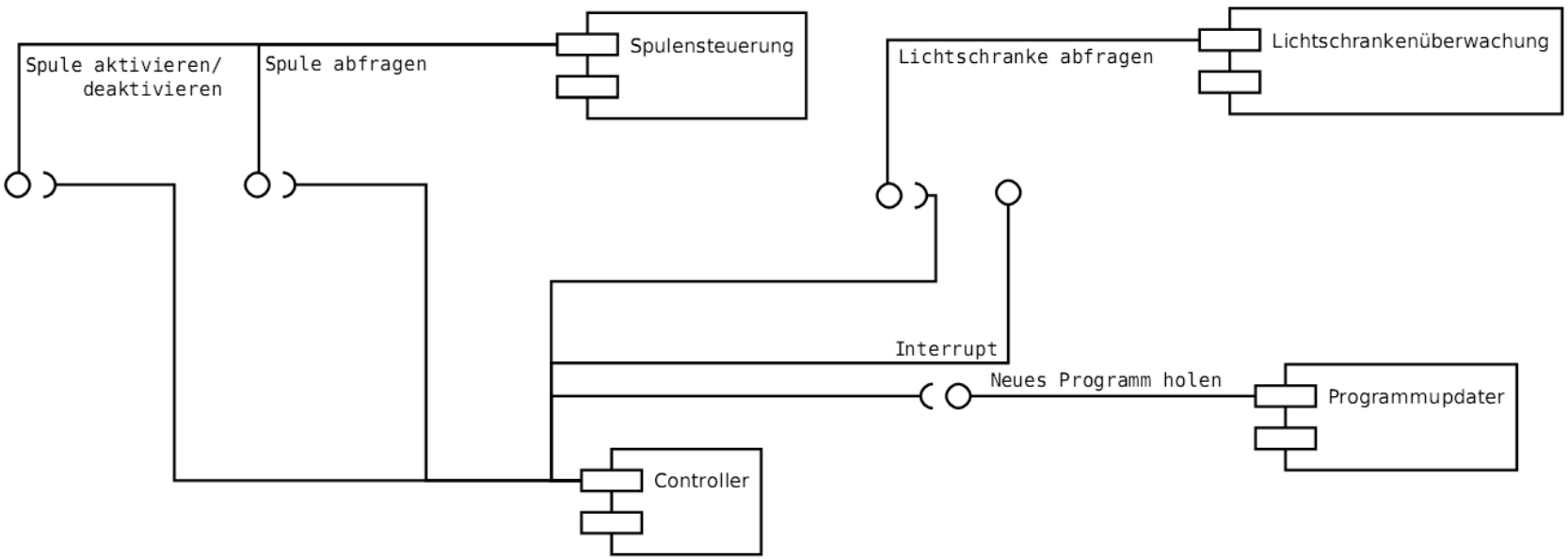
Komponentenaufteilung “Hau den Lukas”

Guilherme Bufolo Martin Oettinger Dirk Pfeiffer

09.06.2009

Inhaltsverzeichnis

Komponentenübersicht



Controller

- Hauptkomponente des Systems
- hält Ablaufplan
- verantwortlich für (schrittweises) Ausführen des Ablaufplans
- verwendet Spulensteuerung und Lichtschrankenüberwachung
- verarbeitet Interrupts u.A. von Lichtschrankensteuerung

Spulensteuerung

- dient dem An- bzw. Abschalten von Spulen
- stellt sicher, dass nur eine Spule gleichzeitig aktiv ist
- stellt sicher, dass Spule nicht zu lange aktiv ist
- zur Zeit aktive Spule kann abgefragt werden

Lichtschrankenüberwachung

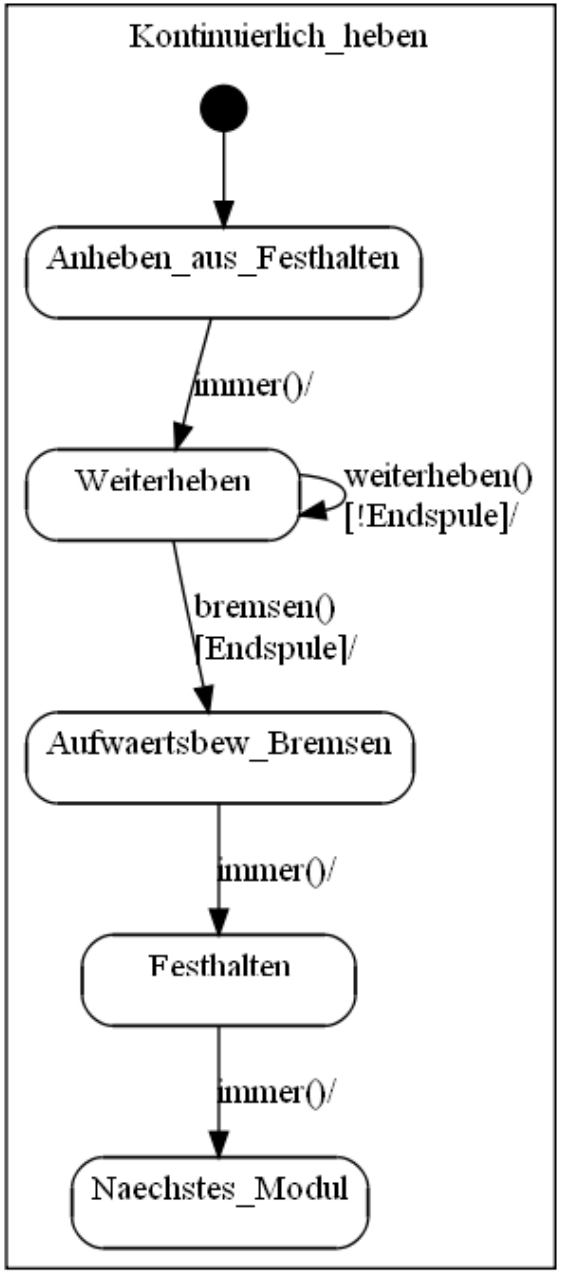
- nimmt Interrupts von der Lichtschrankenhardware entgegen, gibt sie an Controller weiter
- sichert aktuellen Lichtschrankenvektor ($\longrightarrow$ welche Lichtschranken sind ausgelöst)

Folgende Module sind vorgesehen:

- kont_heben
- kont_senken
- pendeln
- schritt_heben
- schritt_senken
- herunterfahren
- schedule monitor
- schedule updater

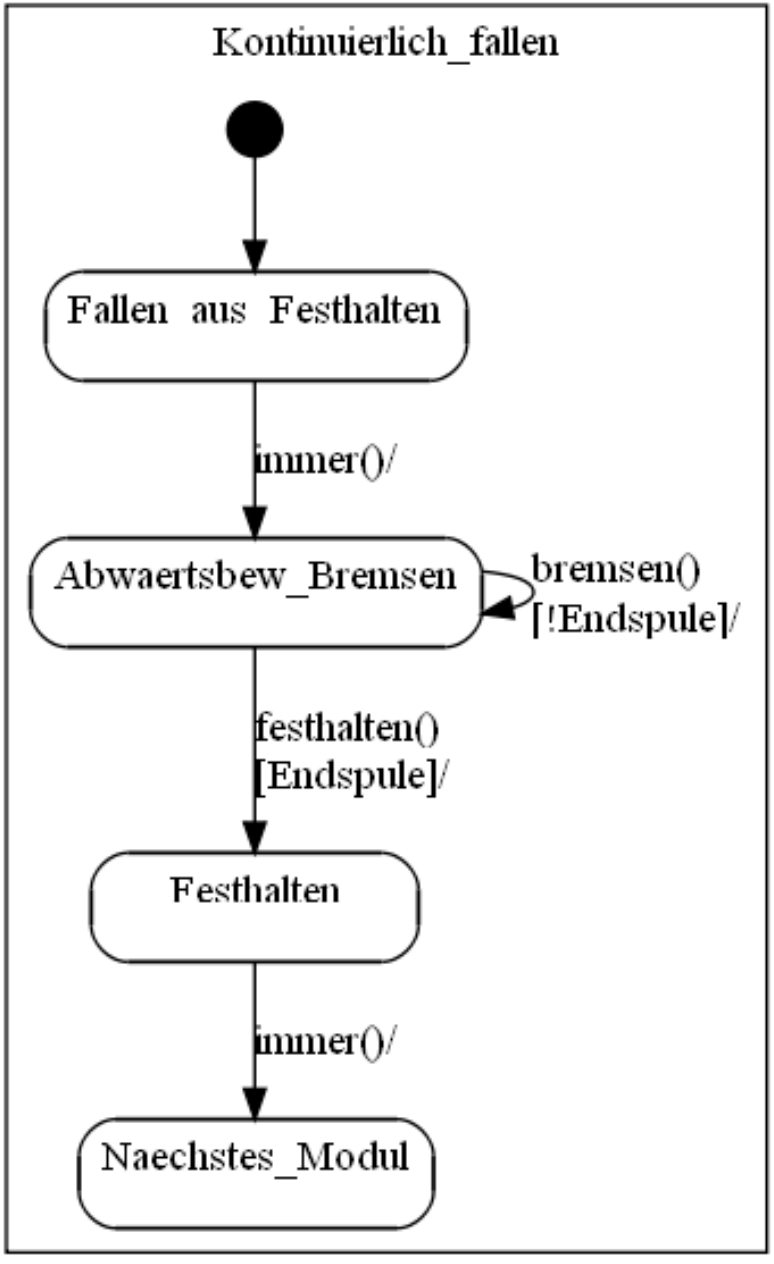
kont_heben(anzahl)

- hebt Projektil um anzahl Spulen nach oben
- Bedingung: anzahl > 0



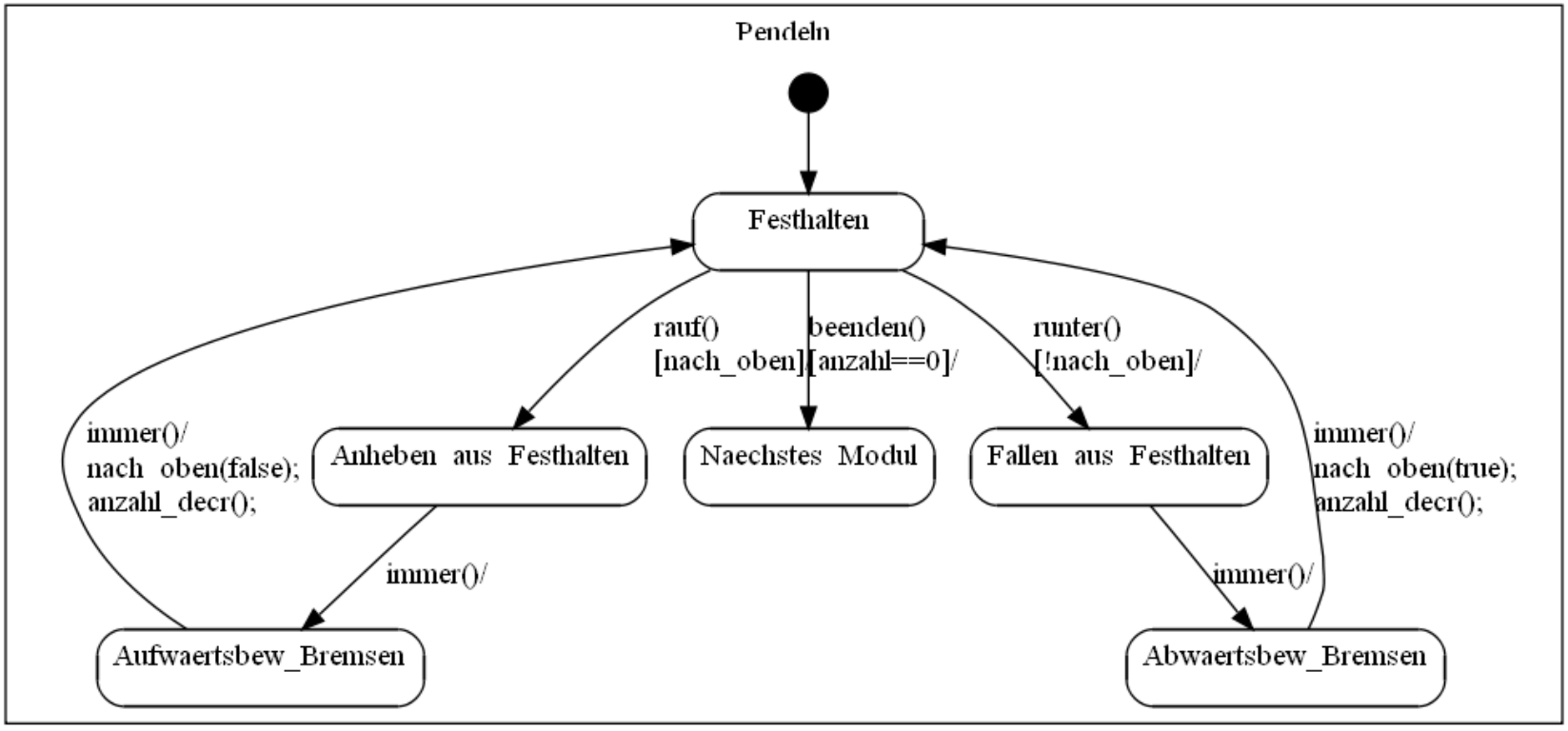
kont_senken(anzahl)

- senkt Projektil um anzahl Spulen ab
- Bedingung: anzahl > 0



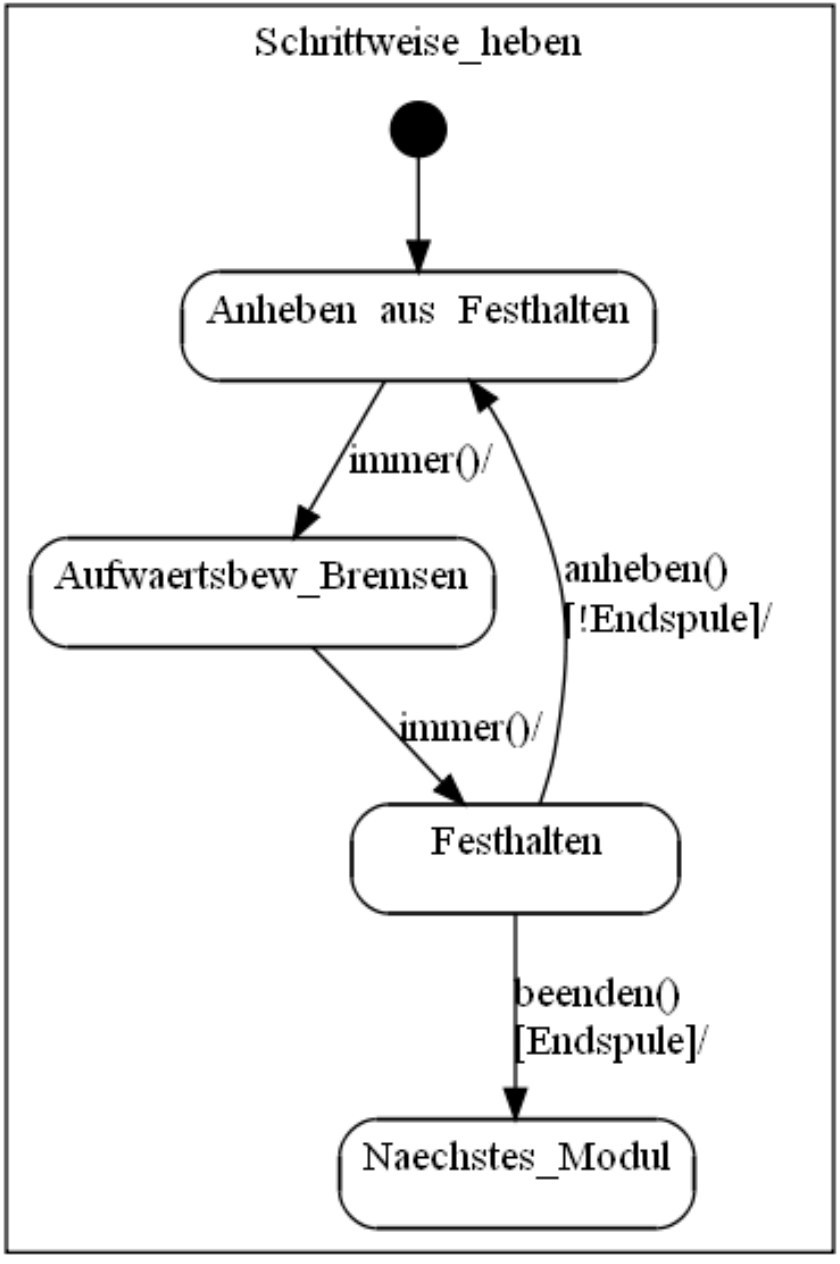
pendeln(anzahl, aufwärts)

- lässt das Projektil zwischen zwei Spulen pendeln
- anzahl gibt die Anzahl der Pendelvorgänge an
- aufwärts gibt an, ob nach oben oder unten gependelt werden soll
- Bedingung: anzahl > 0


Pendeln
Festhalten
rauf()
[nach_oben]
beenden()
[anzahl==0]/
runter()
[!nach_oben]/
immer()/
nach oben(false);
anzahl_decr();
Anheben aus Festhalten
Naechstes Modul
Fallen aus Festhalten
immer()/
nach oben(true);
anzahl_decr();
immer()/
immer()/
Aufwaertsbew_Bremsen
Abwaertsbew_Bremsen

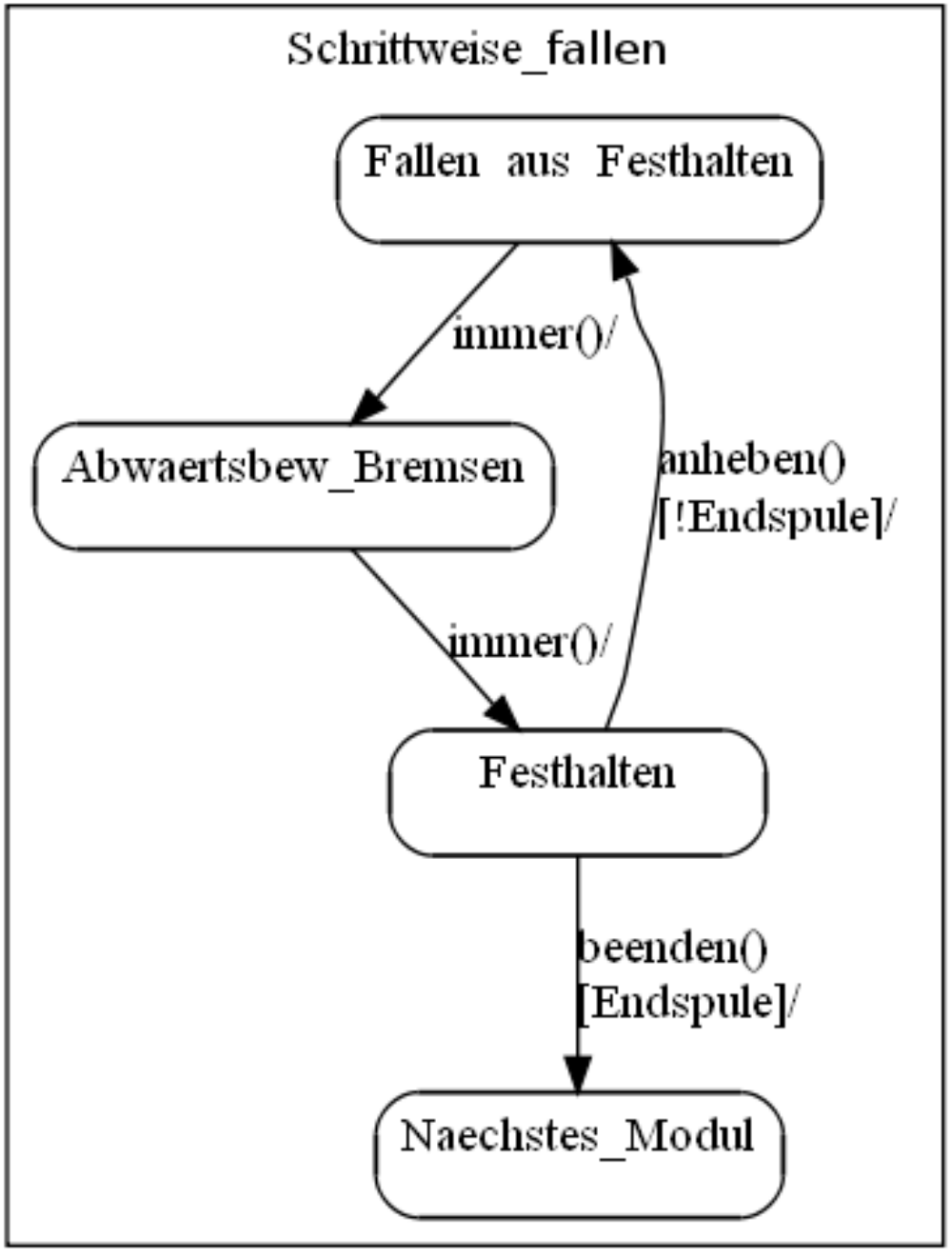
schritt_heben(anzahl)

- hebt das Projektil um anzahl Spulen nach oben, mit kurzem Verweilen an jeder Spule
- Bedingung: anzahl > 0



schritt_senken(anzahl)

- senkt das Projektil um anzahl Spulen nach unten, mit kurzem Verweilen an jeder Spule
- Bedingung: anzahl > 0



herunterfahren

- senkt das Projektil von aktueller Position bis zum Boden ab

schedule monitor

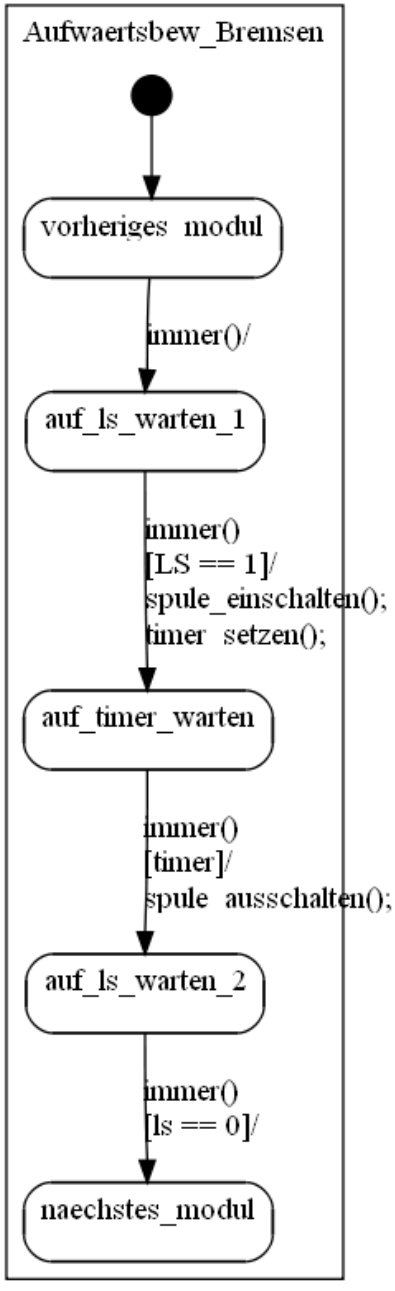
- läuft in jedem DSR
- lastet passendes Modul ein (durch prüfen des Ablaufplans/schedule_vector)
- setzt Flag falls Ablaufplan abgearbeitet
- bei Unregelmäßigkeit wird Fehlerbehandlung ausgelöst

schedule updater

- arbeitet im Hintergrundbetrieb
- nimmt Daten an serieller Schnittstelle entgegen
- füllt sekundären Ablaufplan
- setzt Flag falls dieser komplett übertragen
- falls sekundärer Plan vorhanden: neue Übertragung nur mit explizitem Überschreibbefehl

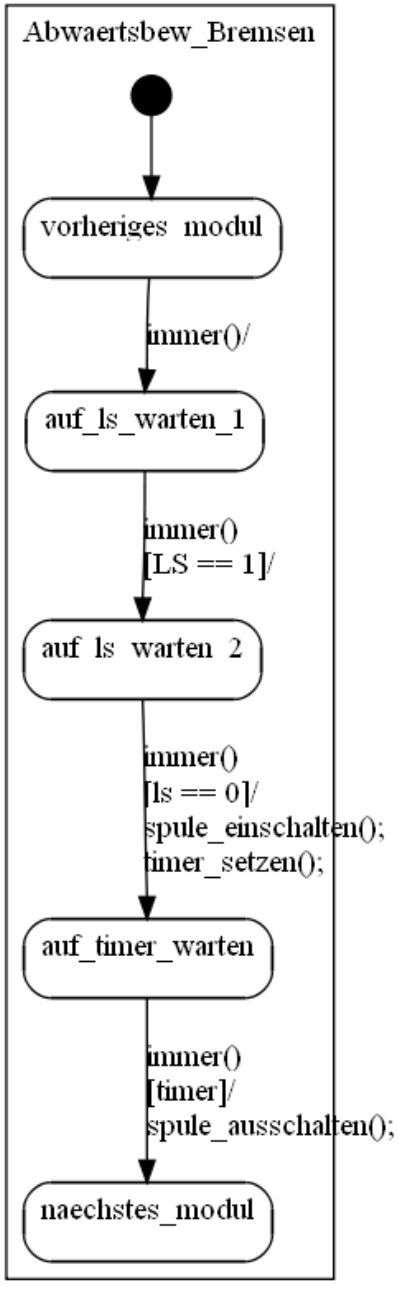
aufwaertsbew_bremsen

- Spule n kurz aktivieren sobald Schranke n belegt wird



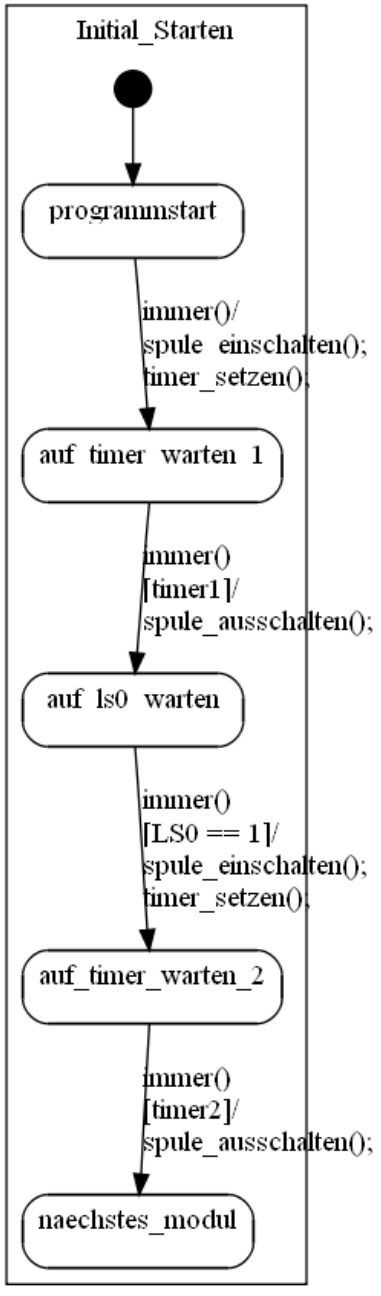
abwaertsbew_bremsen

- Spule n kurz aktivieren sobald Schranke n nach Belegung wieder frei wird



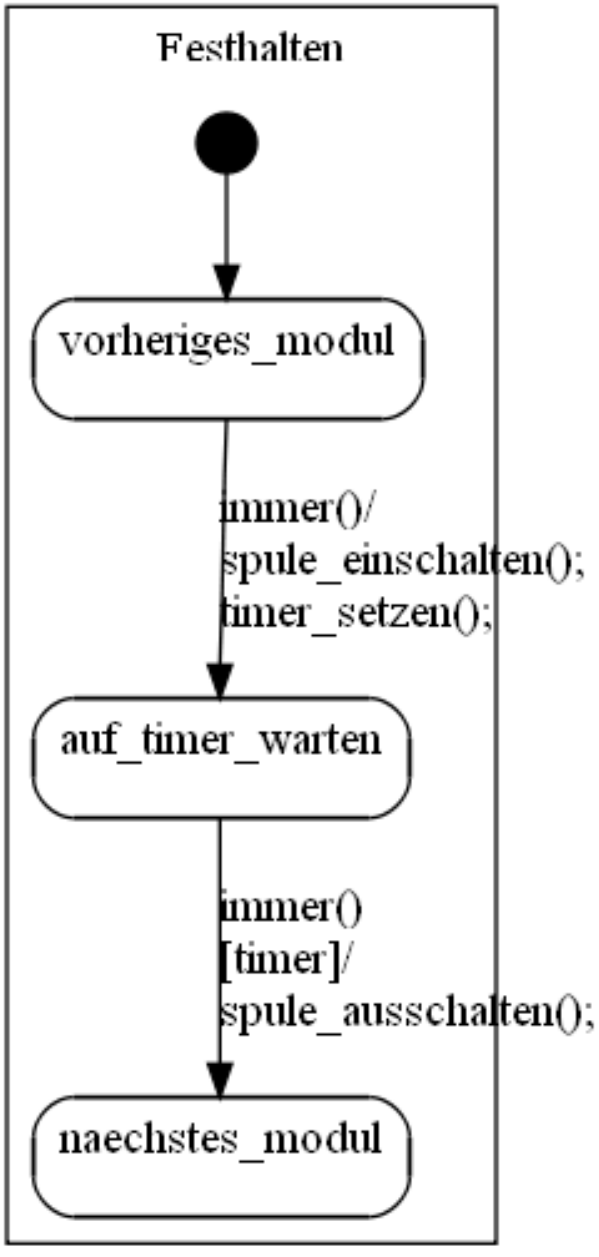
initial_starten

- Spule 0 einschalten bis Schranke 0 belegt



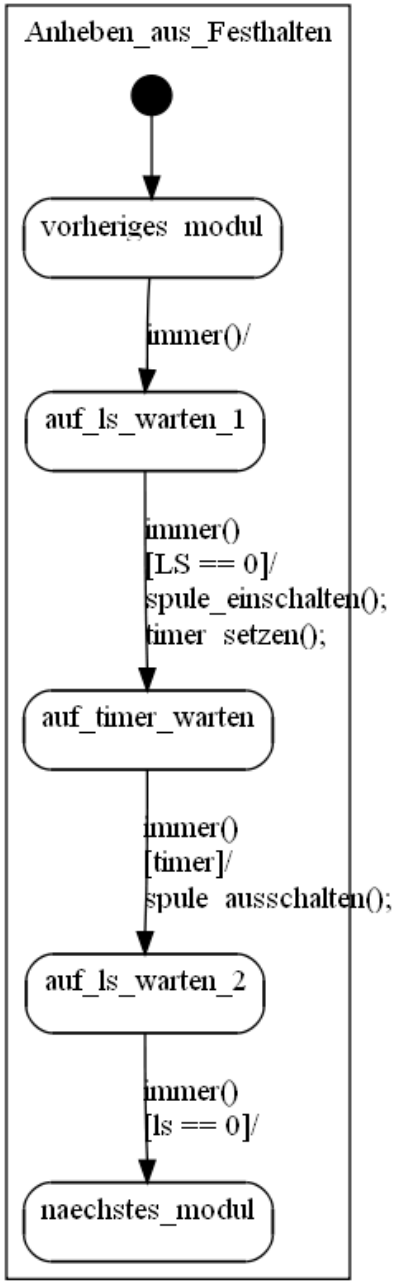
festhalten

- Spule n einschalten



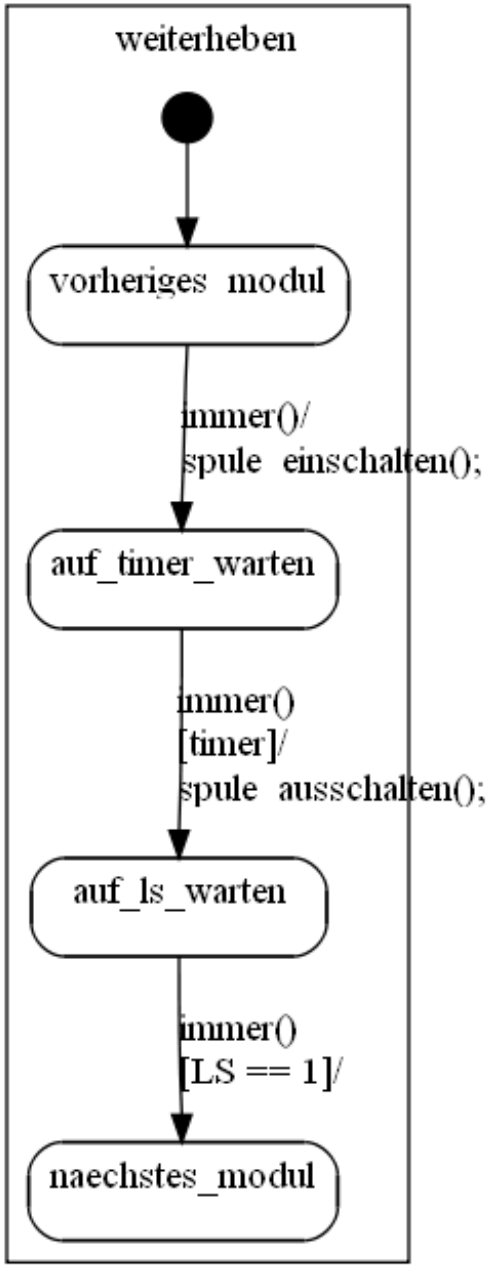
anheben_aus_festhalten

- Spule n deaktivieren bis Schranke n frei wird
- Spule n aktivieren bis Schranke n belegt wird



weiterheben

- Spule n+1 einschalten, sobald Schranke n belegt
- Spule n+1 abschalten sobald Schranke n+1 belegt



Data coupling, Synchronisierung

- LSM und Updater teilen sich eine Variable sowie einen Pointer
- Variable wird by-value geteilt, Pointer wird während Veränderung vor Zugriff geschützt
- andere Module teilen kopierten Lichtschrankenvektor mit Lichtschrankenüberwachung, schreibender Zugriff nur durch LSM
- keine Nebenläufigkeit bei beteiligten Modulen, daher kein Schutz des Vektors nötig

Vielen Dank für Ihre Aufmerksamkeit!
Fragen?