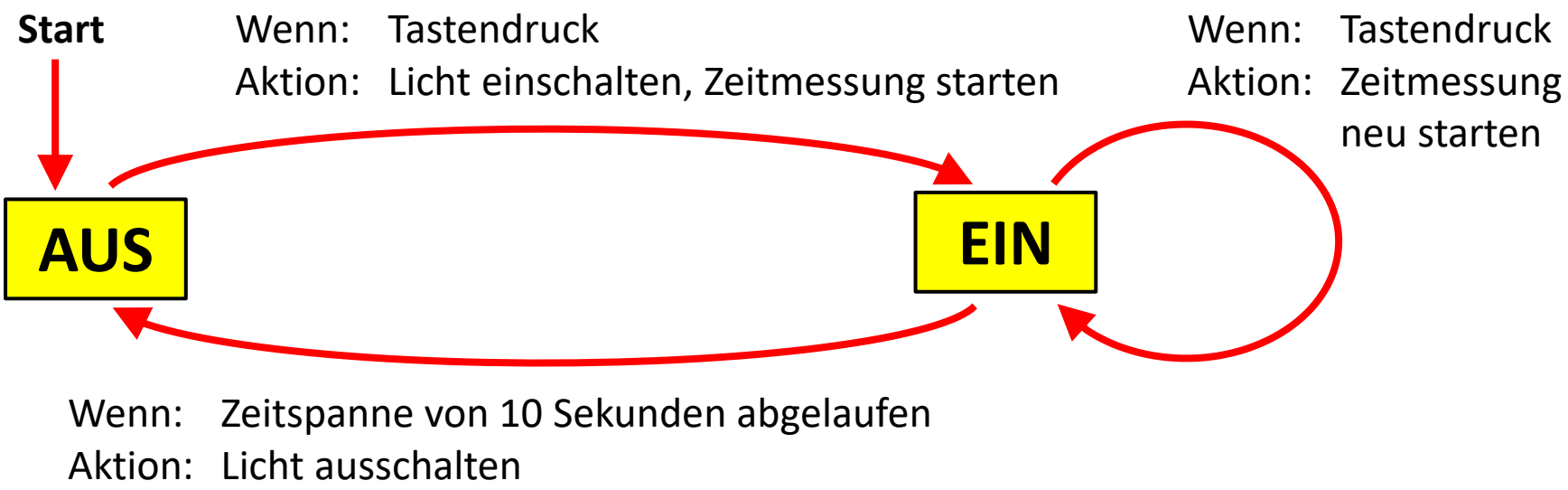
Einleitung

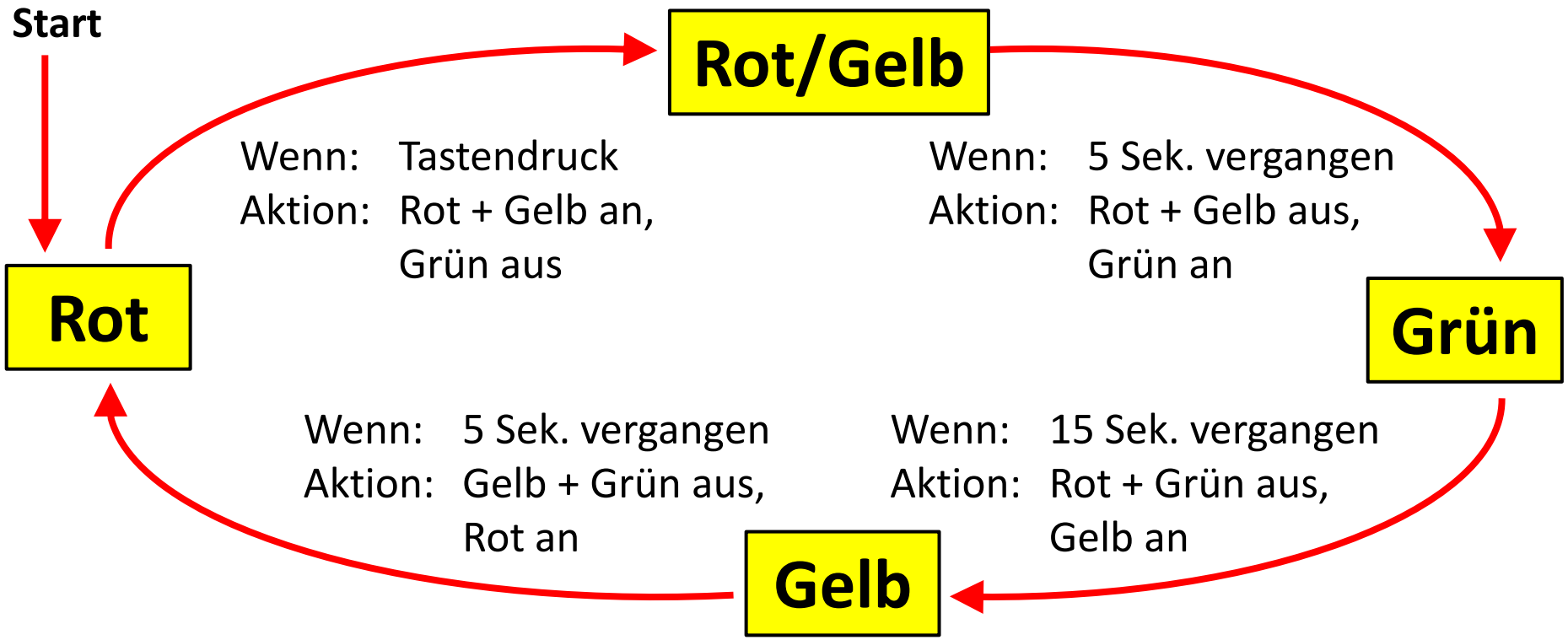
Endliche Automaten sind Modelle, die Systeme mit einer endlichen Anzahl von Zuständen und klar definierten Zustandsübergängen beschreiben. Sie finden in vielen technischen Bereichen Anwendung, etwa um Steuerungen von Aufzügen oder Robotern, Ampelschaltungen, Waschprogramme u. v. a. m. verständlich und übersichtlich zu beschreiben. Das folgende Beispiel zeigt ein automatisches Treppenlicht mit der Möglichkeit zum „Nachdrücken“:



Mit den Zustandsübergängen können Aktionen verknüpft sein, wie das Aktivieren eines Signals, das Starten eines Motors oder das Senden einer Nachricht.

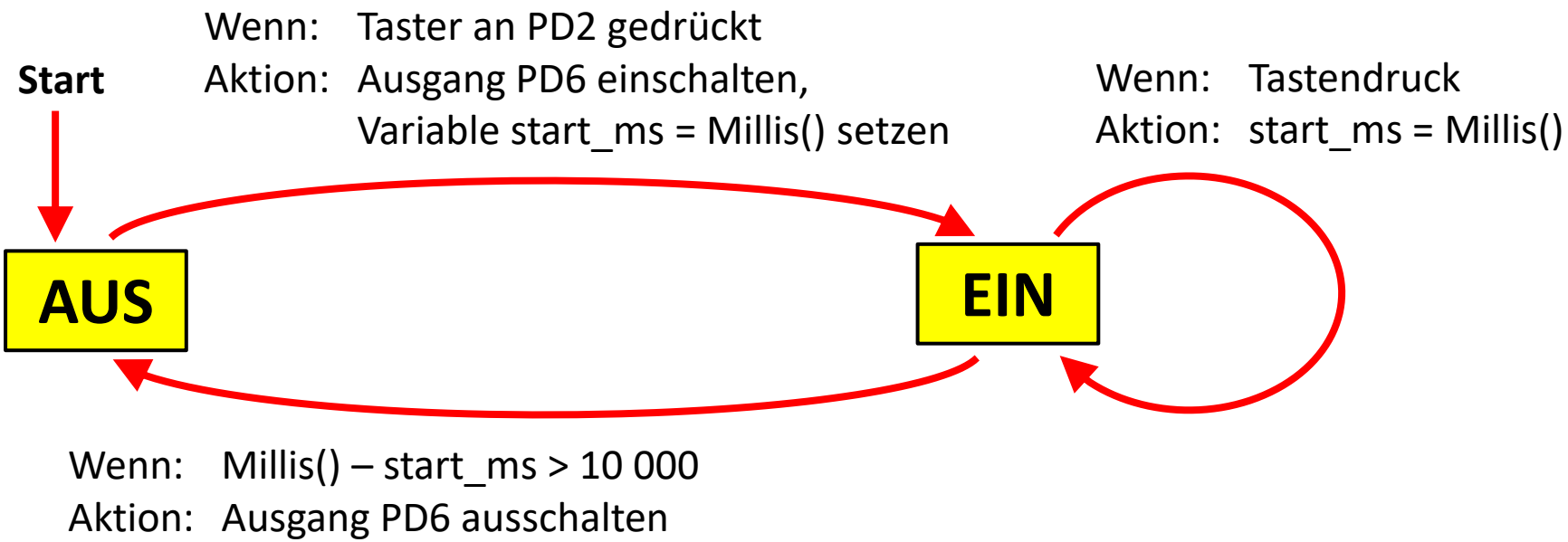
Einleitung

Ein weiteres Beispiel: Auch das Verhalten einer (sehr vereinfachten) Verkehrsampel kann durch einen endlichen Automaten beschrieben werden.



Einleitung

Falls eine Funktion Millis() existiert, welche die Anzahl der Millisekunden seit dem Systemstart ermittelt, dann kann die Steuerung des Treppenlichts folgendermaßen programmiert werden:



Die verschiedenen Zustände („gelbe Kästchen“) werden mittels switch-case programmiert. Aus den Zustandsübergängen („rote Pfeile“) werden if-else-Anweisungen.

Einleitung

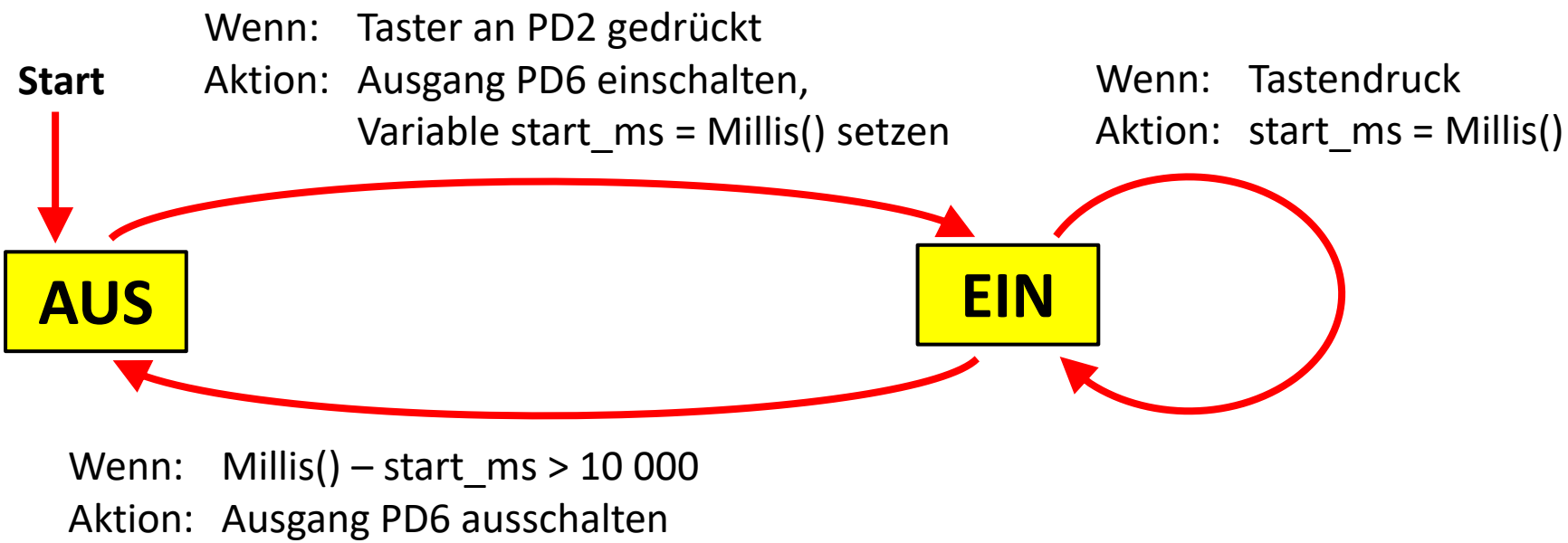
```
// Ports f. Ein-/Ausgänge
#define TASTER    PD2
#define LICHT     PD6

// Liste der Zustände
#define STATE_AUS  1
#define STATE_EIN  2

int main(void)
{
  // Ein-/Ausgänge initialis.
  DDRD  = _BV(LICHT);
  PORTD = _BV(TASTER);

  // Timer 1000 Hz,
  // für Millis()-Funktion
  init_timer();

  // Start-Zustand
  uint8_t state = STATE_AUS;
  uint32_t start_ms = 0;
```

```
while(1)  // Endlosschleife
{
  switch(state)
  {
  case STATE_AUS:
    if((PIND & _BV(TASTER)) == 0)
    { start_ms = Millis();
      PORTD |= _BV(LICHT);
      state = STATE_EIN; }
  break;

  case STATE_EIN:
    if((PIND & _BV(TASTER)) == 0)
    { start_ms = Millis(); }
    else if(Millis() - start_ms > 10000)
    { start_ms = Millis();
      PORTD &= ~_BV(LICHT);
      state = STATE_AUS; }
  break;
  }
}
```

Treppenlicht mit Funktion zum „Nachdrücken“ (a)

Aufgabe G1

Programmieren Sie nun das automatische Treppenlicht, welches durch den aus der Einleitung bekannten endlichen Automaten beschrieben wird:



Zusatzaufgabe: Kurz bevor das Treppenlicht ausgeht, blinkt es 5 Sekunden lang zur „Warnung“. Auch während des Blinkens kann bei Bedarf „nachgedrückt“ werden.

Treppenlicht mit Funktion zum „Nachdrücken“ (b)

Aufgabe G1

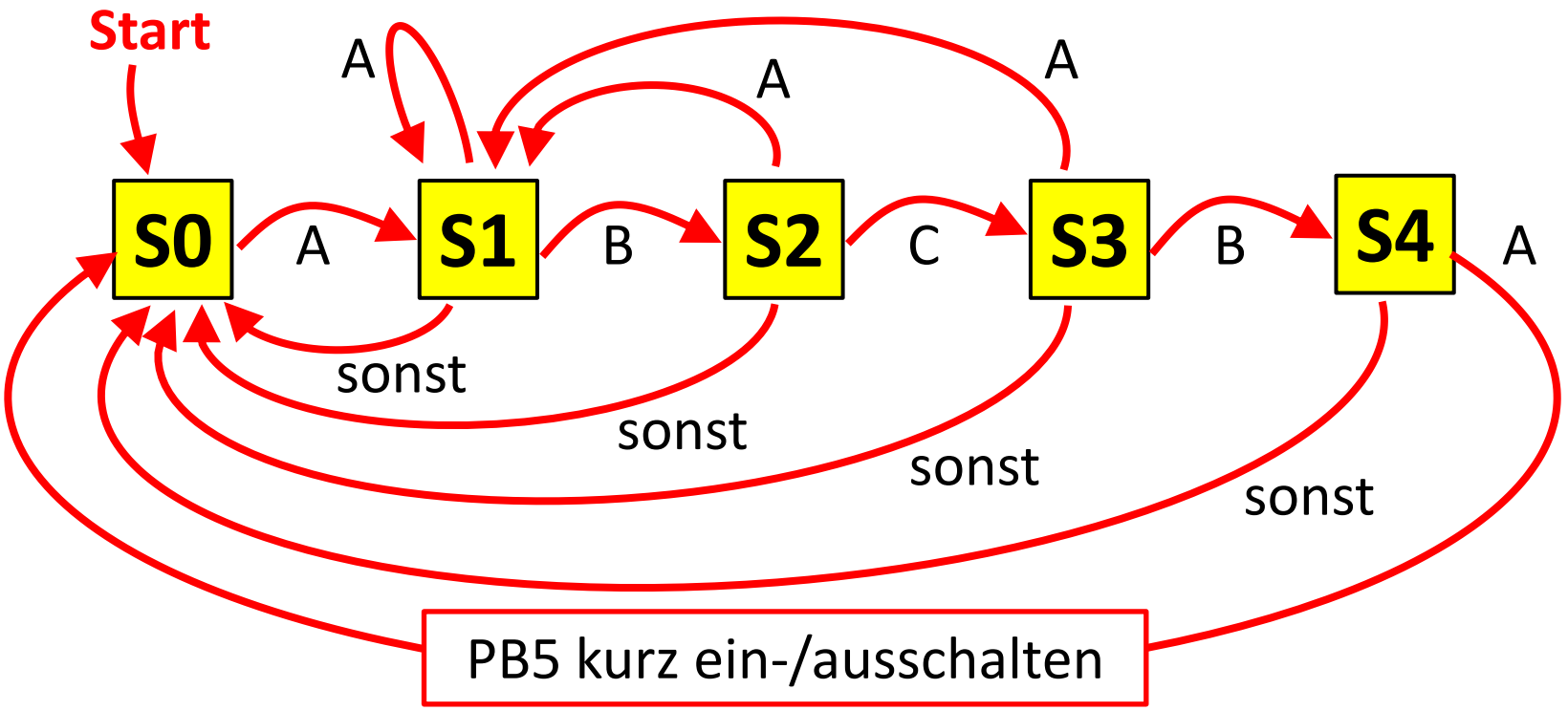
```
volatile uint32_t _millis_intern = 0;  // Millisekunden seit Programmstart

uint32_t Millis(void)  // Millisekunden seit Programmstart abfragen
{
    uint32_t result = 0;
    // Die folgende Zuweisung darf nicht unterbrochen werden!
    cli(); result = _millis_intern; sei();
    return result;
}

ISR(TIMER0_COMPA_vect)  // Timer-Interrupt: Millisekunden zählen
{
    _millis_intern++;
}

void init_timer(void)  // Timer initialisieren, Frequenz = 1 kHz
{
    TCCR0A = _BV(WGM01);              // CTC-Modus (siehe Tab. 19-9, S. 140)
    TCCR0B = _BV(CS01) + _BV(CS00);   // Prescaler = 64 (Tab. 19-10, S. 142)
    OCR0A  = 249;                     // Vergleichswert für Timer

    TIMSK0 = _BV(OCIE0A);             // Compare Match Interrupt (siehe S. 143)
    sei();                            // Interruptsystem des Controllers ein
}
```

Elektronisches Codeschloss

Aufgabe G2

Programmieren Sie ein Codeschloss mit drei Tasten „A“, „B“ und „C“.
Wie lautet der Code zum Öffnen des Schlosses?

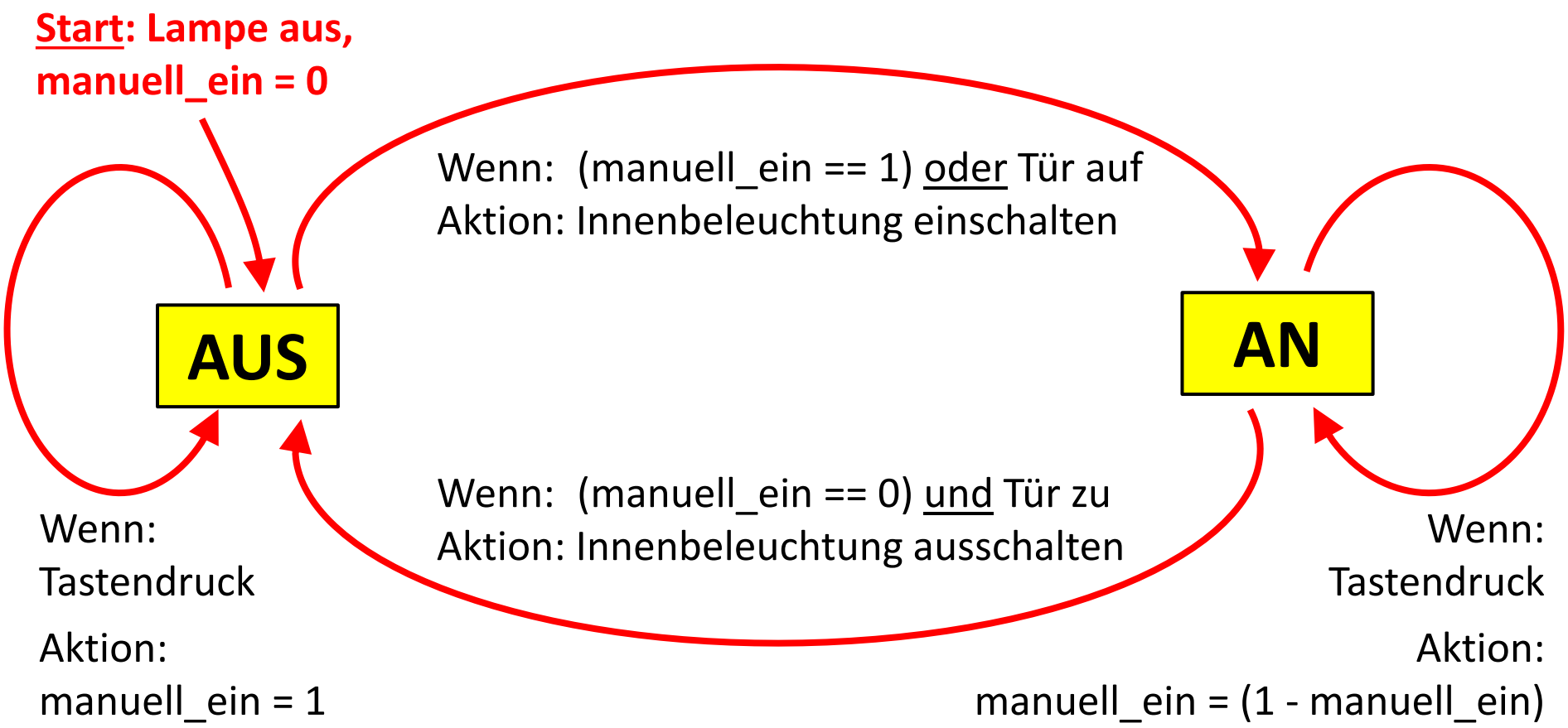
- Die Betätigung der Tasten „A“, „B“ und „C“ wird über die serielle Schnittstelle vom PC an den Mikrocontroller übermittelt.
- Nach Eingabe des korrekten Codes wird die (interne) LED für 500 ms kurz ein- und wieder ausgeschaltet.



Kfz-Innenbeleuchtung

Aufgabe G3

Programmieren Sie ein Steuergerät für die Innenbeleuchtung (LED an PD6) eines Autos. Die Innenbeleuchtung reagiert auf die Stellung der Fahrzeugtür (regelbarer Widerstand an ADC0/PC0) und auf einen Taster (Anschluss PD2).



Weitere Infos zu endlichen Automaten im Internet

[1] Willem van Bergen: Why developers should be force-fed state machines Shopify Engineering Blog, 13.06.2011 (abgerufen: 27.09.2024)
https://shopify.engineering/17488160-why-developers-should-be-force-fed-state-machines

[2] Materialiensammlung des Lehrerfortbildungsservers Baden-Württemberg: Automaten und Sprachen, Bildungsplan 2016 (abgerufen: 27.09.2024)
https://lehrerfortbildung-bw.de/u_matnatech/informatik/gym/bp2016/fb2/05_automaten/

[3] Seite „Endlicher Automat“ in: Wikipedia – Die freie Enzyklopädie. Bearbeitungsstand: 8. Mai 2024, 14:50 UTC (abgerufen: 27.09.2024)
https://de.wikipedia.org/w/index.php?title=Endlicher_Automat&oldid=244789448

In der Informatik wird zwischen „Mealy-Automaten“ und „Moore-Automaten“ unterschieden. Wir haben im Praktikum mit Mealy-Automaten gearbeitet.

[4] Materialiensammlung des Lehrerfortbildungsservers Baden-Württemberg: Mealy-Automaten, Bildungsplan 2016 (abgerufen: 27.09.2024)
https://lehrerfortbildung-bw.de/u_matnatech/informatik/gym/bp2016/fb2/05_automaten/1_hintergrund/1_infos/03_mealy/