 Punkte

a) Grundlagen & Konzepte

Folgend werden grundlegende Konzepte der objektorientierten Programmierung abgefragt. Bitte wählen Sie aus den Antwortalternativen die korrekte Alternative aus (nur Einfachnennung möglich), oder füllen Sie die markierten Lücken mit dem korrekten Begriff.

1. Was trifft *nicht* zu, wenn Klasse B von Klasse A erbt?
( ) Klasse B kann wie Klasse A verwendet werden.
( ) Jede Instanz von B ist auch Instanz von A.
( ) Klasse B kann Methoden von Klasse A überschreiben.
(x) Alle Attribute von Klasse B sind für Klasse A sichtbar.
( ) Klasse B kann die öffentliche Schnittstelle von Klasse A aufrufen
( ) Klasse A kann auf die öffentliche Schnittstelle von Klasse B zugreifen.

2. Instanzen welcher Klasse können auf private Attribute zugreifen?
( ) Instanzen aller Unterklassen der betreffenden Klasse.
( ) Instanzen aller polymorphen Klassen.
( ) Instanzen von aggregierten Klassen.
( ) Keine
( ) Instanzen aller anderen Klassen.
(x) Instanzen der Klasse in der das Attribut definiert wurde.

3. Was beschreibt das Prinzip der Datenkapselung?
( ) Der Zustand eines Objekts kann nicht verändert werden.
( ) Der Zustand eines Objekts darf nur von außen verändert werden.
( ) Alle Zustandsdaten werden in einem eigenen Objekt zusammengefasst.
( ) Ein Objekt muss alle seine Attribute im Konstruktor initialisieren.
(x) Der Zustand eines Objektes kann nur über Methoden verändert werden.
( ) Es kann nicht auf den Zustand eines Objekts zugegriffen werden

4. Was beschreiben die Attribute eines Objekts?
( ) Verhalten
( ) Abstraktion
( ) Schnittstelle
( ) Kommunikation
(x) Zustand
( ) Alle Instanzen

5. Ein Attribut soll nur für Instanzen von abgeleiteten Klassen zugänglich sein. Welches Schlüsselwort verwenden Sie?

protected

6. Handelt es sich bei den folgenden Begriffen um Attribute oder Methoden.

Geschwindigkeit	(x) Attribut	( ) Methode
Schalten	( ) Attribut	(x) Methode

Abbildung C-21.1: Konzepte der objektorientierten Programmierung.

b) Modellierung mit UML

Sie sollen eine Software zur Planung von Möbeln erstellen. Zunächst erfragen Sie von einem Experten die abzubildenden Konzepte. Ein Ausschnitt der Interviewergebnisse, die Sie in ein Klassendiagramm überführen sollen, ist in Abbildung C-21.2 gegeben.

- Die Grundlage sind unterschiedliche Hölzer. Wir verarbeiten entweder Sperrholz oder Massivholz zu Brettern.
- Den unterschiedlichen Hölzern muss immer ein Baum zugeordnet sein, aus dem das Holz überwiegend besteht.
- Ein Möbel ist aus Brettern zusammengesetzt. Wir interessieren uns vor allem für den Preis und das Gewicht der Möbel. Der Preis (Ganzzahl) ist jedem Möbel zugeordnet. Das Gewicht (Ganzzahl) wird durch eine Methode berechnet.

Abbildung C-21.2: Interviewergebnisse, die in ein Klassendiagramm überführt werden.

Füllen Sie die mit Zahlen markierten Lücken im folgenden Klassendiagramm (Abbildung C-21.3). Lücken **zwischen** Klassen erfordern das Einzeichnen von Beziehungen, Lücken **innerhalb** von Klassen erfordern das Einfüllen von Attributen, Methoden oder Klassennamen. Genaue Attribut- und Methodennamen sind **in dieser Teilaufgabe** nicht relevant.

Beachten Sie folgende Konventionen: Attribute werden mit privater Sichtbarkeit und Methoden mit öffentlicher Sichtbarkeit versehen. Verwenden sie, außer wenn dies angegeben ist, ungerichtete Assoziationen.

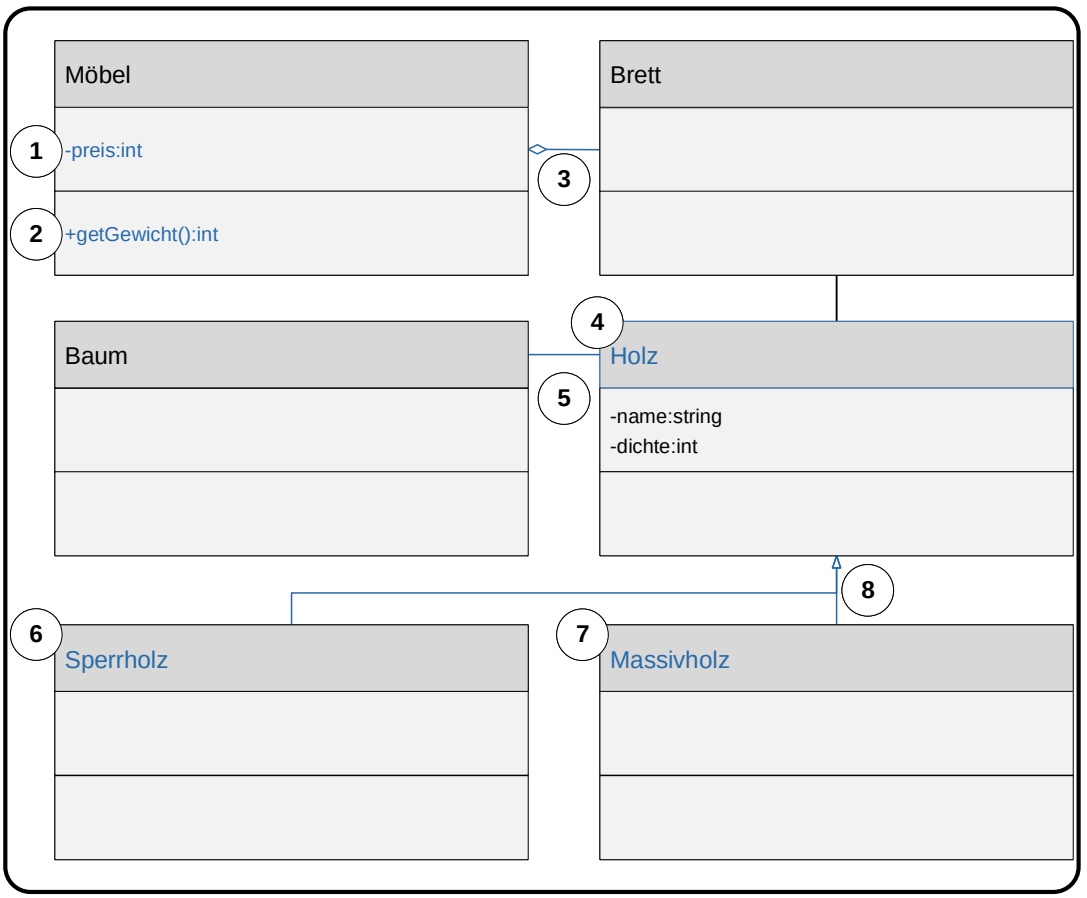


Abbildung C-21.3: Klassendiagramm der Software zur Möbelverwaltung.

c) Vom Code zum Klassendiagramm

Sie haben Programmcode einer Softwarekomponente erhalten, mit der Waldarbeiter und deren Fähigkeiten verwaltet werden. Um die Struktur der Software besser zu verstehen, möchten Sie ein Klassendiagramm des Codeausschnitts in Abbildung C-21.4 erstellen.

Ein Grundgerüst des Klassendiagramms ist in Abbildung C-21.5 gegeben. Die auszufüllenden Lücken sind mit Zahlen markiert.

Lücken **zwischen** Klassen erfordern das Einzeichnen von Beziehungen, Lücken **innerhalb** von Klassen erfordern das Einfüllen von Attributen, Methoden oder Klassennamen.

Achten Sie beim Ausfüllen auf die Datentypen und Namen der Variablen und Parameter, auf die Rückgabewerte der Funktionen, auf die Sichtbarkeiten von Variablen und Methoden sowie die in Teilaufgabe **b)** genannten Konventionen.

```
class Waldarbeiter
{
  public:
    Waldarbeiter(Wald* wald)
  private:
    Schein*[] befaehigungen;
    Wald* wald;
};

class Schein
{
  private:
    string name;
};

class LKW_Schein : public Schein
{};

class Wald
{
  private:
    string sOrt;
    string sName;
};
```

Abbildung C-21.4: Programmcode zur Verwaltung der Waldarbeiter.

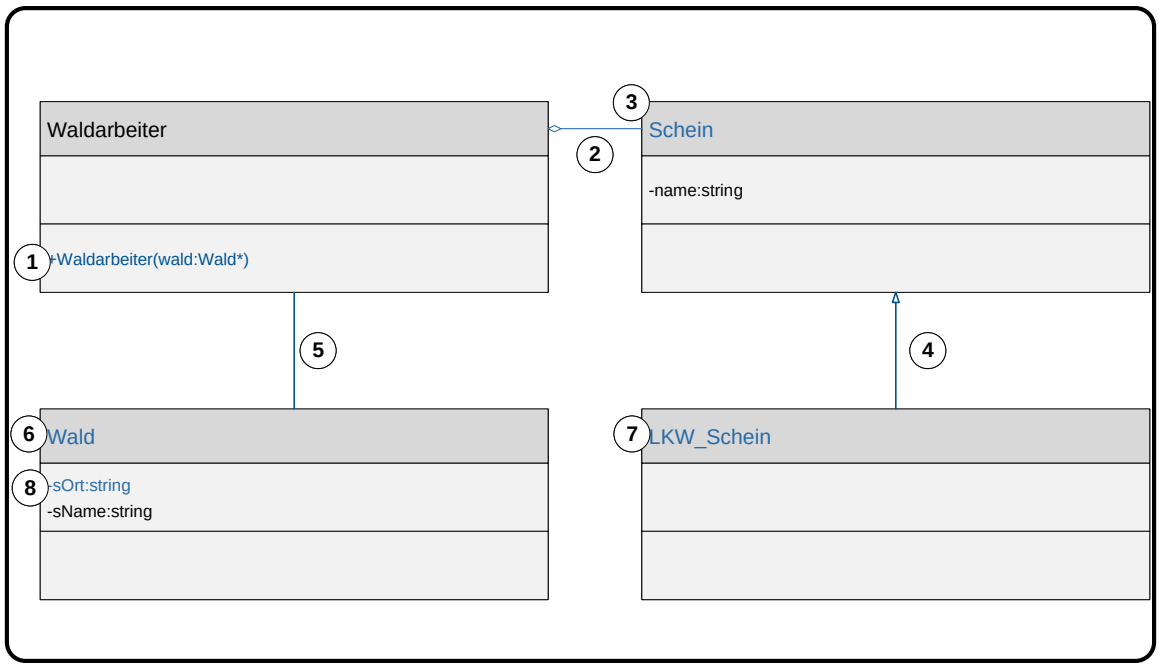


Abbildung C-21.5: Klassendiagramm für die Verwaltung der Waldarbeiter (siehe Abbildung C-21.4).

22. Anlagen/Zustandsautomat

Aufgabe 22:
24 Punkte

Um das Sägewerk aus den vorhergegangenen Aufgaben mit Stämmen zu versorgen, werden sogenannte Holzvollernter (vgl. Abbildung C-22.1) eingesetzt. Diese sind in der Lage, in kürzester Zeit große Mengen an Bäumen zu fällen, zu entasten und vorzusägen. Auf der Erntemaschine ist an einem beweglichen Kranarm der Fällkopf montiert. Dieser besteht aus Entastungsmessern, Walzen zum Klemmen und Vorschub der Baumstämme sowie einer aktivierbaren Kettensäge. In dieser Aufgabe wird lediglich das Programm zur Automatisierung des Fällkopfs betrachtet.

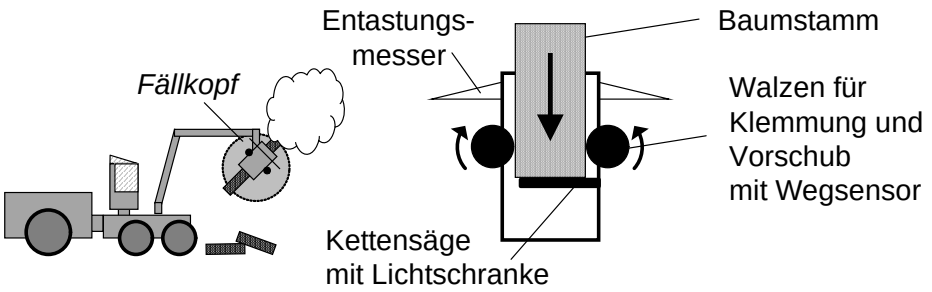


Abbildung C-22.1: Holzvollernter mit Detailansicht Fällkopf.

Folgende Ein- und Ausgänge stehen Ihnen zur Verfügung:

Typ	Name	Beschreibung
AKTOREN	a.EinheitBelegt	Meldet Hauptsteuerung ob Fällkopf belegt (1) oder frei (0) ist.
	a.Saegen	Startet Sägevorgang (1)
	a.WalzenAuf	Öffnet (1) Walzen oder stoppt (0)
	a.WalzenZu	Schließt (1) Walzen oder stoppt (0)
	a.WalzenVor	Rotiert Walzen für Vorschub (1) oder stoppt (0)
SENSOREN	s.WalzenZu	Liefert (1) wenn Klemmung über Walze, sonst (0)
	s.LS	Baumstamm an Kettensäge erkannt (1), sonst (0)
	s.SaegenFertig	Sägevorgang abgeschlossen (1), sonst (0) (Wird zurückgesetzt für jeden Sägevorgang)
	s.Wegmesser	Liefert (1), wenn Vorschub abgeschlossen, sonst (0). (Wird zurückgesetzt für jeden Vorschubvorgang)
VARIABLEN	zeit	Aktuelle Laufzeit des Programms in ms
	t	Variable für timer-Programmierung
	schritt	Aktueller Zustand, welcher ausgeführt wird

Abbildung C-22.2: Sensor- und Aktorvariablen des Fällkopfs, sowie erweiterte Variablen

Zunächst soll der Fällkopf in einen Initial- und Wartezustand gebracht werden. Hierfür müssen die Walzen geöffnet werden, der Vorschub und die Säge werden deaktiviert.

Ab einer Mindestwartezeit von zwei Sekunden im Initial- und Wartezustand (s.o.) und bei einer Detektion eines Baumstamms im Fällkopf (Sensor `LS`) müssen die Walzen geschlossen werden. Sobald die Walzen geschlossen sind (Sensor `WalzenZu`), kann der Sägevorgang beginnen. Hierfür wird das Unterprogramm der Sägeeinheit durch eine Aktivierung des Ausgangs `Saegen` aktiviert. Die Sägeeinheit führt anschließend den Sägevorgang automatisch aus und quittiert einen erfolgreichen Abschluss des Unterprogramms über den Eingang `SaegenFertig`.

Sollte nach Abschluss des Sägevorgangs kein Baumstamm mehr in der Fälleinheit detektiert werden (Sensor `LS`) werden die Walzen wieder geöffnet und zurück in den Initalisierungs- und Wartezustand gesprungen. Der Holzvollernter kann dann einen neuen Stamm aufnehmen.

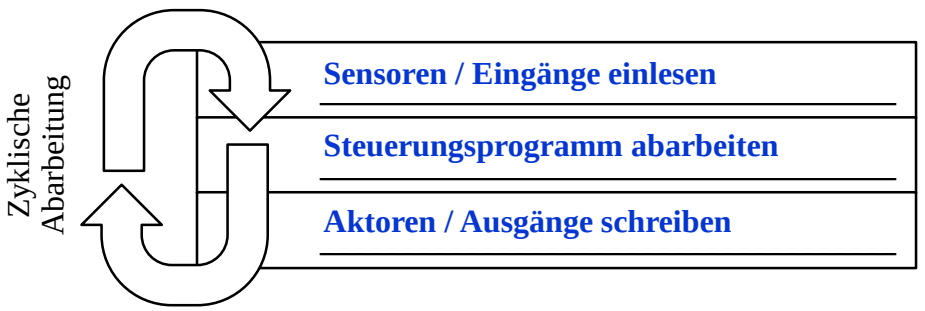
Wird hingegen nach Abschluss des Sägevorgangs weiterhin ein Stamm detektiert, muss dieser für den nächsten Schritt durch den Fällkopf weiterbewegt werden. Hierfür wird der Vorschub der Walzen aktiviert. Ein eingebauter Wegsensor in den Walzen meldet ein Erreichen der definierten Vorschublänge zurück (Sensor `Wegmesser`). Der Vorschub über die Walzen muss dann deaktiviert werden und es wird zum nächsten Schnitt angesetzt.

Bitte beachten Sie, dass Sie stets den aktuellen Belegungsstatus des Fällkopfs über den Ausgang `EinheitBelegt` an das Hauptprogramm des Holzvollernters zurückgeben müssen.

Für die Programmierung einer zeitlichen Verzögerung soll die Hilfsvariable `t` verwendet werden. Die aktuelle Ausführungszeit des Programms ist in `time` gegeben.

a) Wissensfragen

I. Mit welchen Schritten und in welcher Reihenfolge wird allgemein ein Steuerungsprogramm abgearbeitet? Füllen Sie die fehlenden Angaben aus.



II. Wieso können Programmiersprachen wie Java oder C# nicht zur Echtzeit-Programmierung automatisierungstechnischer Systeme genutzt werden?

Nicht deterministisch, zu hoher Ressourcenverbrauch, Garbage Collector

b) Vorgegeben ist das in Abbildung C-22.3 gezeigte Zustandsdiagramm mit den korrespondieren Zustandsnummern. Füllen Sie die durch römische Ziffern und graue Hinterlegung gekennzeichneten Lücken durch Ankreuzen (nur Einfachnennung möglich) bzw. Angabe der Lösung aus.

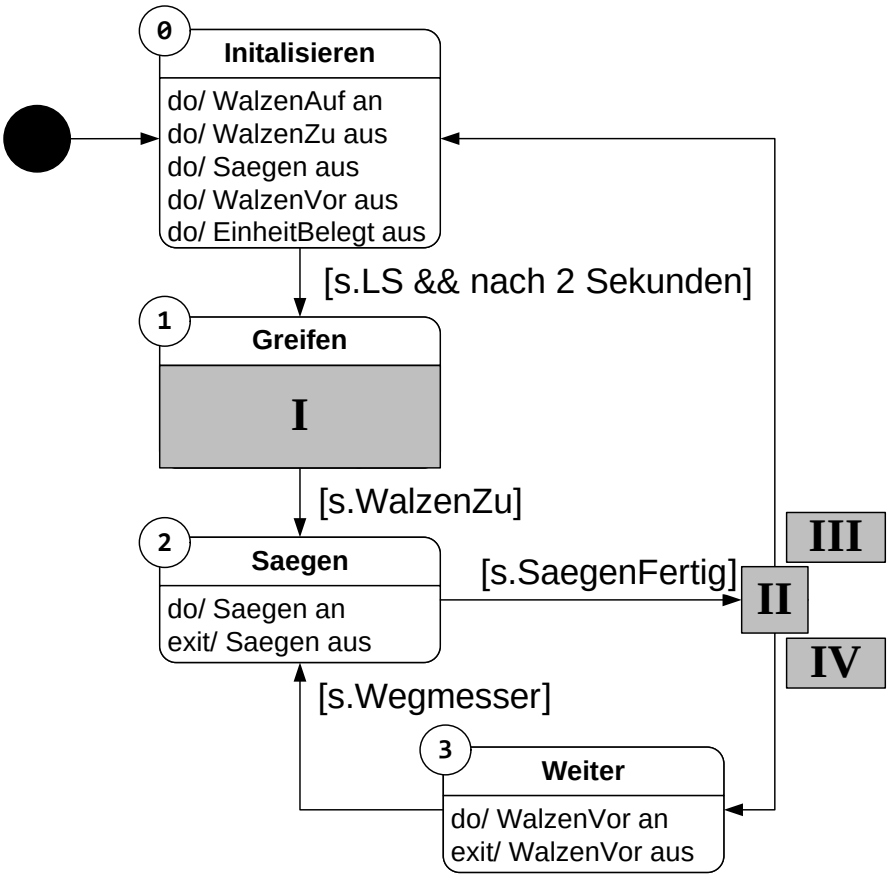
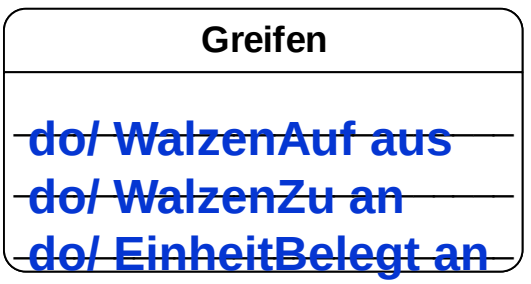
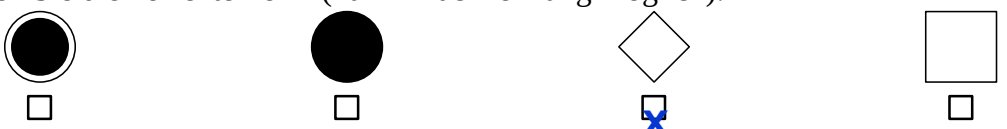


Abbildung C-22.3: Zustandsdiagramm des Unterprogramms für den Fällkopf

I. Modellieren Sie den Zustand `Greifen` korrekt aus.



II. Wählen Sie die korrekte Form (nur Einfachnennung möglich).



☐ ☐ ☒ ☐

III. Geben Sie die korrekte Wächterbedingung an.

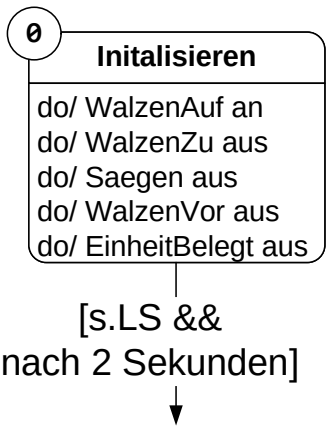
[!s.LS]

IV. Geben Sie die korrekte Wächterbedingung an.

[s.LS]

c) Programmieren Sie im folgenden Antwortfeld (Abbildung C-22.4) den Programmcode für den Zustand `Initialisieren`. Bitte beachten Sie, dass die gegebene Zahl an Leerzeilen nicht der Länge Ihrer Lösung entsprechen muss.

```
case 0    : //Initalisieren        2P
a.WalzenAuf = 1;                   1P
a.WalzenZu = 0;                    1P
a.Saegen = 0;                      1P
a.WalzenVor = 0;                   1P
a.EinheitBelegt = 0;               1P
if (t==0)                          1P
{
    t = zeit;                      1P
}
if (zeit - t >= 2000 && s.LS)      2P
{
    t = 0;                         1P
    schritt = 1;                   1P
}
break;                             1P
```



14P

Abbildung C-22.4: Programmcode für Schritt 0 (Initialisieren).

14P

23. Erweiterte Datenstrukturen und FileIO

Aufgabe 23: 26 Punkte

In dieser Aufgabe schreiben Sie ein Programm, mit dem Sie die Anlieferungen an Ihrem Sägewerk einlesen, verarbeiten und analysieren.

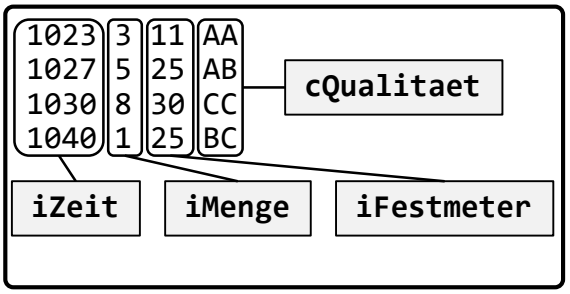


***Abbildung C-23.1*:** Auszug aus der Datei `anlieferungen.txt`.

a) Die aus der txt-Datei (siehe Abbildung C-23.1) gelesenen Messwerte sollen in einer Datenstruktur abgespeichert werden. Die Werte sind durch Leerzeichen getrennt. Diese speichert die Zeit der Anlieferung, die Anzahl an Stämmen samt Festmeter, sowie die Qualität des Holzes.

Erstellen Sie zunächst einen Strukturdatentyp `ANLIEFERUNG`. Ergänzen Sie hierfür den untenstehenden Quelltext (Abbildung C-23.2) in der Headerdatei `datentypen.h` um die notwendigen Typdefinitionen und vervollständigen Sie die Lücken. Verwenden Sie Präprozessordirektiven, um eine mehrfache Einbindung der Headerdatei zu verhindern.

```
#ifndef DATENTYPEN_H_INCLUDED
#define DATENTYPEN_H_INCLUDED
typedef struct
{
    int iZeit;
    int iMenge;
    int iFestmeter
    char cQualitaet[3]
} ANLIEFERUNG;
#endif
```

Abbildung C-23.2: Typdefinition in der Datei `datentypen.h`.

b) Sie sollen nun die Datei `anlieferungen.txt` öffnen und zeilenweise einlesen. Speichern Sie die eingelesenen Daten in das Array `lieferungen`. Deklarieren Sie eine Konstante `LIEFERUNGEN` als Hilfsvariable für das Einlesen auf eine beliebige ganze Zahl. Verwenden Sie die Arraynotation, um die eingelesenen Daten abzuspeichern. Schließen Sie die Datei, nachdem Sie das Einlesen abgeschlossen haben.

Vervollständigen Sie den in Abbildung C-23.3 gegebenen Programmcode.

```
#include <stdio.h>
#include <string.h>
#include "datentypen.h"
#define LIEFERUNGEN 4


int main()
{
    ANLIEFERUNG lieferungen[LIEFERUNGEN];
    ANLIEFERUNG* pAL = lieferungen;
    int i = 0;
    FILE* datei = fopen("anlieferungen.txt", "r");
    for (i = 0; i < LIEFERUNGEN; i++)
    {
        fscanf(datei, "%i %i %i %s",
          &pAL[i].iZeit,
          &pAL[i].iMenge,
          &pAL[i].iFestmeter,
          pAL[i].cQualitaet);
    }
    fclose(datei);
//Wird in Teilaufgabe c) implementiert
```

Abbildung C-23.3: Erster Teil des Programmcodes des Auswerteprogramms.

c) Die Qualität (z.B. hinsichtlich Astigkeit oder Abholzigkeit) der angelieferten Stämme wird durch eine Kombination aus zwei Buchstaben beschrieben. AA, AB und AC bezeichnet Stämme der höchsten Qualität, CC einen Stamm der niedrigsten Qualität. Sie möchten nun berechnen, wie viele Festmeter der Klassen AA, AB und AC Sie erhalten haben.

Hierzu gehen Sie durch Ihre Daten und schreiben die Festmeter getrennt nach Qualität in das Array **quali**. Zum Schluss schreiben Sie die Ergebnisse in eine Textdatei. Die Ausgabe besteht aus den Festmetern der beiden Qualitätsstufen und der Prozentzahl (mit 2 Nachkommastellen) der Stämme in der höchsten Qualität.

Der Code aus Teilaufgabe a) und b) gilt weiterhin.

```
int quali[2] = {0, 0};

for (i = 0; i < LIEFERUNGEN; i++)
{
    if (pAL[i].cQualitaet[0] == 'A')
    {
        quali[0] += pAL[i].iFestmeter;
    } else {
        quali[1] += pAL[i].iFestmeter;
    }
}

FILE* datei2 = fopen("output.txt", "w");
fprintf(datei2, "Beste Qualitaet: %i, %.2f\n",
    quali[0],
    quali[0] / (float) ((quali[0] + quali[1]) * 100));
fprintf(datei2, "Andere Qualitaet: %i", quali[1]);
fclose(datei2);
return 0;
}
```

Abbildung C-23.4: Zweiter Teil des Programmcodes des Auswerteprogramms.