
**Technische Informatik 2**
Sommersemester 2002
Prof. Miroslav Malek

# Brauser

(Die mobile Brauerei)

Virgine Nadine Dsangang Djakou(vdjakou@hotmail.com)
Michael Kreikenbaum(crouching.tiger@web.de)
Andreas Dittrich(andreas.dittrich@berlin.de)
Stefan Gondek(gondek@gmx.net)

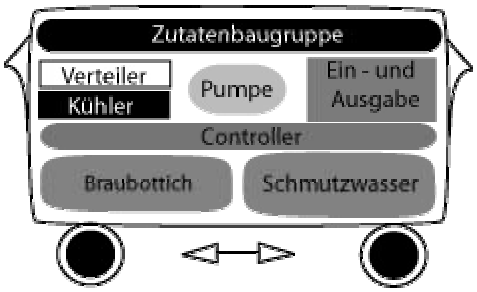
# Inhaltsverzeichnis

## 1. Idee

Es ist Sommer, ein paar Freunde sind zum gemütlichen Beisammensitzen vorbeigekommen. Man geniesst den lauen Abend, spricht über Gott und die Welt und lässt eben jenen einen guten Mann sein. Die Getränkewahl reicht von gemischten Säften über alkoholisierte Longdrinks bis hin zu frisch gezapftem, selbst gebrautem Bier. Der Gastgeber ist Inhaber eines gastronomischen Betriebes? Nein,er hat sich soeben den *Brauser* zugelegt.
Was ist *Brauser*? *Brauser* ist eine mobile Brauerei, Getränkebar und Barkeeper in einem. Und dabei nach den persönlichen Drink Vorlieben einstellbar.Ob Kirsch-Bananensaft, Bloody Mary oder ein sieben-Minuten-Pils. Der kompakte *Brauser* ist in der Lage all jene Getränke auszuschenken und zwar sofort. Das Bier braut er sogar noch selber. Daß er dabei noch mobil ist versteht sich von selbst. Bedarf besteht vieler Orts. Ideal natürlich der Einsatz auf der privaten Gartenparty, beim Camping und eben überall dort, wo viele Leute mit verschiedenen Geschmäckern gemeinsam anstoßen wollen.

## 2. Konzept

Wir planen ein mobiles System um unsere Idee zu realisieren. So sollen alle Baugruppen in einem stabilen Gehäuse untergebracht und außerdem das Gerät mit einer extra langen Stromzuführ versehen sein.




(Abb.2.1: Der *Brauser* als Gerät)

Natürlich ist der *Brauser* auch stationär nutzbar und fügt sich dank seiner Standardmaße nahtlos in jede Küche ein. Für den Benutzer führt diese Funktionsvielfalt nicht in die Unbedienbarkeit. Der *Brauser* hat ein einheitliches Interface.Grundsätzlich lässt ich der Aufbau in drei Zonen aufteilen: Oben wird befüllt.Hier befinden sich 8 Zutatenbehälter, welche mit Säften,Softdrinks, Spirituosen etc befüllen lassen. Die Reinigung gestaltet sich denkbar einfach. Die knapp acht Liter fassenden Kunstoffbehälter sind mit einer Vierteldrehung abgenommen und außerdem spülmaschinenfest. Unten, im Inneren des *Brauser* befindet sich die Brauanlage: ein 16 Liter Bier fassender Braukessel. Ein gutes Pils zapft sieben Minuten, zum Brauen braucht es allerdings etwas länger. Den zwei-wöchigen Brauvorgang inklusive Reinigung überwacht die Kontrolleinheit des *Brauser*, der Benutzer muß lediglich für die Zutaten sorgen.Von da an übernimmt der *Brauser* den Rest und meldet sich erst wieder, sobald das Bier fertig gebaut auf den Ausschank wartet.

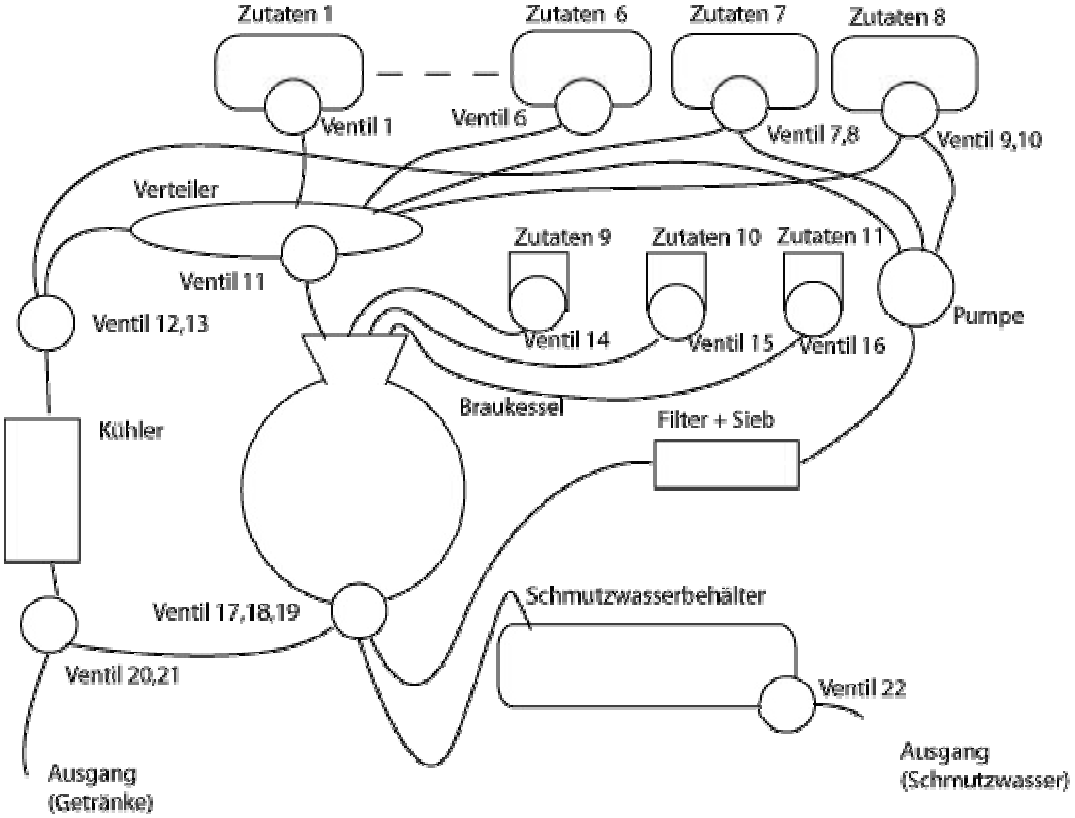
Diese Miniaturausgabe besitzt dabei alle Vorteile einer großen Brauerei.Im Gegensatz zu den “Brausäcken” aus dem Getränkemarkt bewahrt der Kupferkessel den typischen Pilsgeschmack, das Bier wird sogar noch gefiltert bevor es ausgeschenkt werden kann.
Vorne liegt das Kontrollzentrum: Nachdem die Party vorbereitet wurde, was in unserem Falle heißt, daß alle gewünschten Getränkezutaten aufgefüllt wurden, ist der *Brauser* startklar. Neben dem selbstreinigenden Zapfhahn, durch den alle Getränke fließen, befindet sich das Display. Hier kann der Benutzer den Drink auswählen, der dann sofort ausgeschenkt wird. Außerdem lassen sich über das Display eigene Getränkekreationen neben den voreingestellten speichern.
Der Nutzer soll seine Getränke gut gekühlt erhalten, deshalb enthält *Brauser* eine Kühlung. Da es sich um kein System handelt, daß in Echtzeit Daten verarbeiten muß, haben wir uns für eine Arbeitsfrequenz von 1KHz entschieden. Diese Frequenz stellt die geringsten Anforderungen an Baugruppen und lässt auch keine Probleme mit den verwendeten Geräten aufkommen (Frequenzabhängigkeit).
Wenn *Brauser* arbeitet, verläuft ein typischer Mischvorgang folgendermaßen:

(1) Der Bediener hat alle Zutatenbehälter entsprechend der Rezepte, die Verwendung finden sollen, aufgefüllt.
(2) Der Bediener hat die zur Anwendung kommenden Rezepte (auf EEPROM) in den *Brauser* geladen.
(3) Der Bediener hat den *Brauser* eingeschaltet.
(4) Der Bediener wählt aus dem Menu mit Hilfe der Anzeige ein Programm.
(5) Der Bediener gibt die Anzahl der gewünschten Getränke ein.
(6) Der Bediener kann nun (4) – (5) beliebig oft wiederholen.

# 3. Geräte in der Übersicht

Im folgenden sollen alle im *Brauser* vorkommenden Baugruppen vorgestellt werden. Auch soll auf deren Funktion kurz eingegangen werden.
Die unten gezeigte Abbildung soll dem Leser einen Eindruck vom Zusammenwirken der Baugruppen geben.

Der *Brauser* ist ca. 1,3 m breit, 0,6 m tief und 1,2 m hoch.



(Abb.3.1: Bezeichnung der Ventile und Baugruppen)

## 3.1 Zutatenbehälter

Alle Zutaten, die für die verschiedenen Rezepte benötigt werden, kommen über die Zutatenbehälter in das System. Zusätzlich werden für den technologischen Prozess des Bierbrauens noch drei Feststoffbehälter benötigt. Diese Behälter sind in Form von Schubladen gestaltet. In diese Schubladen werden dann Hopfen (in Form von Pellets), Hefe und Malz gegeben.

(Abb.3.1.1: Zutatenbehälter, hier nur die Anordnung der Aufnahmen)

*Funktion:*
Im *Brauser* selbst sind die die Aufnahmen für die Zutatenbehälter fest installiert (siehe Abb.). Sie bilden das Gegenstück zu den Behältern: Am unteren Ende befindet sich ein Drehmechanismus, welcher den Zutatenbehälter öffnet.
Dieses Bauteil kann elektrisch nicht manipuliert werden.

Die Zutatenbehälter sind aus hygienischen Gründen auswechselbar. Um Probleme mit kohlensäurehaltigen Zutaten zu vermeiden setzen wir, wie bereits erwähnt, das System unter Druck.


*Funktion:*
Wie bereits im Konzept beschrieben dient ein Behälter aus Kunststoff zum Einbringen der flüssigen Zutaten in das System. Er verfügt über zwei Öffnungen. Am unteren Ende liegt ein Systemverschluß, der beim Adaptieren öffnet. Oben ist eine Drucklufteinspeisung (mechanisches Ventil).
Der Einsatz ist ca 20x20 cm groß und fasst ca. 8 Liter Flüssigkeit.
Das Bauteil verfügt über keine elektrische Funktionen.

(Abb. 3.1.2: Zutateneinsatz mit Druckeinspeisung)

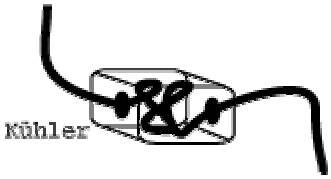
## 3.2 Ventile

Aus den Zutatenbehältern werden über Schläuche die Basisflüssigkeiten zum Verteiler gelenkt. Ein Ausnahme bilden dabei die Behälter 7 und 8, die zusätzlich direkt mit einer Pumpe verbunden sind, um das Bier aus dem Braubottich zurückzupumpen. In diesem Fall werden diese Behälter am Anfang für Brauwasser und während des Brauprozesses als temporäre Behälter genutzt.

Die Ventile sind einzeln steuerbar. Über den im Speicher an entsprechender Adresse abgelegten Wert ( 0 – Zu oder 1 – Auf) werden sie geöffnet und geschlossen.

Untereinander und mit allen beteidigten Baugruppen sind die Ventile über handelsübliche Plastikschläuche verbunden (siehe Abb. "Bezeichnung der Ventile und Baugruppen, Seite 5").

## 3.3 Kühlung

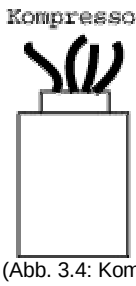



*Funktion:*
Die Kühlung wird mit dem System eingeschaltet und läuft außer beim Bierbrauen immer. Wir haben uns für einen Spiralkühler entschieden. Dieser bezeichnet in diesem Fall ein gewundenes Rohr, das in einer Kühlbox liegt. Die Kühlung ist mit dem Einschalten funktionsbereit.

(Abb. 3.3: Kühlung)

Beim technologischen Ablauf des Bierbrauens ist es beim Übergang vom Hopfen zum Lagern nötig, den Sud abzukühlen, um die Hefe hinzuzufügen. Nach Aussagen eines erfahrenden Bierbrauers (Berliner Schultheiss Brauerei) muß dies nicht zwingend schnell erfolgen. Auch reicht eine Temperatur von ca. 30 Grad, um die Hefe einzubringen. Da das Bierbrauen in unserem Konzept nicht im Vordergrund steht, erlauben wir es uns, hier eine Wartezeit einzuräumen, bis der Sud langsam durch den Kühler gelaufen ist. Ausserdem kommt es bei einer Brauzeit von 14 Tagen (bei Lagerbier) auf einige Stunden mehr nicht an. Da die zu verarbeitende Menge nicht so groß ist, entsteht keine unzumutbare Wartezeit.

## 3.4 Kompressor

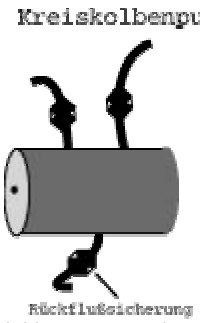



*Funktion:*
Das Gerät arbeitet weitestgehend unabhängig vom Gesamtsystem und regelt sich über den Druck an einem Ausgleichzylinder. Dabei lassen wir einen Toleranzbereich von 0.5 Bar zu. An diesem Ausgleichszylinder befinden sich zudem noch Schläuche die mit Schnappventilen versehen sind (über die Verteilerkappe). Der Kompressor läuft ab dem Einschalten des *Brauser*.

(Abb. 3.4: Kompressor)

Da wir nicht ausschließen können, daß sich in unserem System auch Zutaten mit Kohlensäure befinden, setzen wir das Gesamtsystem unter Druckluft. Der Kompressor ist ein Kolbenkompressor aus Plastik, der das System mit dem nötigen Druck versorgt. Beim Befüllen des Systems muss der Nutzer als letzte Handlung den Schlauch vom Kompressor auf jeden Zutatenbehälter stecken. Die Druckluft wird dann über das Öffnen eines Ventils (manuell) in das System eingespeist.

## 3.5 Pumpe



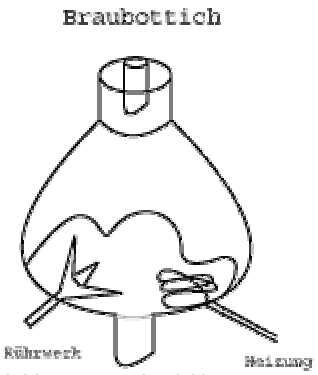

*Funktion:*
Die Pumpe wird zum Umwälzen des Suds benötigt. Es handelt sich dabei um ein handelsübliches System mit zwei Ausgängen und einem Eingang. Die Pumpe ist mit dem Einschalten betriebsbereit.

(Abb. 3.5: Pumpe)

Aus Gründen der Rationalisierung haben wir uns entschlossen, zwei der Zutatenbehälter mit einem Spezialventil auszurüsten und dieses mit der Pumpe zu koppeln. Um nun den technologischen Prozess beim Bierbrauen besser kontrollieren zu können, nutzen wir diese beiden Zutatenbehälter als temporäre Gefäße. Die Pumpe muß so dimensioniert sein, daß sie zum einen in der Lage ist, gegen den Druck im Zutatenbehälter anzukommen und zum anderen die Maische (Gerste mit Hopfen) durch einen Filter in diese Behälter pumpen kann. Zwei Behälter (Nr. 7 und 8) sind dabei parallel geschaltet. Mit diesem System sparen wir uns einen zweiten Kupferkessel zum maischen und hopfen.

## 3.6 Der Braubottich

Ist ein aus Kupfer bestehender Behälter, der mit einer Heizung und einem Rührwerk versehen ist. Heizung und Rührwerk sollen voneinander unabhängig aktiviert und deaktiviert werden. Darüber hinaus verfügt dieses Gerät über einen Auslass, der ebenfalls separat gesteuert wird.



(Abb. 3.6: Braubottich)

*Funktion:*
Im *Brauser* ist der Kessel zum Bierbrauen enthalten. Am oberen und unteren Ende befinden sich Öffnungen, die mit Ventilen versehen sind. Am unteren Ende befinden sich drei Ventile, am oberen Ende liegen drei Einlassmöglichkeiten. Der Braubottich verfügt über einen Füllhöhenanzeiger. Dieser Anzeiger liefert eine '1' wenn die maximale Füllhöhe erreicht ist, ansonsten eine '0'.
Der Braubottich fasst ca. 10 Liter.
Das Gerät ist mit dem Einschalten betriebsbereit.
Das Rührwerk und die Heizung werden separat vom IO-Controller aktiviert.

## 3.7 Schmutzwasserbehälter

Der Schmutzwasserbehälter dient ausschließlich der Aufnahme von Flüssigkeit, die bei der Reinigung anfällt; er ist austauschbar im Montagerahmen eingebaut (hygienischer). Der Behälter verfügt über einen Schwimmer, der die aktuelle Befüllung erfasst. Dabei sind, wie beim Braubottich, nur der Zustand *voll* oder *ok* relevant.
Der Schmutzwasserbehälter ist mit dem Einschalten der Betriebsspannung aktiv.

## 3.8 Feststoffbehälter

Um das zusätzliche Feature 'Bierbrauen' in das Konzept aufnehmen zu können, ist es notwendig, drei Behälter für Feststoffe wie Brauhefe, Hopfen und Malz vorzusehen. Diese Behälter sind als Schubladen in der Front angebracht. Wir erfassen bei dieser Baugruppe keine Befüllung. Wenn der Nutzer Bier braut, sollten diese Behälter auch befüllt sein.
Mit dem Einschalten ist diese Baugruppe aktiv.

## 3.9 Filterbaugruppe

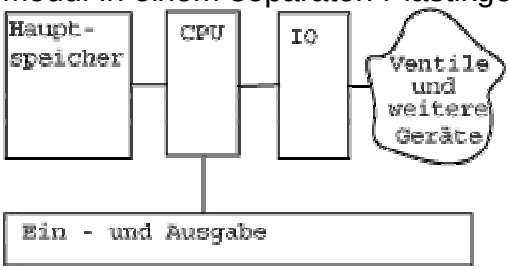
Ebenfalls um das Bierbrauen zu ermöglichen ist es notwendig, einen Filter in das System einzubauen. Nach der Enzymierungsphase (maischen) muß der Sud gefiltert werden. Dieser Filter sollte leicht auswechselbar sein, da die Maische relativ grobkörnig ist, der Filter also oft gereinigt werden muss. Er darf nicht zu feinporig sein, da ansonsten zu viele Schwebstoffe mit in das Bier geraten. Unsere Idee ist es, eine Kassette zu bauen, die oben und unten aus groben Metallsieben besteht und im Inneren - auswechselbar - mehrere Lagen Filterstoff enthält. Es wird bei diesem Bauteil kein Meßwert erfaßt. Wir gehen auch hier davon aus, daß der Nutzer den Filter nach dem Brauen selbst reinigt.
Diese Baugruppe ist mit den Einschalten aktiviert.

## 4. Die zentrale Steuereinheit

Die zentrale Steuereinheit ist innerhalb des *Brausers* installiert.

Die Stromversorgung erfolgt über das 230 V Stromnetz. Wir verwenden dabei eine extra lange Zuleitung um die Mobilität zu gewährleisten. Ein internes Netzteil versorgt dann alle Geräte mit +5 Volt Gleichspannung.

Dieses Gerät ist extra gegen Feuchtigkeit und Schmutz gesichert. Wir denken daran, das Modul in einem separaten Plastikgehäuse zu verpacken.




(Abb. 4.1: Steuereinheit und deren Zusammenwirken mit anderen Baugruppen)

Nach Anschluß der Versorgungsspannung erfolgt der Start des Microcodes im ROM-Speicher ab der Adresse 0 automatisch. Das Hauptprogramm ist eine Endlosschleife, die auf Eingabe wartet. Als mögliche Eingaben kommen Speicheradressen als Unterprogramme in Frage (siehe 6.1). Diese Unterprogramme sind jeweils in einem separaten Speicherbaustein untergebracht und können modular eingesetzt werden. Angesprungen wird die Adresse der Speicherbank; ist dort kein Modul vorhanden erfolgt automatisch der Rücksprung in die Eingabeschleife.

Wir verwenden einen 12-Bit Mikroprozessor, der preisgünstig im Handel zu beziehen ist. Ein Programm (mitgeliefert) arbeitet eine Schleife in ca. 1000 Takten ab. Die Überlegung: Wenn ein Schleifendurchlauf ca. 1 Sekunde dauert, haben wir für die Abfrage der nötigen Meßwerte und für die Ausgabe ebenfalls 1 Sekunde Zeit. Es ergibt sich somit ein sinvoller Prozessortakt von 1 Khz.

Daten- und Prozessorbus sind zwölf Bit lang. Für den Zugriff auf den I/O-Controller und alle anderen Geräte am externen Bus wird eine Wortlänge von zwölf Bit bereitgestellt.

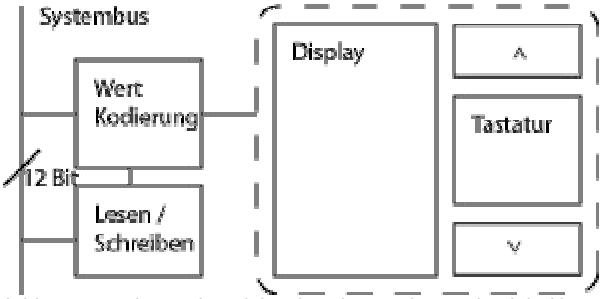
# 5. Braugruppen

## 5.1 I/O-Controller

Dieses Gerät dient dazu, alle Ventile, die Pumpe, die Heizung, das Rührwerk, den Ausgang und den Kühler zu steuern. Alle aufgezählten Geräte besitzen eine einfache an/aus-Steuerung und werden von diesem Gerät bedient. Darüber hinaus ermittelt dieses Gerät den Überlauf für den Schmutzbehälter und den Braubottich.
Wir entscheiden uns für diese Baugruppe aus Kostengründen. Da alle Geräte ausschließlich ein- bzw. auszuschalten sind oder einen Wert voll bzw. leer liefern, können diese effektiver von einem I/O-Controler verwaltet werden. Eine separate Steuerung für jedes Gerät über den Bus erfordert einen höheren Aufwand für den Bus und erhöht somit die Kosten. Der I/O-Controller bezieht aus dem externen Bus den Wert für ein zu öffnendes oder zu schließendes Ventil, für das an- und ausschalten der Heizung, der Kühlung und des Rührwerks. Letzteres ist ein wichtiger Bestandteil, wenn es um das Bierbrauen geht. Die Sensoren für Temperatur bzw Überlauf werden ebenfalls von diesem Gerät verwaltet.

## 5.2 Eingabe

Der Nutzer soll mit unserem System die Möglichkeit haben aus einer Vorgabe von Getränkerezepten wählen zu können. Dazu wählt er über dieses Gerät aus einer Menge von Programmen. Wie bereits erwähnt ist diese Menge durch programmierten Speicher, der in das System eingesteckt wurde, bestimmt. Darüber hinaus wird über die Eingabe die Anzahl der Getränke bestimmt.




Das Gerät ist aktiv wenn folgende Parameter erfüllt sind:
(1) Der *Brauser* ist angeschlossen (elektrisch).
(2) Der *Brauser* bearbeitet kein anderes Programm.
(3) Der Nutzer aktiviert die Eingabe mit einem Druck auf irgendeinen Knopf.
Mit dem Einschalten ist dieses Gerät betriebsbereit.

(Abb. 5.2: Design und Funktion der Ein – und Ausgabeeinheit)

Der Nutzer erhält von der Beriebsbereitschaft durch einen kurzen Piep Kenntnis. Zum Lieferumfang von *Brauser* gehört eine Tabelle, in der die mitgelieferten Rezepte durch ihren Zahlencode repräsentiert werden (als Beispiel).

## 5.3 Ausgabeeinheit

Dieses Gerät liefert den aktuellen Status des *Brauser* und gibt aufgetretene Fehler aus. Wir haben uns der Einfachheit halber dafür entschieden mit Fehlercodes zu arbeiten.

| ***Fehler*** | ***Wert*** | ***Ausgabe*** |
|---|---|---|
| Kein Fehler | #0 | OK |
| Falsche Eingabe | #1 | ERROR 1 |

| *Fehler* | *Wert* | *Ausgabe* |
|---|---|---|
| Schmutzwasser voll | #2 | ERROR 2 |
| Braubottich voll | #3 | ERROR 3 |
| Programm am Ende | #4 | ENDE PROG |
| Reinigung am Ende | #5 | ENDE CLEAN |

(Tabelle 5.3: Ausgaben)

Im wesentlichen empfängt das Gerät alles, was an seine Adresse gesendet wird und stellt es in einer für den Nutzer sichtbaren Form dar. Mit Hilfe einer mitgelieferten Tabelle kann der Nutzer den Fehler feststellen. Aus Kostengründen ist dieses Gerät zusammen mit der Eingabeeinheit in einem Modul untergebracht.

## 5.4 Temperatursensor

Dieser Sensor erfasst die Temperatur im Braubottich. Die vom Gerät gelieferten Daten werden vom I/O-Controller verarbeitet (quantisiert) und werden in einem Speicherblock abgelegt.
Es sollen Temperaturänderungen von 2° C eine Änderung des Datenworts bewirken. Erfasst wird der Bereich zwischen 20° und 90° C. Bei einer Genauigkeit von 2° C ergibt das
90 – 20 = 70 / 2 = 35 Werte, die sich in ld ( 35 ) = 5,13 => 6 Bit Kodieren lassen.

## 5.5 Hauptspeicher

Der Hauptspeicher soll halbwort-adressierbar sein. Da alle Gerätecodes, Steuerwerte und Meßwerte jeweils in sechs Bit passen.

Wir wählen eine max. Speichergröße von 24 Kbit, was 24 * 1024 / 6 = 4096 Halbworten entspricht. Daraus gibt sich, wegen $2^{12} = 4096$, eine 12 Bit Adressierung.

Für die ersten sechs KBit verwenden wir ROM-Speicher, z.B. EPROM-Speicher. Im ROM-Speicher befinden sich der Microcode, das Hauptprogramm (siehe 6.1) und das Reinigungsprogramm (siehe 6.3). Der Mikrocode befindet sich ab Adresse 0 und wird beim Anlegen der Versorgungsspannung automatisch ausgeführt (ab Adresse 0). Der *Brauser* befindet sich dann in einer Endlosschleife die auf Eingabe wartet.

Ab Adresse #1024 folgen 6 KBit RAM – Speicher die der Aufnahme der Gerätewerte dienen

Für den Speicher ab Adresse #2048 ist die Form eines Memorychips denkbar. Ab dieser Adresse sollen dann Rezepte (austauschbar) liegen. Neue Programme können so separat auf einem Computer geschrieben und dann in den *Brauser* eingesetzt werden. Diese erste Bank hat eine feste Adresse, welche aus der Hauptschleife angesteuert wird. Ab Adresse #2048 liegt dann das erste Unterprogramm, welches abgearbeitet werden kann. Alle folgenden Rezepte liegen dann 1024 Adressen weiter. Das heißt, das nächste Programm liegt bei #3072 und das letzte bei #4096.

Ein erweitertes Konzept ist denkbar: So könnten Rezepte auch aufeinanderfolgend addresiert werden (z.B. #2048 – Rezept 1, #2049 – Rezept 2, usw.). Diese Unterprogramme laufen dann aber in einem Speicherbereich, der für das System nicht abbildbar ist. Dies bedeutet, daß das Unterprogramm erst in freien Speicher kopiert werden muß. Dieser Aufwand ist jedoch mit höheren Kosten verbunden.

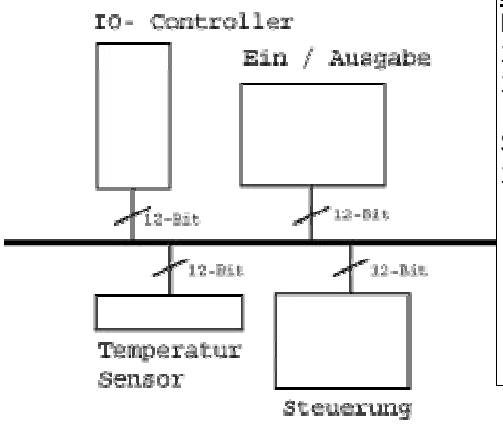
Für jedes Gerät wird der aktuelle Status (an oder aus) im Speicher hinterlegt. Für die Temperaturerfassung gilt: Der aktuelle Wert im Wertebereich wird erfasst und gespeichert.

| ***Label*** | ***Beschreibung*** | ***Wertebereich*** | ***Adresse*** |
|---|---|---|---|
| V1OnOff | Ventil 1 auf/zu | 1;0 | #1024 |
| V2OnOff | Ventil 2 auf/zu | 1;0 | #1025 |
| V3OnOff | Ventil 3 auf/zu | 1;0 | #1026 |
| V4OnOff | Ventil 4 auf/zu | 1;0 | #1027 |
| V5OnOff | Ventil 5 auf/zu | 1;0 | #1028 |
| V6OnOff | Ventil 6 auf/zu | 1;0 | #1029 |
| V7OnOff | Ventil 7 auf/zu | 1;0 | #1030 |
| V8OnOff | Ventil 8 auf/zu | 1;0 | #1031 |
| V9OnOff | Ventil 9 auf/zu | 1;0 | #1032 |
| V10OnOff | Ventil 10 auf/zu | 1;0 | #1033 |
| V11OnOff | Ventil 11 auf/zu | 1;0 | #1034 |
| V12OnOff | Ventil 12 auf/zu | 1;0 | #1035 |
| V13OnOff | Ventil 13 auf/zu | 1;0 | #1036 |
| V14OnOff | Ventil 14 auf/zu | 1;0 | #1037 |
| V15OnOff | Ventil 15 auf/zu | 1;0 | #1038 |
| V16OnOff | Ventil 16 auf/zu | 1;0 | #1039 |
| V17OnOff | Ventil 17 auf/zu | 1;0 | #1040 |
| V18OnOff | Ventil 18 auf/zu | 1;0 | #1041 |
| V19OnOff | Ventil 19 auf/zu | 1;0 | #1042 |
| V20OnOff | Ventil 20 auf/zu | 1;0 | #1043 |
| V21OnOff | Ventil 21 auf/zu | 1;0 | #1044 |
| V22OnOff | Ventil 22 auf/zu | 1;0 | #1045 |
| POnOff | Pumpe an/aus | 1;0 | #1046 |
| HOnOff | Heizung an/aus | 1;0 | #1047 |
| ROnOff | Rührwerk an/aus | 1;0 | #1048 |
| WarnS | Überlauf Schmutzwasser | 1;0 | #1049 |
| WarnB | Überlauf Braubottich | 1;0 | #1050 |
| TempB | Temperatur Braubottich | +20 .. +90 | #1051 |
| InpIO | Eingabe | 0 – 7 | #1052 |
| OutIO | Ausgabe | 0 – 7 | #1053 |
| LstErr | Letzter Fehler | 0 - 5 | #1054 |

(Tabelle 5.5: RAM Speicherbelegung bezüglich der Baugruppen)

## 5.6 Bus

Der Bus ist hauptsächlich für die Verbindung zwischen den Geräten zuständig. Er ist mit einer Breite von zwölf Bit so dimensioniert, daß alle möglichen Gerätekodierungen und Meßwerte in ein Datenwort passen.




*Funktion:*
Prozessorbus:
12 – Bit Adressbus (nicht im Bild)
12 – Bit Datenbus (nicht im Bild)

Systembus:
12 – Bit Länge, dient der Kommunikation mit der Peripherie

(Abb.5.6 Bus)

## 5.7 Mikroprozessor

Wir haben uns in unserem Projekt für eine zwei-Adress-Maschine entschieden.
Der Prozessor hat vier Register (R0 .. R3), vier Adressierungsarten (Register, Register indirekt, Immediate [Operand folgt Anweisung], Direkte Adressierung) und einen vier-Bit-Opcode.

Durch geschicktes Ausnutzen der arithmetischen Befehle versuchen wir den Befehlssatz zu reduzieren. So verwenden wir zur Zeitverzögerung nicht einen gesonderten Befehl sondern addieren nur Nullen miteinander.

Im Adressierungsmodus *Immediate* und *Direkte Adressierung* sind die Bits 6 – 7 und 10 – 11 unbesetzt.

| ***Opcode*** | ***Src Adress Mode*** | ***Src Register*** | ***Dst Adress Mode*** | ***Dst Register*** |
|---|---|---|---|---|
| 4 Bit | 2 Bit | 2 Bit | 2 Bit | 2 Bit |
| 0 3 | 4 5 | 6 7 | 8 9 | 10 11 |

(Tabelle 5.7.1: Befehlsformat)

Für Sprungbefehle gilt:

| ***Opcode*** | ***Near Sprung Adresse*** |
|---|---|
| 4 Bit | 8 Bit |

(Tabelle5.7.2: Sprungsbefehle)

Der Immediate Operand ist zusammengesetzt aus:

| 12 Bit Wort | |
|---|---|
| 6 Bit Halbwort | 6 Bit Halbwort |

(Tabelle5.7.3: Zusammensetzung Immediate Operand)

Für die Direkte Adressierung folgt:

| 12 Bit Adresse |
|---|

(Tabelle5.7.4: Direkte Adressierung)

Die Adressierungsarten kodieren wir wie folgt:

| ***Adressierungsart*** | ***Kodierung*** | ***Beispiel*** |
|---|---|---|
| Register direkt | R0 .. R3 | MOVE R0, R1 |
| Register indirekt | @R0 .. @R3 | MOVE @R0, @R1 |
| Immediate | #31 (Dezimal) | MOVE #31, R1 |
| Direkte Adressierung | [2048] | MOVE [2048], R1 |

(Tabelle5.7.5: Adressierungsarten)

Die ALU beherrscht die Operationen *plus* und *minus*. Für unseren *Brauser* ist keine Implementation eines Multiplikations- oder Divisionsalgorithmus nötig.Generell rechnen wir mit dem Zweierkomplement, um Schwierigkeiten mit der Null aus dem Weg zu gehen.

| ***1. Operand*** | ***2. Operand*** | ***Operation*** | ***Flag G (größer Null)*** |
|---|---|---|---|
| A | B | A + B | G = 1, wenn A + B > 0, sonst G = 0 |
| A | B | A – B | G = 1, wenn A – B > 0, sonst G = 0 |

(Tabelle5.7.6: ALU Operationen)

## 5.7.1 Befehlssatz

| ***Typ*** | ***Befehl*** | ***Beschreibung*** |
|---|---|---|
| Datentransfer | MOVE Src, Dst | Dst <= Src #Transferiert ein Wort von Src nach Dst |
| Arithmetik | ADD Src, Dst | Dst <= Dst + Src #Summe der beiden Operanden |
| | SUB Src, Dst | Dst <= Dst – Src #Differenz zweier Operanden |
| Ein- /Ausgabe | IN Gerätecode, Wert | |
| | OUT Gerätecode, Wert | |
| Sprung | JUMP Adresse | Springe zur Adresse, unbedingter Sprung, lädt PC mit Adresse |
| Sprung | JGT Adresse | - Springe zur Adresse, wenn Flag G = 1<br>- ist G = 1 erfüllt lade PC mit Adresse, sonst mache nichts |

(Tabelle: 5.7.1.1: Befehlssatz)

## 5.7.2 Microcode

Allgemein gilt:

Am Start:
PC <= L, wobei L das Label für den ersten Befehl des Betriebsmodus ist

Für alle Befehle außer dem Sprungbefehl gilt:
T1: PC out, MAR in, READ, PC <= [PC] + 2 #der PC erhöht sich unabhängig von der ALU
T2: MDR out, IR in, Befehl dekodieren

(Abb.5.7.2.1: Mikrocode 1)

## Der MOVE Befehl (MOVE Src, Dst):

```
T3: SWITCH (IR <4:5>) #Src Adress Mode
    CASE (Dst Adress Mode = <Register direkt>):
            SWITCH (IR <8:9>) #Dst Adress Mode
            CASE (Dst Adress Mode = <Register direkt>):
                    Reg[IR <6:7>] in, #Src Register
                    Reg[IR <10:11>] out #Dst Register
            CASE (Dst Adress Mode = <Register indirekt>):
                    Reg[IR <10:11>] in, MAR out,
T4:                 Reg[IR <6:7>] in, MDR out, WRITE,
            CASE (Dst Adress Mode = <Direkte Adressierung>):
                    PC in, MAR out, PC <= PC + 2, READ
T4:                 MDR in, MAR out,
T5:                 Reg[IR <6:7>] in, MDR out, WRITE
    CASE (Src Adress Mode = <Register indirekt>):
            SWITCH (IR <8:9>) #Dst Adress Mode
            CASE (Dst Adress Mode = <Register direkt>):
                    Reg[IR <6:7>] in, MAR out, READ,
T4:                 MDR in, Reg[IR <10:11>] out #Dst Register
            CASE (Dst Adress Mode = <Register indirekt>):
                    Reg[IR <6:7>] in, MAR out, READ, #Daten im MDR
T4:                 Reg[IR <10:11>] in, MAR out, WRITE
            CASE (Dst Adress Mode = <Direkte Adressierung>):
                    Reg[IR <6:7>] in, MAR out, READ,
T4:                 MDR in, H1 out, #H1 = Hilfsregister 1
T5:                 PC in, MAR out, PC <= PC + 2, READ,
T6:                 MDR in, MAR out,
T7:                 H1 in, MDR out, WRITE
    CASE (Src Adress Mode = <Immediate>):
            SWITCH (IR <8:9>) #Dst Adress Mode
            CASE (Dst Adress Mode = <Register direkt>):
                    PC in, MAR out, PC <= PC + 2, READ,
T4:                 MDR in, Reg[IR <10:11>] out #Dst Register
            CASE (Dst Adress Mode = <Register indirekt>):
                    PC in, MAR out, PC <= PC +2, READ, #Daten im MDR
T4:                 Reg[IR <10:11>] in, MAR out, WRITE
            CASE (Dst Adress Mode = <Direkte Adressierung>):
                    PC in, MAR out, PC <= PC + 2, READ,
T4:                 MDR in, H1 out
T5:                 PC in, MAR out, PC <= PC + 2, READ,
T6:                 MDR in, MAR out,
T7:                 H1 in, MDR out, WRITE
    CASE (Src Adress Mode = <Direkte Adressierung>):
            SWITCH (IR <8:9>) #Dst Adress Mode
            CASE (Dst Adress Mode = <Register direkt>):
                    PC in, MAR out, PC <= PC + 2, READ,
T4:                 MDR in, MAR out, READ,
T5:                 MDR in, Reg[IR <10:11>] out #Dst Register
            CASE (Dst Adress Mode = <Register indirekt>):
                    PC in, MAR out, PC <= PC + 2, READ,
T4:                 MDR in, MAR out, READ,
T5:                 Reg[IR <10:11>] in, MAR out, WRITE
            CASE (Dst Adress Mode = <Direkte Adressierung>):
                    PC in, MAR out, PC <= PC + 2, READ,
T4                  MDR in, MAR out, READ,
T5:                 MDR in, H1 out,
T6:                 PC in, MAR out, PC <= PC + 2, READ,
T7:                 MDR in, MAR out,
T8:                 H1 in, MDR out, WRITE
```

## Der Subtrahieren Befehl (SUB Src, Dst):

```
T3: SWITCH (IR <4:5>) #Src Adress Mode
    CASE (<Register direkt>):
            SWITCH (IR <8:9>) #Dst Adress Mode
            CASE (Dst Adress Mode = <Register direkt>):
                    Reg[IR <6:7>] in, #Src Register
                    X out, #ALU Register für  ersten Operanden
T4:                 Reg[IR <10:11>] in, #Dst Register
                    Y out, #ALU Register für zweiten Operanden
                    SUBTRAHIEREN (z <= X – Y),
T5:                 Z in, # ALU Register für das Ergebnis
                    Reg[IR <10:11>] out
            CASE (Dst Adress Mode = <Register indirekt>):
                    Reg[IR <6:7>] in, X out,
T4:                 Reg[IR <10:11>] in, MAR out, READ,
T5:                 MDR in, Y out, SUBTRAHIEREN (Z <= X – Y),
T6:                 Z in, MDR out, WRITE
            CASE (Dst Adress Mode = <Direkte Adressierung>):
                    Reg[IR <6:7>] in, X out,
T4:                 PC in, MAR out, PC <= PC + 2, READ,
T5:                 MDR in, MAR out, READ,
T6:                 MDR in, Y out, SUBTRAHIEREN (Z <= X – Y),
T7:                 Z in, MDR out, WRITE
    CASE (Src Adress Mode = <Immediate>):
            SWITCH(IR <8:9>) #Src Adress Mode
            CASE (Dst Adress Mode = <Register direkt>):
                    PC in, MAR out, PC <= PC + 2; READ,
T4:                 MDR in, X out,
T5:                 Reg[IR <10:11>] in, Y out,
                    SUBTRAHIEREN (Z <= x – Y),
T6:                 Z in, Reg[IR <10:11>] out
            CASE (Dst Adress Mode = <Register indirekt>):
                    PC in, MAR out, PC <= PC + 2, READ,
T4:                 MDR in, X out,
T5                  Reg[IR <10:11>] in, MAR out, READ,
T6:                 MDR in, Y out, SUBTRAHIEREN (Z <= X – Y),
T7:                 Z in, MDR out, WRITE
            CASE (Dst Adress Mode = <Direkte Adressierung>):
                    PC in, MAR out, PC <= PC + 2, READ,
T4:                 MDR in, X out,
T5:                 PC in, MAR out, PC <= PC + 2, READ,
T6:                 MDR in, MAR out, READ,
T7:                 MDR in, Y out, SUBTRAHIEREN (Z <= X – Y),
T8:                 Z in, MDR out, WRITE
     CASE (Src Adress Mode = <Direkte Adressierung>):
            SWITCH(IR <8:9>) # Dst Adress Mode
            CASE (Dst Adress Mode = <Register direkt>):
                    PC in, MAR out, PC <= PC + 2, READ,
T4:                 MDR in, MAR out, READ,
T5:                 MDR in, X out,
T6:                 Reg[IR <10:11>] in, Y out,
                    SUBTRAHIEREN (Z <= X – Y),
T7:                 Z in, Reg[IR <10:11>] out
```

(Abb.5.7.2.3: Fortsetzung des Mikrocode SUB – Befehls)

```
                CASE (Dst Adress Mode = <Register indirekt>):
                                PC in, MAR out, PC <= PC + 2, READ,
T4:                             MDR in, MAR out, READ,
T5:                             MDR in, X out,
T6:                             Reg[IR <10:11>] in, MAR out, READ,
T7:                             MDR in, Y out, SUBTRAHIEREN (Z <= X – Y),
T8:                             Z in, MDR out, WRITE
    CASE (Dst Adress Mode = <Direkte Adressierung>):
                                PC in, MAR out, PC <? PC + 2, READ,
T4:                             MDRin, MAR out, READ,
T5:                             MDR in, X out,
T6:                             PC in, MAR out, PC <= PC  + 2, READ,
T7:                             MDR in, MAR out, READ,
T8:                             MDR in, Y out, SUBTRAHIEREN (Z <= X – Y),
T9:                             Z in, MDR out, WRITE
```

(Abb. Fortsetzung des Mikrocode SUB – Befehls)

## Der Addieren Befehl verläuft analog zum Subtrahieren Befehl, mit der Ausnahme:

```
Nur statt SUBTRAHIEREN (Z <= X – Y) jetzt ADDIEREN (Z <= X + Y) [ Dst < = Dst + Src
Der ADD Befehl also ADD Src, Dst
```

(Abb.5.7.2.4: ADD – Befehl)

## Der Eingabe Befehl (IN Gerätecode, Dst):

```
T3: IR <4:7> in, #Gerätekodierung
    A out, #Gerätecontroler Adressregister
    SWITCH (IR <8:9>) #Src Address Mode
    CASE (Dst Adress Mode = <Register direkt>):
T4:         D in, #Gerätecontroller Datenregister
            Reg[IR <10:11>] out #Src Register
    CASE (Dst Adress Mode = <Register indirekt>):
T4:         Reg[IR <10:11>] in, MAR out,
T5:         D in, MDR out, WRITE
    CASE (Dst Adress Mode = <Direkte Adressierung>):
T4:         PC in, MAR out, PC <= PC + 2, READ,
T5:         MDR in, MAR out,
T6:         D in, MDR out, WRITE
```

(Abb.5.7.2.5: Mikrocode IN – Befehl)

## Der Ausgabe Befehl (OUT Src, Gerätecode):

```
        SWITCH (IR <4:5>) #Src Adress Mode
        CASE (Src Adress Mode = <Register direkt>):
T3:             Reg[IR <6:7>] in, #Src Register
                D out #Gerätecontroller Datenregister
        CASE (Src Adress Mode = <Register indirekt>):
T3:             Reg[IR <6:7>] in, MAR out, READ,
T4:             MDR in, D out
        CASE (Src Adress Mode = <Immediate>):
T3:             PC in, Mar out, PC <= PC + 2, READ,
T4:             MDR in, D out
        CASE (Src Adress Mode = <Direkte Adressierung>):
T3:             PC in, MAR out, PC <= PC + 2, READ,
T4:             MDR in, MAR out, READ,
T5:             MDR in, D out
                Reg[IR <8:11>] in, Gerätekodierung
                A out, #Gerätecontroller Adressregister
```

(Abb.5.7.2.6: Mikrocode OUT – Befehl)

## Der Sprungbefehl Springe (JUMP Adresse):

Die Adresse steht als 8-Bit Near Adresse hinter dem Opcode. PC steht nach Ausführung auf neuer Adresse
T1: PC <4:11> <= IR <4:11>

(Abb.5.7.2.7: Mikrocode JUMP – Befehl)

## Der Sprungbefehl Springe wenn größer (JGT Adresse):

Die Adresse steht, wie oben, als 8 – Bit Near Adresse hinter dem Opcode. Der PC steht auf der neuen Adresse wenn das G – Flag gleich 1.
T1: WENN (Flag G = 1), DANN PC <4:11> <= IR <4:11>

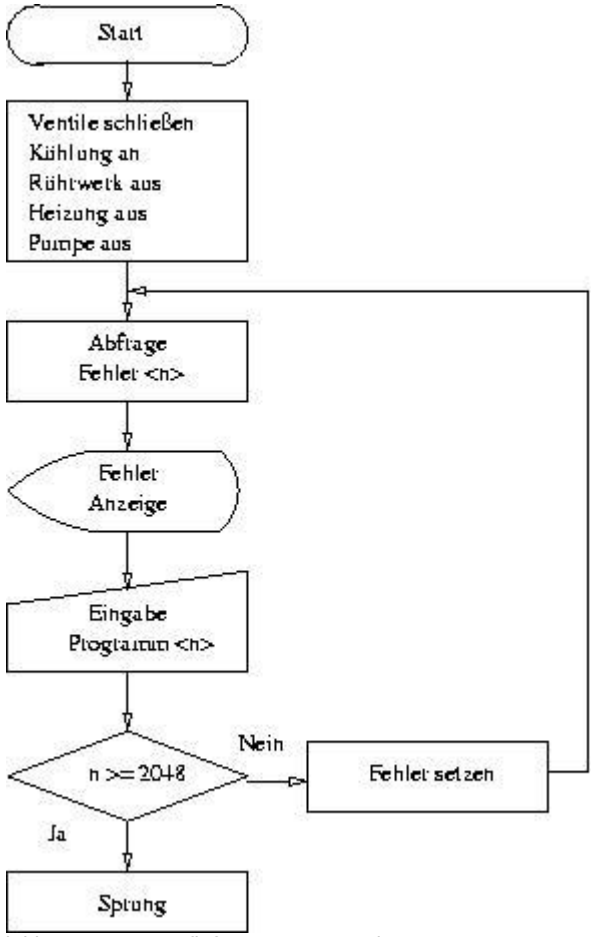
(Abb.5.7.2.8: Mikrocode JGT – Befehl)

# 6 Programme

## 6.1Das Hauptprogramm

Im wesentlichen wartet das Hauptprogramm endlos auf eine Eingabe.
Da wir nicht ausschließen können, dass der Speicher auch wirklich immer beim einschalten mit Null initialisiert wird, ist die erste Aufgabe des Hauprogramms, alle Baugruppen zu initialisieren.

Zusätzlich wird hier die Kühlung aktiviert. Ein gestartetes Bierbrauprogramm deaktiviert diese dann wieder und schaltet sie bei Bedarf zu. Ebenso verfährt das Reinigungs-programm.

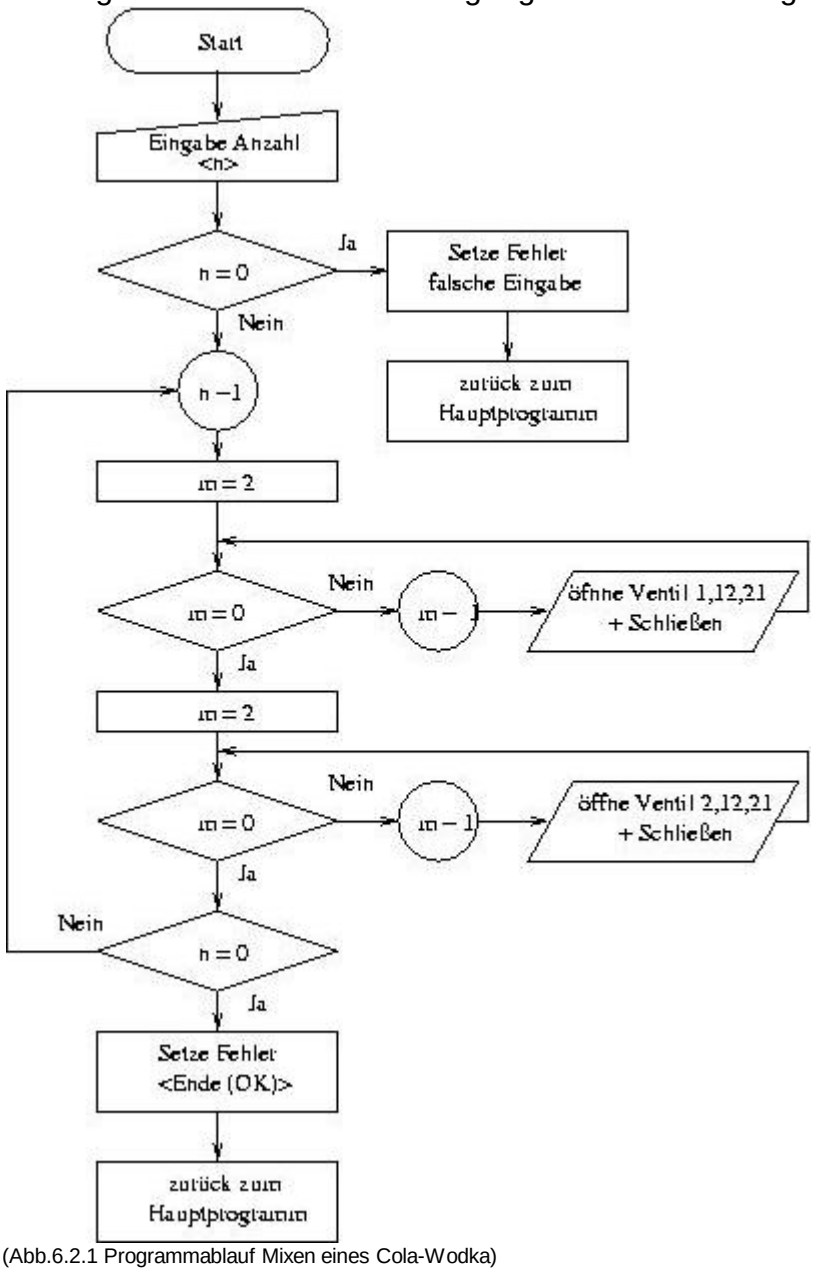



(Abb. 6.1 Programmfluß Hauptprogramm)

```
HProg:    MOVE #24, R0     ;Anzahl der auszuschal-
                           ;tenden Geräte
          OUT #1, KOnOff   ;Kühlung an
          MOVE #1024, R1   ;Adresse des ersten
                           ;Gerät
          JUMP Init        ;Sprung nach Init
Init:     OUT #0, @R1      ;Gerät ausschalten /
                           ;schließen
          ADD #1, R1       ;inc. der aktuellen
                           ;Speichestelle
          SUB #1, R0       ;G Flag setzen
          JGT Init         ;wiederholen wenn G=1
          JUMP Anzeige     ;Forwärtsdeklaration
Anzeige:  MOVE LstErr, R0  ;letzten Fehler holen
          OUT R0, OutIO    ;Fehler schreiben
          JUMP Eingabe     ;Eingabe lesen
Eingabe:  IN InpIO, R0     ;
          MOVE R0, R2      ;sichern
          SUB #2047, R2    ;G Flag setzen
          JGT Exec         ;Sprung nach Exec
          MOVE #1, LstErr  ;Error setzen
          JUMP HProg       ;
Exec:     MOVE #0, LstErr  ;Status OK
          JUMP @R0         ;Ausführen
```

## 6.2 Programm eines Rezepts

Typischerweise besteht der ideale Cola-Wodka aus 2 Teilen Wodka und 3 Teilen Cola. Im Programmablauf wird der Vorgang schematisch dargestellt.




(Abb.6.2.1 Programmablauf Mixen eines Cola-Wodka)

```
Start:      IN InpIO, R0          ;Anzahl der Getränke einlesen
            ADD #0, R0            ;G Flag wenn Eingabe > 0
            JGT  Anzahl
            MOVE #1, LstErr       ;“falsche Eingabe“
            JUMP HProg            ;zum Hauptprogramm
Anzahl:     ADD #0, R0            ;Anzahl noch größer 0 ?
            JGT Wodka             ;Eingang
            MOVE #0, LstErr       ;kein Fehler
            JUMP HProg            ;Start des Hauptprogramms
```

```
Wodka:    MOVE #2, R1          ;temporärer Zähler (Wodka)
          OUT #1, V12OnOff     ;Ventil 12 öffnen
          OUT #1, V21OnOff     ;Ventil 21 öffnen
          JUMP W2              ;Forwärtsdeklaration
W2:       OUT #1, V1OnOff      ;Ventil 1 öffnen
          SUB #1, R1           ;erster Teil durchgelaufen
          OUT #0, V1OnOff      ;Ventil wieder zu
          ADD #0, R1           ;setze G Flag wenn R1 >  0
          JGT W2               ;den zweiten Durchlauf
          OUT #0, V12OnOff     ;Ventil schließen
          OUT #0, V21OnOff
          JUMP Cola            ;springe zur Cola
Cola:     MOVE #3, R1          ;Zähler für die Cola Teile
          OUT #1, V12OnOff     ;Ventil öffnen
          OUT #1, V21OnOff
          JUMP C2              ;Sprung nach C2
C2:       OUT #1, V2OnOff      ;Ventil öffnen
          SUB #1, R1           ;Zähler dekrementieren
          OUT #0, V2OnOff      ;Behälter schließen
          ADD #0, R1           ;G Flag setzen wenn größer
          JGT C2               ;Sprung nach C2
          OUT #0, V12OnOff     ;Ventil schließen
          OUT #0, V21OnOff
          SUB #1, R0           ;Anzahl der Getränke dekr.
          JUMP Anzahl          ;Sprung nach Anzahl
```

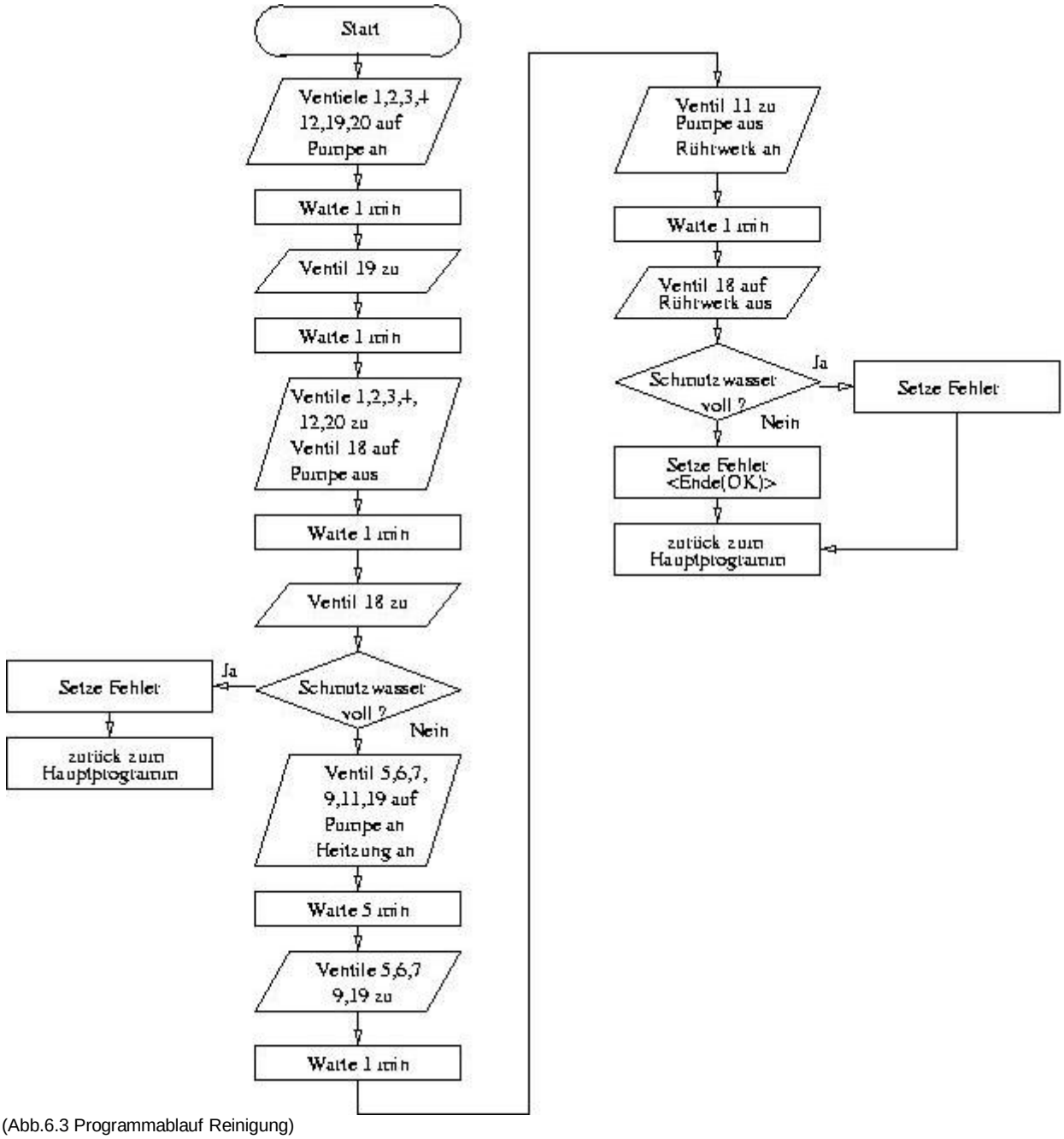
**(Abb.6.2.2 Implementation des Rezept, Fortsetzung von Seite 19)**

## 6.3 Das Reinigungsprogramm

Nach dem alle Getränke gemischt wurden muß das Gerät auch gereinigt werden. Der Nutzer muß dazu alle Mischbehälter mit Reinigungsflüssigkeit befüllen. Das Reinigungsprogramm besorgt den Rest.

Wie die Abbildung (siehe Seite 19) zeigt, ist die Reinigung kein trivialer Prozess. Zur Erklärung soll der Ablauf im Telegrammstil erläutert werden:

(1) Alle Zutatenbehälter für Flüssigkeiten müssen mit Wasser aufgefüllt werden.
(2) Aus den Zutatenbehältern 1 – 4 läuft das Wasser durch die Kühlanlage und den Zapfhan, die Pumpe sorgt dabei für die Umwälzung (ca. 1 min).
(3) Der Zugang zum Schmutzwasserbehälter wird geöffnet (für ca 1 min).
(4) Programm Ende bei Überlauf Schmutzwasserbehälter.
(5) Geöffnete Ventile wieder zu und Wasser aus Zutatenbehältern 5 – 8 durch den Braubottich pumpen, dazu die Heizung an (ca. 5 min).
(6) Den Braubottich „dicht“ machen und das Rührwerk Einschalten.
(7) Ablauf aktivieren (ca. 1 min) und Rücksprung ins Hauptprogramm.




(Abb.6.3 Programmablauf Reinigung)