# Befehlsvalidierung Modul-Design

Für das studentische Projekt *Sichere Eisenbahnsteuerung*

| | |
|---|---|
| **Datum** | 18.05.2010 |
| **Quelle** | Dokumente\02_Design\02.02_Moduldesign\02.02.00_PDFs\ Modul-Design_Befehlsvalidierung |
| **Autoren** | O. Bohn |
| **Version** | 2.0 |
| **Status** | Zur Verwendung freigegeben |

# 1 Historie

| Version | Datum | Autor | Bemerkung |
|---|---|---|---|
| 0.1 | 24.11.09 | Vitali Voroth | Initial-Version, Einleitung |
| 0.2 | 14.12.09 | Vitali Voroth | Kapitel 5 und 6 bearbeitet |
| 0.3 | 16.12.09 | Vitali Voroth | Kapitel 5 und 6 überarbeitet |
| 0.4 | 19.12.09 | Vitali Voroth | Weitere Verfeinerungen in Kapitel 5 und 6 |
| 0.5 | 21.12.09 | Vitali Voroth | Gleissensoren nummeriert, Schnittstellenbeschreibungen hinzugefügt. |
| 0.6 | 22.12.09 | Vitali Voroth | Kapitel 5.2: Gleisabschnitt als Struct-Array (ohne Zeiger) definiert<br>Kapitel: 5.6: Schnittstellen zur Leitzentrale um Streckenbelegung erweitert, verwendete globale Variablen aufgelistet. |
| 1.0 | 10.01.10 | Vitali Voroth | Vorschläge aus dem Review (außer Nr. 2) eingearbeitet |
| 1.1 | 18.01.10 | Vitali Voroth | Aktivitätsdiagramm „Teilaufgaben des Moduls“ verändert, die Streckentopologie wird jetzt seltener abgeglichen. (Kapitel. 5.1)<br>Nachbedingung von checkSensorDaten geändert (Kapitel 5.5) |
| 1.2 | 25.01.10 | Vitali Voroth | Einleitung: Satz entfernt, in dem fälschlicherweise stand, dass die BV ein Not-Aus macht, wenn die LZ einen schlechten Befehl liefert.<br>Im Kapitel 5.5 (Lokale Funktionen) erwähnt, dass es noch weitere lokale Funktionen geben darf. |
| 1.3 | 04.02.10 | Vitali Voroth | Kapitel 5.6.3: „Auditing-System“ hinzugefügt.<br>Die Variablen *criticalStateCounter* und *nextState* sind nicht mehr global. |
| 2.0 | 18.05.10 | O. Bohn | Abgleich zwischen Designdokument und Quellcode.<br>Abschnitt 4: Referenzierte Dokumente an den Stand des SoSe Projekts anpassen.<br>Korrektur von Rechtschreib- bzw. Satzbaufehlern in Abschnitt 6.<br>Ergänzen der lokalen Hilfsfunktionen in Abschnitt 5.5 sowie ein kurze Beschreibung zur Funktionalität der Hilfsfunktionen. |

# 2 Inhaltsverzeichnis

# 3 Einleitung

Das Modul *Befehlsvalidierung* ist in der Sicherheitsschicht der Eisenbahnsteuerung-Software angesiedelt, die im Rahmen des studentischen Projekts *Sichere Eisenbahnsteuerung* im Wintersemester 2009/2010 an der Hochschule Bremen entwickelt wird.

Die Befehlsvalidierung prüft zum einen die von der Leitzentrale (Anwendungsschicht-Modul) kommenden Streckenbefehle auf Gültigkeit und leitet sie an die Ergebnisvalidierung weiter. Zum anderen reicht sie die vom S88-Treiber kommenden Sensordaten an die Leitzentrale weiter falls dieser keinen Fehler gemeldet hat. Falls der S88-Treiber aber einen Fehler meldet, schaltet die Befehlsvalidierung das System über den Not-Aus-Treiber aus.

Für die Prüfung der Streckenbefehle hat die Befehlsvalidierung ein eigenes Streckenabbild inklusive Gleisabschnitts-Belegung. Die in der Befehlsvalidierung definierte Streckentopologie kann von der Leitzentrale lesend mitbenutzt werden um Redundanz zu vermeiden.

# 4 Referenzierte Dokumente

- Dokumente\01_Anforderungsanalyse\01.00_Pflichtenheft\01.00.00_PDFs
- Dokumente\02_Design\02.01_Subsystemdesign\02.01.02_noch nicht freigegebene Dokumente\Hardware-Design
- Dokumente\02_Design\02.01_Subsystemdesign\02.01.02_noch nicht freigegebene Dokumente\Software-Design
- Dokumente\02_Design\02.02_Moduldesign\02.02.00_PDFs\Modul-Design_Betriebsmittelverwaltung

# 5 Architektur

## 5.1 Übersicht

In Abbildung 1 sieht man alle moduleigenen Komponenten, die gebraucht werden, um das aktuelle Streckenabbild inklusive Weichenstellung, Gleisabschnittsbelegung und Zugposition zu speichern.

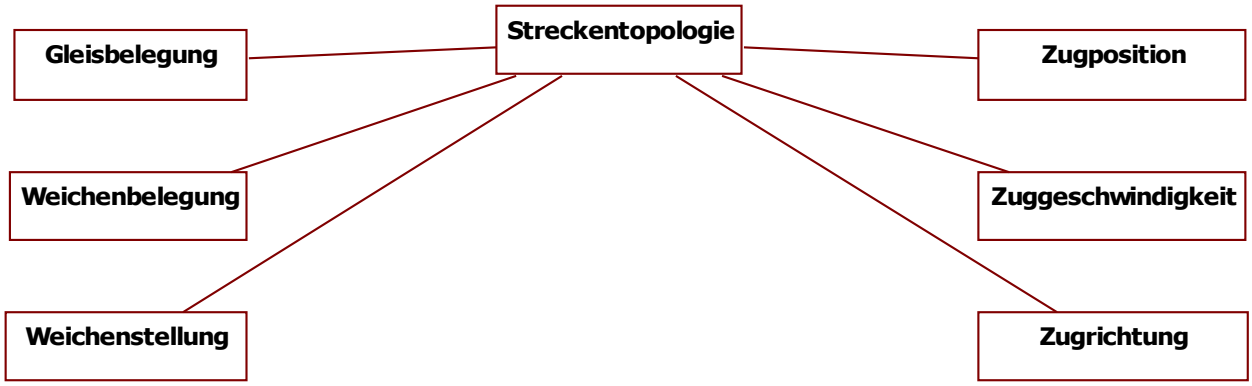



*Abbildung 1: Streckenabbild Komponenten*

In Abbildung 2 sind die Teilaufgaben des Moduls dargestellt, die nacheinander abgearbeitet werden. Sollten sie zusammen zu lange brauchen um in der vorgegebenen Maximalzeit fertig zu werden, kann am Ende jeder Teilaufgabe die Kontrolle an die Betriebsmittelverwaltung zurückgegeben werden - der nächste abzuarbeitende Schritt ist dann in der statischen Variable *nextState* zu speichern.

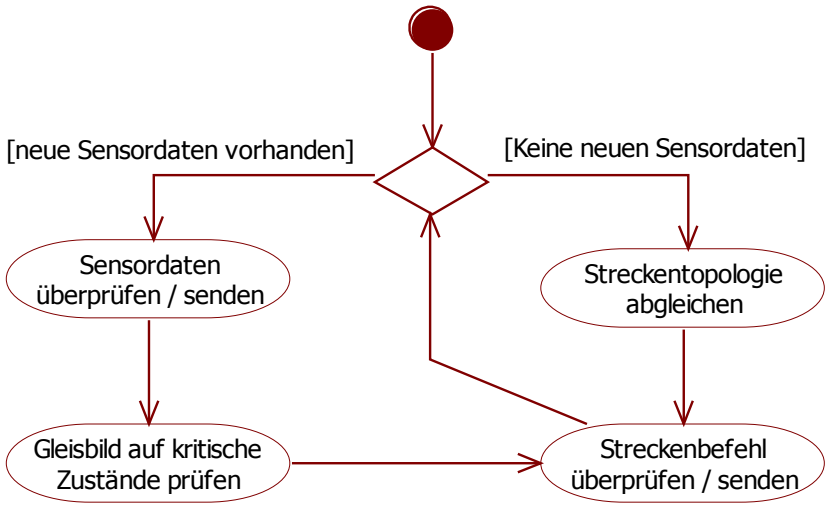



*Abbildung 2: Teilaufgaben des Moduls*

### 5.2 Streckentopologie

Die Topologie der Strecke wird unter anderem durch die Lage der Rückmeldesensoren festgelegt. Ausgehend von dem Streckenaufbau (siehe auch Abschnitt4: Referenzierte Dokumente: Hardware-Design) wurde hier nun die Lage der Sensoren festgelegt und das Gleissystem somit in Gleisabschnitte *1* bis *9* unterteilt. Die rot nummerierten Sensoren sollen sich in etwa an den in Abbildung 3 mit roten Linien markierten Stellen befinden, die exakten Positionen (vor allem der Sensoren neben den Entkopplern) werden noch ausgemessen und dokumentiert. Ferner sind die Weichen *a*, *b und c* zu sehen. Die automatischen Entkoppler werden mit *E1* und *E2* bezeichnet.

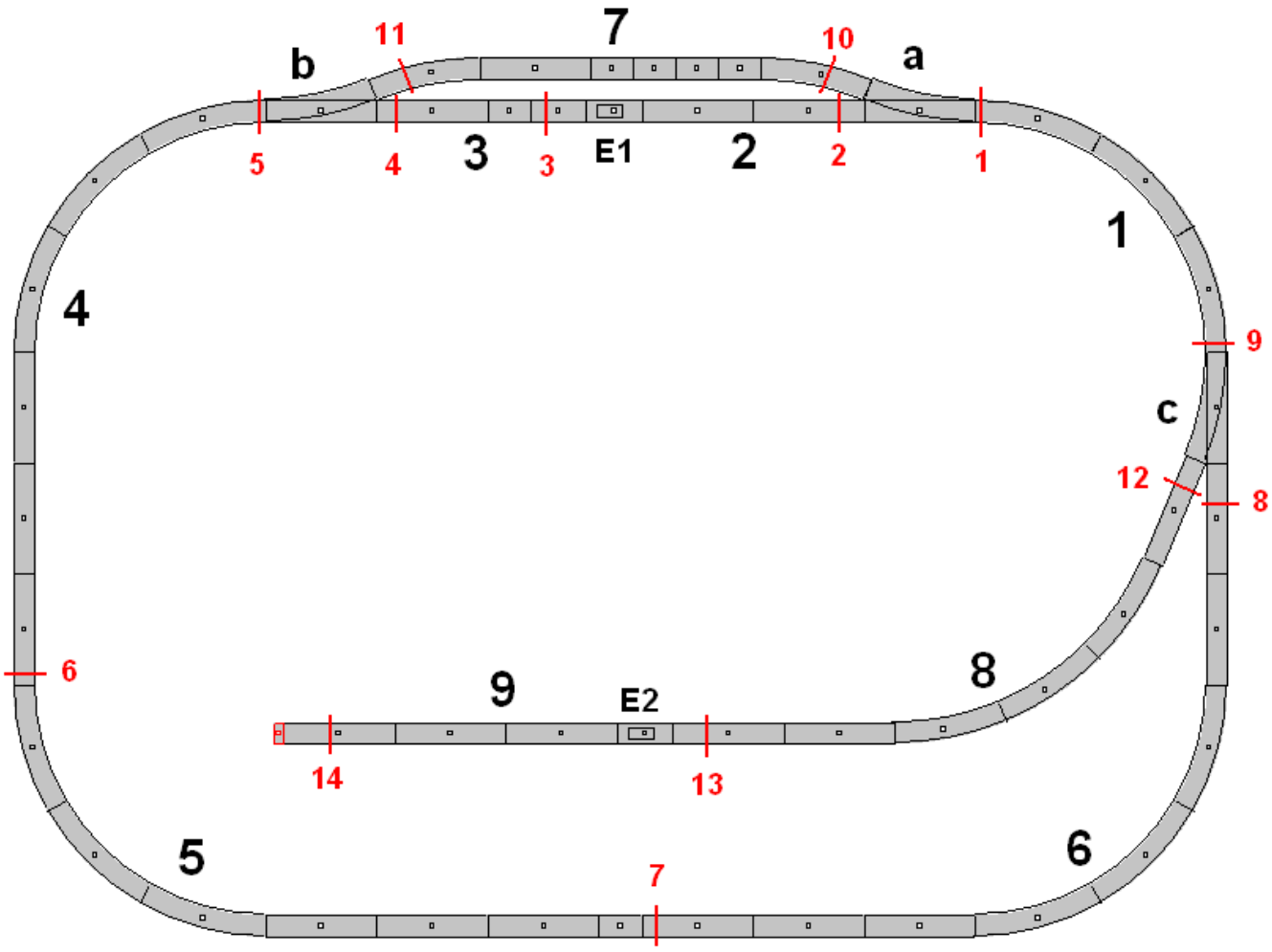



*Abbildung 3: Gleisabschnitte, Weichen und Entkoppler*

Die Streckentopologie wird mit einer mehrfach verketteten Liste von Gleisabschnitten modelliert (siehe Abb. 4). Jeder Gleisabschnitt hat mindestens einen Nachfolger (gegen den Uhrzeigersinn) und mindestens einen Vorgänger (mit dem Uhrzeiger), außerdem sind höchstens zwei Weichen an einem Gleisabschnitt dran. Es gibt ein statisches Array mit den *Gleisabschnitt*-Strukturen, *next* und *prev* sind als Indizes auf dieses Array zu verstehen.

Abbildung 5 zeigt beispielhaft die Belegung der Felder vom Gleisabschnitt 1.

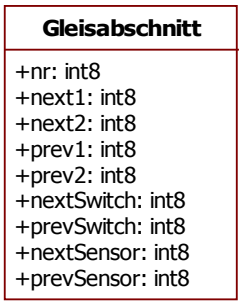



*Abbildung 4: Gleisabschnitt*

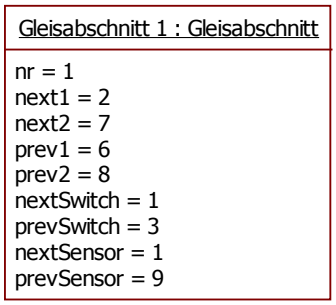



*Abbildung 5: Beispiel: Gleisabschnitt 1*

Diese Liste von Gleisabschnitten wird der Leitzentrale in Form der globalen Variable *BV_streckentopologie* zur Verfügung gestellt. Damit die Leitzentrale keine Änderungen daran vornehmen kann, bekommt sie nur eine Kopie davon. Die Kopie darf ebenfalls nicht geändert werden und kann bei jedem Aufruf des Moduls gegen das Original geprüft werden, um bei Inkonsistenz ein Not-Aus auszulösen.

### 5.3 Gleisabschnitts- und Weichen-Belegung

Für jeden Gleisabschnitt und jede Weiche gibt es einen Zähler, der wenn er Null ist besagt, dass der entsprechende Abschnitt frei ist. Frei bedeutet, dass nichts (keine Lok und kein Waggon) darauf steht. Die Zähler sind zu Arrays zusammengefasst und werden beim Start entsprechend der Fahraufgabe initialisiert.

Das Array *gleisBelegung* hat wegen einfacherer Indizierung ein Element mehr als es Gleisabschnitte gibt: Die Anzahl der Waggons und Lokomotiven auf Gleisabschnitt 1 wird demnach in dem Zähler *gleisBelegung[1]* gezählt.

Das Array *weichenBelegung* hat ebenfalls ein Element mehr, als Weichen in der Topologie vorhanden sind. Weiche a entspricht dem Index 1, b = 2, c = 3.

Die Anfangszählerstände für Fahraufgabe 1 sind:

- gleisBelegung[7] = 3 // Lok1 inkl. zwei Passagierwaggons
- gleisBelegung[2] = 3 // Drei Güterwaggons
- gleisBelegung[8] = 1 // Rangierlokomotive
- Alle anderen (auch weichenBelegung) = 0

### 5.4 Zugposition und Weichenstellung

Für jeden Zug auf der Strecke wird seine Position (als Nummer des Gleisabschnitts) gespeichert. Als Zug ist dabei alles definiert, was fahren kann: Es kann also eine Lok sein, oder aber eine Lok mit angehängten Waggons. Dabei spielt es keine Rolle wo sich die Lokomotive befindet (vor den Waggons, danach oder mittendrin).

Genau genommen, wird die Position des Zuganfangs in Fahrtrichtung gespeichert. Zum Beispiel: Schiebt die Rangierlokomotive die drei Güterwaggons vom Nebengleis auf das Abstellgleis, so bekommt der Zug(-Anfang) die Position 1 zugewiesen, sobald der erste Waggon vom Gleisabschnitt 2 über den ersten Sensor der Weiche a fährt.

Die Weichen selbst zählen also nicht als gesonderte Gleisabschnitte, d.h. so bald man in einen Weichenabschnitt fährt, gilt man als in dem Gleisabschnitt dahinter. Fährt nun Lok1 vom Gleisabschnitt 1 in den Weichenabschnitt a, wird die nächste Position (2 oder 7) anhand der aktuellen Weichenstellung ermittelt.

Weil Waggons nicht selbst fahren können und wir sicherstellen, dass zwei Lokomotiven nie aneinander gekoppelt werden, reicht es so viele Speicher für Zugpositionen zu definieren, wie es Loks gibt. Diese werden im Array *zugPosition* zusammengefasst. Zusätzlich zu der Position eines Zuges werden seine Fahrtrichtung und Geschwindigkeit in den Arrays *zugRichtung* und *zugGeschwindigkeit* gespeichert.

Die Stellung der Weichen wird in dem Array *weichenStellung* gespeichert, dabei bedeutet 0 = links und 1 = rechts (von vorne betrachtet, also vor der Aufspaltung der Schienen). Die Zuordnung der Buchstaben a-c zu den Indizes 0-2 ist genau so wie beim Array *weichenBelegung*.

### 5.5 Lokale Funktionen

| **Name** | **checkStreckenTopologie** |
|---|---|
| Beschreibung | Vergleicht die originale Streckentopologie mit der globalen Kopie, die für die Leitzentrale zur Verfügung gestellt wird. |
| Vorbedingung | Die originale (lokale) Streckentopologie ist erstellt und wurde zur Laufzeit nicht geändert. |
| Parameter | Keine |
| Rückgabe | 1, falls die Kopie nicht vom Original abweicht.<br>0, falls die Kopie abweicht. |
| Nachbedingung | Funktion hat keine Seiteneffekte, es werden also weder die Kopie der Streckentopologie noch das Original verändert. |

| **Name** | **checkSensorDaten** |
|---|---|
| Beschreibung | Überprüft die Sensordaten des S88-Treibers und aktualisiert das Streckenabbild entsprechend, wenn sie richtig sind. |
| Vorbedingung | Sensordaten sind im Shared Memory zwischen S88-Treiber und Befehlsvalidierung. |
| Parameter | Keine |
| Rückgabe | 1, falls Sensordaten gültig und konsistent sind.<br>0, falls Fehlerbit gesetzt ist oder die Sensordaten unlogisch sind. |
| Nachbedingung | Das interne Streckenabbild ist aktualisiert, wenn Sensordaten gültig und konstant waren. |

| **Name** | **sendSensorDaten** |
|---|---|
| Beschreibung | Leitet die S88-Sensordaten an die Leitzentrale weiter. |
| Vorbedingung | Gültige Sensordaten sind im Shared Memory zwischen S88-Treiber und Befehlsvalidierung. |
| Parameter | Keine |
| Rückgabe | 1, falls Sensordaten im Shared Memory zwischen Befehlsvalidierung und Leitzentrale leer waren.<br>0, falls sich noch alte Sensordaten im Shared Memory zwischen Befehlsvalidierung und Leitzentrale befanden. |
| Nachbedingung | Der Shared Memory zwischen S88-Treiber und Befehlsvalidierung wird geleert. Im Shared Memory zwischen Befehlsvalidierung und Leitzentrale stehen am Ende die neuen Sensordaten. |

| **Name** | **checkStreckenBefehl** |
|---|---|
| Beschreibung | Überprüft den Streckenbefehl der Leitzentrale zunächst auf syntaktische Korrektheit und dann darauf, dass er keinen unsicheren Zustand auslöst. |
| Vorbedingung | Ein Streckenbefehl ist im Shared Memory zwischen Leitzentrale und Befehlsvalidierung. |
| Parameter | Keine |
| Rückgabe | 1, falls der Streckenbefehl gültig und sicher ist.<br>0, falls der Streckenbefehl ungültig ist oder einen kritischen Zustand nach sich ziehen würde. |
| Nachbedingung | Funktion hat keine Seiteneffekte. |

| **Name** | **sendStreckenBefehl** |
|---|---|
| Beschreibung | Sendet den Streckenbefehl der Leitzentrale an die Ergebnisvalidierung. |
| Vorbedingung | Ein gültiger und sicherer Streckenbefehl ist im Shared Memory zwischen Leitzentrale und Befehlsvalidierung. |
| Parameter | Keine |
| Rückgabe | Keine |
| Nachbedingung | Der Streckenbefehl im Shared Memory zwischen Leitzentrale und Befehlsvalidierung wird geleert und die Bestätigung = 1 gesetzt.<br>Im Shared Memory zwischen Befehlsvalidierung und Ergebnisvalidierung steht am Ende der neue Streckenbefehl, vorhandene alte werden einfach überschrieben. |

Die hier beschriebenen Funktionen sind ausführlich dokumentiert, weil sie mit den externen Schnittstellen über einen gemeinsam genutzten Speicher zu kommunizieren. Darüber hinaus stehen weitere lokale Hilfsfunktionen zur Verfügung. Diese werden im Rahmen dieses Dokuments jedoch nicht so detailliert beschrieben.

Zu den Hilfsfunktionen zählen:

- static void copyStreckenBelegung(void): Erzeugt die Kopien der Gleisbelegung, der Weichenbelegung und der Zugposition.
- static void defineStreckenTopologie(void): Erstellt die lokale Variable Streckentopologie und erstellt daraus die globale Kopie BV_Streckentopologie.
- static boolean zugNebenSensor(byte sensorNr, byte *zugNr, byte *richtung): Überprüft anhand der Sensornummer welche Nachbarabschnitte zu diesem Sensor gehören (maximal drei).
- static boolean sensorNachbarn(byte sensorNr, byte *nextAbs, byte *prevAbs, byte *nSwitch, byte *pSwitch): Durchsuchen der Streckentopologie nach dem entsprechenden Sensor, Nachbarabschnitte des Sensors bestimmen.
- static boolean checkKritischerZustand(void): Überprüfen, ob ein kritischer Zustand auftreten kann. Zur Definition der kritischen Zustände siehe Abschnitt 6.2.
- static boolean weicheRichtig(byte zugPos, byte richtung, byte ziel, byte weiche): Überprüfen, ob die Weiche richtig ist, liegt keine Weiche vor wird die Funktion verlassen.
- static boolean zugFaehrtAufWeicheZu(byte weicheNr): Es werden nur Züge betrachtet, die sich auf den Gleisen bewegen. Liefert true zurück, sofern die Weiche mit der entsprechenden Nummer richtig ist.
- static void zielGleisUndWeiche(byte zugPos, byte richtung, byte *ziel, byte *weiche);
- static void sendNachricht(Zustand zustand, Fehler fehler): Sendet eine sechs Byte lange Nachricht.

## 5.6 Schnittstellenbeschreibung

### 5.6.1 Streckentopologie und Gleisbild

Die Befehlsvalidierung stellt der Leitzentrale Kopien der (statischen) Streckentopologie und einen Teil des aktuellen Gleisabbildes in Form von globalen Variablen zur Verfügung:

- *BV_streckentopologie:* Gleisabschnitt[]
- *BV_gleisBelegung*: int8[]
- *BV_weichenBelegung*: int8[]
- *BV_zugPosition*: int8[]

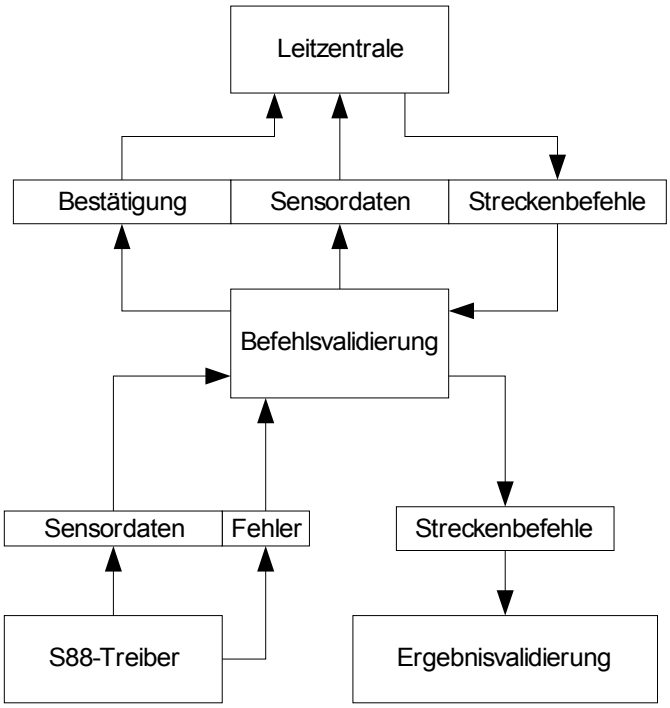
### 5.6.2 Shared-Memory

Die Befehlsvalidierung kommuniziert mit den Modulen Leitzentrale, Ergebnisvalidierung und S88-Treiber jeweils über Shared-Memories. (Vergleiche auch Kapitel 5.5)

Die Syntax der Shared-Memory-Felder ist im Software-Design, Kapitel 6.3 beschrieben.

Die Streckbefehle sind dabei drei Bytes lang und bestehen aus je ein Byte langen Befehlen für eine Lokomotive, eine Weiche und einen Entkoppler.

Von den Sensordaten wird (da wir keine Weichensensoren haben) nur die Gleispunktbelegung ausgewertet.




Verwendete globale Variablen (vergleiche auch Betriebsmittelverwaltung-Design) sind:

- BV_LZ_sensordaten : Sensordaten
- BV_LZ_bestaetigung : byte
- BV_EV_streckenbefehl : Streckenbefehl
- LZ_BV_streckenbefehl : Streckenbefehl
- S88_BV_sensordaten : Sensordaten

### 5.6.3 Auditing-System

Die Befehlsvalidierung sendet bei verschiedenen Gelegenheiten (meistens bei erkannten Fehlern) an das Auditing-System Nachrichten, die aus sechs Bytes bestehend. Die Bedeutung und Kodierung der einzelnen Bytes sind nachfolgend dargestellt.

| Byte | Bedeutung |
|---|---|
| 0 | Zustandscode (siehe unten) |
| 1 | Fehlercode (siehe unten) |
| 2 | *nextState* (interner Zustand, vgl. Kapitel 5.1) |
| 3 | *criticalStatecounter* (vgl. Kapitel 6.2) |
| 4 | Position von Zug Nr. 1 (Nr. des Gleisabschnitts) |
| 5 | Position von Zug Nr. 2 (Nr. des Gleisabschnitts) |

**Zustandscodes**

| Kodierung | Bedeutung |
|---|---|
| 0 | Programm befindet sich in der Hauptroutine workBV() |
| 1 | Programm befindet sich in der Funktion checkSensorDaten() |
| 2 | Programm befindet sich in der Funktion sendSensorDaten() |
| 3 | Programm befindet sich in der Funktion checkStreckenBefehl() |
| 4 | Programm befindet sich in der Funktion sensorNachbarn() |
| 5 | Programm befindet sich in der Funktion checkKritischerZustand() |

**Fehlercodes**

| Kodierung | Bedeutung |
|---|---|
| 0 | Kein Fehler |
| | **Allgemeine Fehler** |
| 1 | Sensordaten sind fehlerhaft |
| 2 | Kritischer Zustand wurde zu oft festgestellt |
| 3 | Kopie(n) der Streckentopologie manipuliert |
| 4 | Falschen internen Zustand erkannt |
| | **Sensordaten-Fehler** |
| 8 | Fehlerbyte in den Sensordaten gesetzt |
| 9 | Kein Zug neben dem aktivierten Sensor |
| 10 | Alte Sensordaten noch nicht von LZ verarbeitet |
| 11 | Sensor hat weder Nachfolger noch Vorgänger |
| | **Streckenbefehl-Fehler** |
| 16 | Syntaxfehler: Entkoppler-Nr. ungültig |
| 17 | Syntaxfehler: Weichen-Nr. ungültig |
| 18 | Entkoppeln, während ein schneller Zug auf diesem Gleisabschnitt ist |
| 19 | Weiche soll gestellt werden, die belegt ist |
| 20 | Weiche soll gestellt werden, die von einem anderen Zug angefahren wird |
| 21 | Lokbefehl: Mit Vollgas auf belegtes Gleis fahren |
| 22 | Lokbefehl: Weiche zum Ziel ist belegt |
| 23 | Lokbefehl: Weiche zum Ziel ist falsch gestellt |
| | **Kritischer-Zustand-Fehler** |
| 32 | Ein Zug fährt mit Vollgas in Richtung eines belegten Abschnitts |
| 33 | Zwei Züge in benachbarten Abschnitten fahren aufeinander zu |
| 34 | Ein Zug fährt auf eine für ihn falsch gestellte Weiche |
| 35 | Zu viele Waggons und Loks sind auf einem Abschnitt |

# 6 Dynamisches Verhalten

## 6.1 Auswerten der Sensordaten

Bei jedem Eintreffen von gültigen Sensordaten wird das Streckenabbild angepasst. Hat z.B. der Sensor zwischen den Gleisabschnitten 4 und 5 gemeldet, dass da etwas durchgefahren ist, so werden die Zähler *gleisBelegung[4]* und *gleisBelegung[5]* abhängig von der Richtung des entsprechenden Zuges inkrementiert oder dekrementiert. Sollte zu diesem Zeitpunkt aber keiner der Züge die Position 4 oder 5 gehabt haben, ist das ein logischer Fehler, der zu einem Not-Aus führt.

## 6.2 Prüfen auf kritische Zustände

Anschließend (also immer wenn das Streckenabbild verändert wurde) findet eine Prüfung auf unsichere Zustände statt. Nachfolgend ist definiert, wann ein unsicherer Zustand vorliegt:

- Ein Zug fährt (nicht im Rangierbetrieb) in Richtung eines belegten Gleisabschnitts.
  - Beispiel: Lok1 befindet sich auf Gleisabschnitt 1 und fährt mit Vollgas gegen den Uhrzeigersinn, Weiche a ist nach links gestellt. Auf Gleisabschnitt 2 stehen Waggons → kritischer Zustand!
- Zwei Züge befinden sich auf unmittelbar benachbarten Gleisabschnitten und mindestens einer der Züge fährt auf den anderen zu.
  - Beispiel 1: Lok2 fährt auf Gleisabschnitt 5 und Lok1 auf Gleisabschnitt 4 (beide vorwärts), dann fährt Lok1 auf Lok 2 zu → kritischer Zustand!
  - Beispiel 2: Lok2 ist auf Gleisabschnitt 1 und fährt in Richtung 8 (Weiche c ist nach rechts gestellt). Lok1 steht auf Gleisabschnitt 6. Dann fährt keiner der Züge auf den anderen zu → kein kritischer Zustand.
- Ein Zug fährt auf eine Weiche zu, die falsch gestellt ist.
  - Beispiel: Lok2 fährt von Gleis 8 in Richtung 1 (Weiche c ist nach rechts gestellt) und Lok1 fährt von Gleisabschnitt 6 auf Gleisabschnitt 1. In diesem Fall ist die Weiche für die Lok1 falsch gestellt → kritischer Zustand!
- Die maximale Anzahl von Waggons und Loks auf einem Gleisabschnitt überschritten wird. Für beide Fahraufgaben ist diese maximale Anzahl = 4.

Wenn ein kritischer Zustand erkannt wird, soll nicht sofort ein Not-Aus erfolgen. Stattdessen soll die Leitzentrale die Chance bekommen, darauf zu reagieren (in dem sie z.B. einen Zug stoppt). Die Befehlsvalidierung merkt sich, dass ein kritischer Zustand vorliegt und gibt die Kontrolle an die Betriebsmittelverwaltung zurück.

So lange ein kritischer Zustand vorliegt, inkrementiert die BV bei jedem Aufruf des Moduls den Zähler *criticalStateCounter* um eins. So bald dieser Zähler den Wert 5 erreicht, wird ein Not-Aus generiert. Wenn die Leitzentrale es vorher schafft, den kritischen Zustand aufzulösen, wird der *criticalStateCounter* wieder auf 0 gesetzt, was bedeutet, dass kein kritischer Zustand vorliegt.

### 6.3 Überprüfen der Streckenbefehle

Nach einer syntaktischen Prüfung der Befehle (siehe Schnittstellenbeschreibung) wird geprüft, ob die Befehle ausgeführt werden können ohne damit einen kritischen Zustand hervorzurufen. Es gelten folgende einfache Regeln:

- Ein Zug darf nur fahren, wenn der nächste Gleisabschnitt in Fahrtrichtung frei ist, eine evtl. vorhandene Weiche auf dem Weg dahin frei ist und die Weiche richtig gestellt ist.
- Ein Zug darf auch auf einen belegten Gleisabschnitt fahren, wenn er im Rangierbetrieb ist, also mit der Geschwindigkeit fährt, die für das Ankuppeln vorgesehen ist.
- Eine Weiche darf nur geschaltet werden, wenn der Weichenabschnitt frei ist und kein Zug von einem angrenzenden Abschnitt auf die Weiche zu fährt.

Wenn ein Streckenbefehl unzulässig ist, wird noch kein Not-Aus generiert, vielmehr wird der Befehl einfach nicht ausgeführt. Beim nächsten Mal überprüft die Befehlsvalidierung dann erneut, ob der Befehl ausgeführt werden kann.

### 6.4 Überprüfen der Streckentopologie

Die Felder der originalen Streckentopologie und der Kopie werden Element für Element verglichen. Falls sie irgendwo nicht übereinstimmen, wird ein Not-Aus generiert.