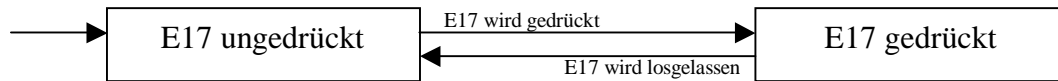


# Musterlösung 1

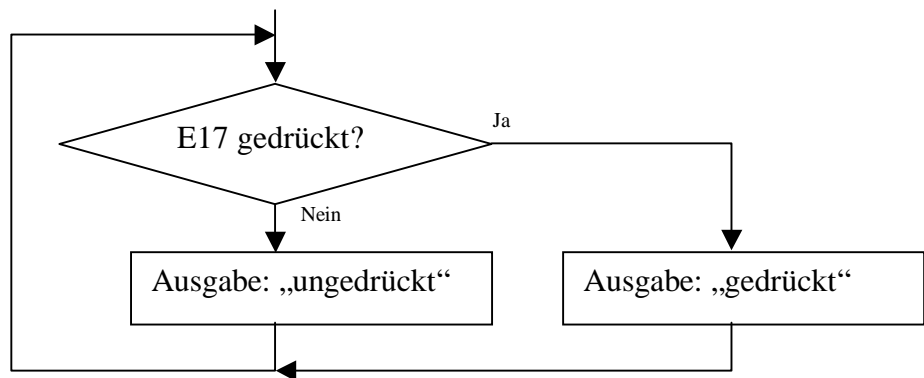
## (1) Schalter 1

Wenn ein Taster (E17) gedrückt wird, soll auf der Anzeige des Terminals das Wort „gedrückt“ erscheinen, andernfalls „ungedrückt“.

Zustandsdiagramm:



Steuerflussdiagramm:



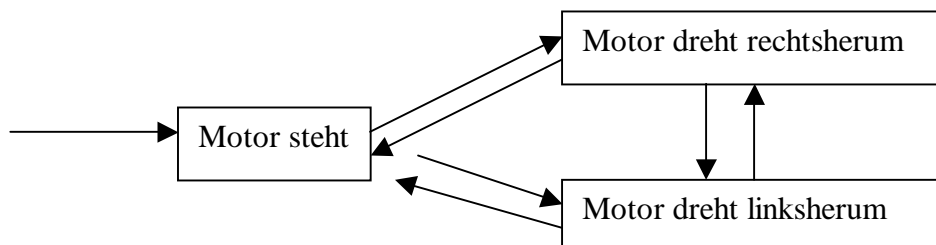
## (2) Schalter 2

Das Programm soll dem aus (1) entsprechen, aber eine Lampe an M1 ansteuern, wenn Taster E1 gedrückt wird.

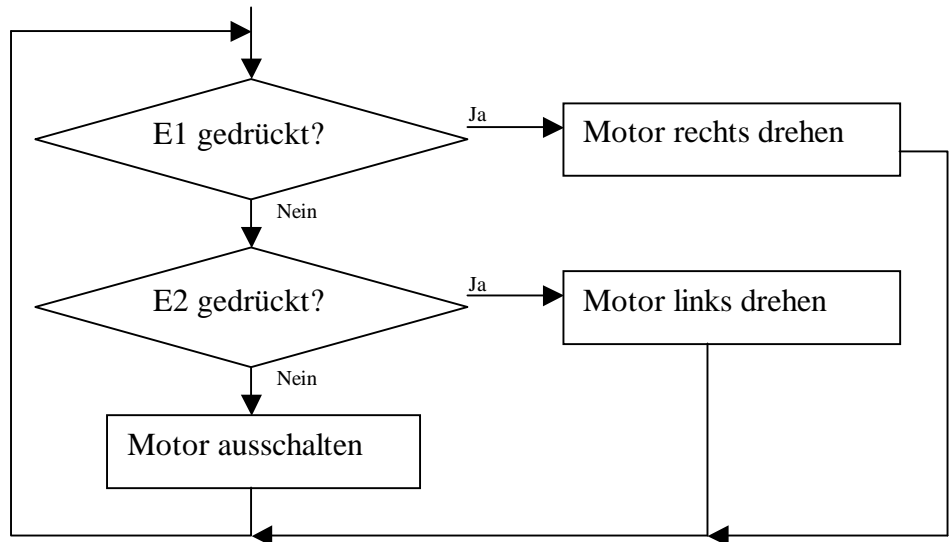
## (3) Motor

Durch Schließen des Tasters an E1 soll ein Motor M1 nach rechts drehen, wird E2 gedrückt, so soll sich der Motor linksherum drehen, ist kein Taster gedrückt, so steht der Motor.

Zustandsdiagramm:



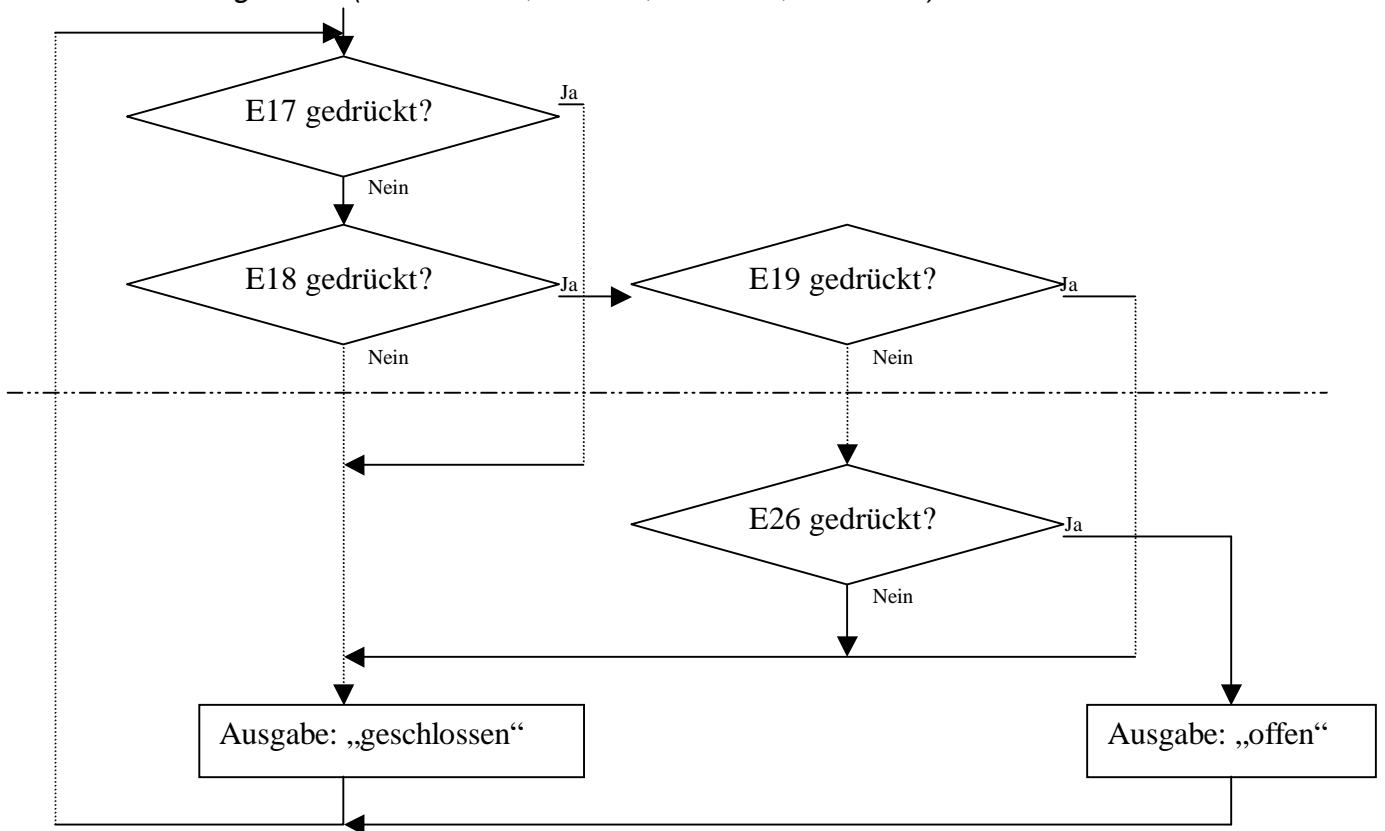
Steuerflussdiagramm:



**(4) Einfaches Codeschloss**

Ein Codeschloss soll simuliert werden, indem auf dem Terminaldisplay „offen“ oder „geschlossen“ angezeigt wird. Um das Schloss zu öffnen, müssen die Schalter E17-E26 richtig ein- oder ausgeschaltet sein.

Steuerflussdiagramm: (für E17 aus; E18 an; E19 aus; ...E26 an)



### (5) Komplexes Codeschloss

Das in (4) beschriebene Schloss ist natürlich nicht besonders sicher. Möglich wäre es, z.B. 10x die richtige Kombination von vier Zahlen (EA-ED) zu fordern, um das Schloss zu öffnen. Hier kann man seinen Gedanken freien Lauf lassen!

### (6) Zufallszahlen-Generator

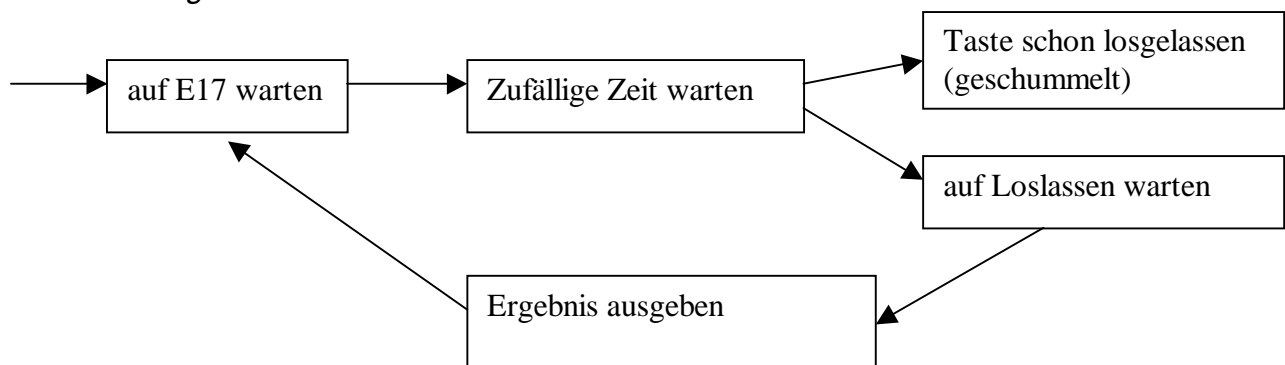
Das Programm soll auf dem Display 1 des Terminals ständig wechselnde Zufallszahlen ausgeben.

*Mögliche Lösung:* In einem Prozess wird eine Variable ständig erhöht (Überlauf-Schutz!), ein anderer Prozess wartet eine zufällig lange Zeit und ließt dann die Variable aus. (Zufallszahl, da der Auslesezeitpunkt vorher nicht festgelegt war)

### (7) Reaktionstester

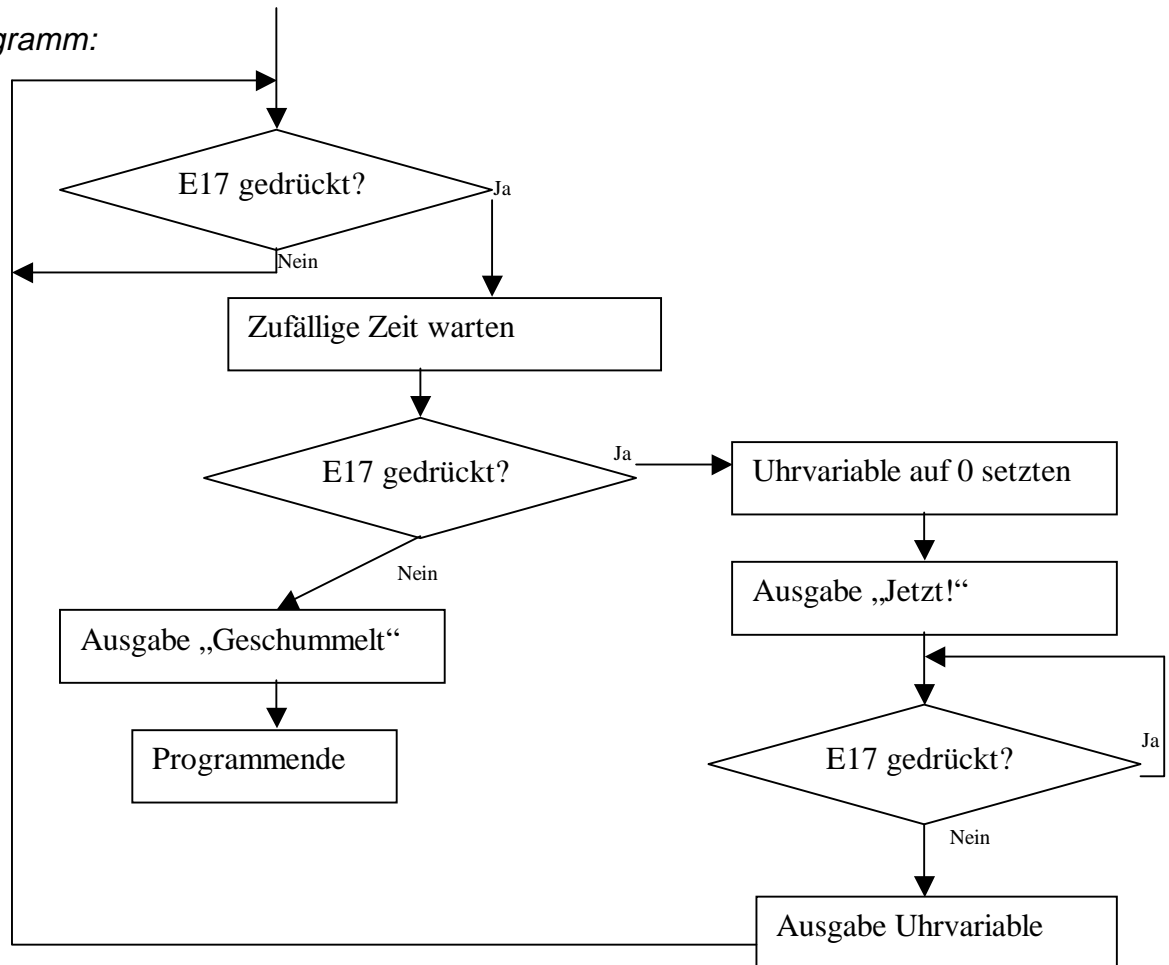
Sobald der Taster E17 geschlossen wurde, löscht das Programm den Meldungstext im Terminalbaustein, wartet eine zufällige Zeit (max. 10 sec), um dann im Terminal den Text „Jetzt!“ auszugeben. Dann wird die Zeit gemessen (in 1/10 sec), die es dauert, bis E17 wieder losgelassen wurde. Diese Zeit wird dann im Display 1 des Terminals ausgegeben und das Programm wartet auf ein erneutes Drücken von E17. Wird der Taster vorher losgelassen, so erscheint ein Meldungstext „Geschummelt!“ und das Programm wird beendet.

*Zustandsdiagramm:*



Ich benötige zwei Prozesse: Einen Hauptprozess, der die oben dargestellten Zustände durchläuft, und einen zweiten, der ständig als Uhr in 1/10-Sekunden-Schritten aufwärts zählt. Sie tauschen Daten über eine Variable aus, die im Hauptprozess initialisiert und im Uhr-Prozess erhöht wird, um dann wieder im Hauptprozess ausgelesen zu werden.

Flussdiagramm:



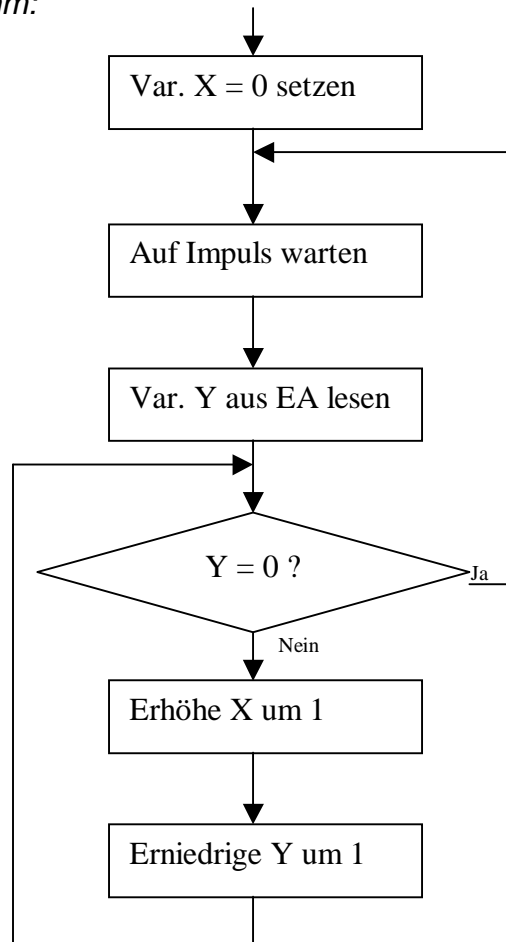
### (8) Addierer (Taschenrechner)

Bei Programmstart zeigt Display 1 des Terminals die Zahl „0“. Bei einem Impuls auf E17 (Drücken und Loslassen) wird zu der Zahl in Display 1 der Wert des (positiven) Terminalparameters EA addiert, und wieder in Display 1 ausgegeben. Dies soll beliebig oft wiederholbar sein.

LLWin bietet uns die Möglichkeit, den Wert einer festgelegten Variable um 1 zu erhöhen oder zu erniedrigen. Außerdem können wir in LLWin Schleifen konstruieren und bedingte Sprünge (Vergleiche) einbauen.

*Informelle Beschreibung:* Eine Addition eines Wertes Y zu einem Wert X kann als eine Y-fache Erhöhung von X um 1 angesehen werden. Wir können also in einer Schleife den Wert von X um 1 erhöhen und den Wert von Y um 1 senken, bis Y=0 ist.

Steuerflussdiagramm:



*Erweiterung:* Bei Impuls auf E18 soll zusätzlich eine Subtraktion durchgeführt werden.

Um eine Subtraktion zu erreichen, muss X in der Schleife erniedrigt werden. An Stelle des Warten auf einen E17-Impuls werden die beiden Eingänge E17 und E18 nacheinander in einer Schleife auf Druck überprüft.

*Erweiterung:* Welche Funktionen kannst du noch einbauen?

Es ist möglich, einen vollständigen Ganzzahl-Taschenrechner in LLWIn zu programmieren (UEB1\_8B.MDL). Nur Mut!