

# Testspezifikation 'Befehlsvalidierung'

Für das studentische Projekt *Sichere Eisenbahnsteuerung*

<b>Datum</b>	17.06.2010
<b>Quelle</b>	Dokumente\04_Test\04.01_Testspezifikation\04.01.00_PDFs
<b>Autoren</b>	O. Bohn
<b>Version</b>	1.0
<b>Status</b>	freigegeben

---

## 1 Historie

Version	Datum	Autor	Bemerkung
0.1	26.05.2010	O. Bohn	Initialisierung der Testspezifikation: Anlegen der Testfälle 5 bis 9.
0.2	27.05.2010	O. Bohn	Anlegen der Testfälle 10 bis 16.
0.3	03.06.2010	O. Bohn	Überarbeiten der bereits angelegten Testfälle 1 bis 13, Entfernen von Rechtschreibfehlern. Anlegen der fehlenden Testfälle.
0.4	08.06.2010	F. Geber	Ausbessern von Rechtschreibfehlern, Anderes im Review-Dokument
1.0	17.06.2010	O. Bohn	Ergänzen der Testskript - Beschreibung in Kapitel 16 und 17.

---

## 2 Inhaltsverzeichnis

<b>1 Historie.....</b>	<b>2</b>
<b>2 Inhaltsverzeichnis.....</b>	<b>3</b>
<b>3 Identifikation des Testobjekts.....</b>	<b>7</b>
<b>4 Testziele.....</b>	<b>8</b>
<b>5 Testfall 1 „Initialisierung der Befehlsvalidierung“.....</b>	<b>9</b>
5.1 Identifikation des Testobjektes.....	9
5.2 Test-Identifikation.....	9
5.3 Testfallbeschreibung.....	9
5.4 Testskript.....	9
5.5 Test-Protokoll.....	10
<b>6 Testfall 2 „checkStreckenTopologie“.....</b>	<b>11</b>
6.1 Identifikation des Testobjektes.....	11
6.2 Test-Identifikation.....	11
6.3 Testfallbeschreibung.....	11
6.4 Testskript.....	11
6.5 Test-Referenz.....	11
6.6 Test-Protokoll.....	12
<b>7 Testfall 3 „sendSensorDaten“.....</b>	<b>13</b>
7.1 Identifikation des Testobjektes.....	13
7.2 Test-Identifikation.....	13
7.3 Testfallbeschreibung.....	13
7.4 Testskript.....	13
7.5 Test Referenz.....	14
7.6 Test-Protokoll.....	14
<b>8 Testfall 4 „checkStreckenBefehl“.....</b>	<b>15</b>
8.1 Identifikation des Testobjektes.....	15
8.2 Test-Identifikation.....	15
8.3 Testfallbeschreibung.....	15
8.4 Testskript.....	15

---

---

8.5 Test-Protokoll.....	16
<b>9 Testfall 5 „sendStreckenBefehl“.....</b>	<b>17</b>
9.1 Identifikation des Testobjektes.....	17
9.2 Test-Identifikation.....	17
9.3 Testfallbeschreibung.....	17
9.4 Testskript.....	17
9.5 Test Referenz.....	18
9.6 Test-Protokoll.....	18
<b>10 Testfall 6 „checkSensorDaten“.....</b>	<b>19</b>
10.1 Identifikation des Testobjektes.....	19
10.2 Test-Identifikation.....	19
10.3 Testfallbeschreibung.....	19
10.4 Testskript.....	19
10.5 Test-Protokoll.....	20
<b>11 Testfall 8 „zugNebenSensor“.....</b>	<b>21</b>
11.1 Identifikation des Testobjektes.....	21
11.2 Test-Identifikation.....	21
11.3 Testfallbeschreibung.....	21
11.4 Testskript.....	21
11.5 Test-Protokoll.....	21
<b>12 Testfall 9 „sensorNachbarn“.....</b>	<b>22</b>
12.1 Identifikation des Testobjektes.....	22
12.2 Test-Identifikation.....	22
12.3 Testfallbeschreibung.....	22
12.4 Testskript.....	22
12.5 Test-Referenz.....	22
12.6 Test-Protokoll.....	23
<b>13 Testfall 10 „sendNachricht“.....</b>	<b>24</b>
13.1 Identifikation des Testobjektes.....	24
13.2 Test-Identifikation.....	24
13.3 Testfallbeschreibung.....	24

---

13.4 Testskript.....	24
13.5 Test-Protokoll.....	24
<b>14 Testfall 11 „checkKritischerZustand“ .....</b>	<b>25</b>
14.1 Identifikation des Testobjektes.....	25
14.2 Test-Identifikation.....	25
14.3 Testfallbeschreibung.....	25
14.4 Testskript.....	25
14.5 Test-Protokoll.....	26
<b>15 Testfall 12 „weicheRichtig“ .....</b>	<b>27</b>
15.1 Identifikation des Testobjektes.....	27
15.2 Test-Identifikation.....	27
15.3 Testfallbeschreibung.....	27
15.4 Testskript.....	27
15.5 Test-Protokoll.....	27
<b>16 Testfall 13 „zugFährtaufWeicheZu“ .....</b>	<b>28</b>
16.1 Identifikation des Testobjektes.....	28
16.2 Test-Identifikation.....	28
16.3 Testfallbeschreibung.....	28
16.4 Testskript.....	28
16.5 Test-Protokoll.....	28
<b>17 Testfall 14 „ZielGleisundWeiche“ .....</b>	<b>29</b>
17.1 Identifikation des Testobjektes.....	29
17.2 Test-Identifikation.....	29
17.3 Testfallbeschreibung.....	29
17.4 Testskript.....	29
17.5 Test-Protokoll.....	29
<b>18 Testfall 15 „void workBV“ .....</b>	<b>30</b>
18.1 Identifikation des Testobjektes.....	30
18.2 Test-Identifikation.....	30
18.3 Testfallbeschreibung.....	30
18.4 Testskript.....	30

---

18.5 Test-Referenz.....	31
18.6 Test-Protokoll.....	32
<b>19 Auswertung.....</b>	<b>33</b>

---

### 3 Identifikation des Testobjekts

Es wird der Programmcode zum Softwaremodul „Befehlsvalidierung“ getestet:

- Befehlsvalidierung.c (Version 1.1, Repository-Nr. 213)
- Befehlsvalidierung.h (Version 1.1, Repository-Nr. 213)

Das Modul „Befehlsvalidierung“ ist in der Sicherheitsschicht der Software angeordnet.

Das Modul prüft die von der Leitzentrale (Anwendungsschicht-Modul) kommenden Streckenbefehle auf deren Gültigkeit und leitet sie an die Ergebnisvalidierung weiter. Darüber hinaus reicht sie die vom S88-Treiber kommenden Sensordaten an die Leitzentrale weiter falls dieser keinen Fehler gemeldet hat. Meldet der S88-Treiber jedoch einen Fehler, schaltet die Befehlsvalidierung das System über den Not-Aus-Treiber ab.

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 Testziele

Der Test des Software-Moduls „Befehlsvalidierung“ soll sicherstellen, dass die im Rahmen der Einleitung dargestellte Funktionalität vom Modul vollständig und fehlerfrei zur Verfügung gestellt wird. Hieraus resultieren im Detail die folgenden zwei globalen Testziele:

- Durch das Erreichen des ersten Testziels ist sicherzustellen, ob ein gültiger Streckenbefehl von der Befehlsvalidierung an die Ergebnisvalidierung weitergeleitet wird.
- Als weiteres Testziel ist zu überprüfen, ob die S88-Sensordaten, sofern kein Fehler gemeldet wurde, an die Leitzentrale gesendet werden. Ist hingegen ein Fehler aufgetreten, muss durch den Test überprüft werden, ob tatsächlich eine Abschaltung über den Not-Aus-Treiber erfolgt.

Zum Erreichen dieser Testziele ist die einwandfreie Funktionalität der innerhalb des Moduls definierten Hilfsfunktionen zu überprüfen.



---

## 5 Testfall 1 „Initialisierung der Befehlsvalidierung“

### 5.1 Identifikation des Testobjektes

Siehe Kapitel 3

### 5.2 Test-Identifikation

Testname: Test\_Befehlsvalidierung\_Init

Verzeichnisse:

Testskripts: Google Code → Dokumente → 04\_Tests → 04.02\_Testskript → 04.02.01\_Befehlsvalidierung\_Init

Testprotokolle: Google Code → Dokumente → 04\_Tests → 04.03\_Testprotokolle → 04.03.01\_Befehlsvalidierung\_Init

### 5.3 Testfallbeschreibung

Dieser Testfall dient dazu, zu überprüfen, ob das Modul Befehlsvalidierung durch den Aufruf der void initBV(void) Schnittstelle aus der Betriebsmittelverwaltung heraus der Spezifikation entsprechend initialisiert wird.

Das heißt, dass durch einen Aufruf der Funktion static void defineStreckenTopologie (void) die Streckentopologie bei der Initialisierung angelegt wird. Des Weiteren ist die Kopie der Streckentopologie für die Leitzentrale durch einen Aufruf von static void copyStreckenBelegung(void) bereitzustellen.

Durch diesen Testfall wird das Verhalten von den Funktionen static void defineStreckenTopologie (void) und static void copyStreckenBelegung(void) mit überprüft. Aus diesem Grund wird für beide Funktionen kein separater Testfall angelegt.

### 5.4 Testskript

Um den im vorangegangenen Abschnitt beschriebenen Testfall zu realisieren, wird ein Testskript angefertigt. Mit Hilfe dieses Skripts wird die void initBV (void) Funktion des Moduls Befehlsvalidierung aufgerufen. Dies führt implizit zu einem Aufruf der Funktionen static void defineStreckenTopologie (void) und static void copyStreckenBelegung(void).

Im ersten Schritt ist zu überprüfen, ob beide obengenannten Funktionen innerhalb der initBV () aufgerufen werden. Ist dies sichergestellt, muss überprüft werden, ob static void defineStreckenTopologie (void) und static void copyStreckenBelegung(void) vollständig abgearbeitet wurden.

Zu diesem Zweck werden im Anschluss die globalen Variablen BV\_Streckentopologie, BV\_weichenBelegung, BV\_streckenBelegung, BV\_gleisBelgung und BV\_zugPosition überprüft. Wurden die obengenannten Funktionen vollständig und der Spezifikation entsprechend ausgeführt, müssen diese Variablen einen definierten Inhalt haben. Dieser wird mit Hilfe einer for - Schleife durchlaufen. Der Inhalt der Variablen wird mit dem erwarteten Ergebnis verglichen.

---

Sollte eine Abweichung auftreten, wird diese auf der Konsole ausgegeben. Es sollen sowohl die Variable, in der die Abweichung aufgetreten ist, als auch die interne Stelle (Index) ausgegeben werden.

Dies wird mit folgendem Test-Skript realisiert:

siehe 'Google Code → 04\_Test → 04.02\_Testskripts → 04.02.01\_Befehlsvalidierung\_Init'

### **5.5 Test-Protokoll**

Das Konsolen-Ergebnis wird in das Dokument 'Protokoll\_Test\_Befehlsvalidierung\_Init' kopiert und diese Datei im Ordner 'Google Code → Dokumente → 04\_Tests → 04.03\_Testprotokolle → 04.03.01\_Befehlsvalidierung' abgelegt.

---

## 6 Testfall 2 „checkStreckenTopologie“

### 6.1 Identifikation des Testobjektes

siehe Kapitel 3

### 6.2 Test-Identifikation

Testname: Test\_Befehlsvalidierung\_checkStreckenTopologie

Verzeichnisse:

Testskripts: Google Code → Dokumente → 04\_Tests → 04.02\_Testskript → 04.02.01\_Befehlsvalidierung\_checkStreckenTopologie

Testprotokolle: Google Code → Dokumente → 04\_Tests → 04.03\_Testprotokolle → 04.03.01\_Befehlsvalidierung\_checkStreckenTopologie

### 6.3 Testfallbeschreibung

Dieser Testfall dient zur Überprüfung der lokalen Funktion `static boolean checkStreckenTopologie (void)` im Modul `Befehlsvalidierung`. Diese vergleicht die originale Streckentopologie (`streckentopologie`) mit der globalen Kopie, die für die Leitzentrale (`BV_Streckentopologie`) erzeugt wird. Das Verhalten dieser Funktion nach einem Aufruf ist im entsprechenden Modul-Design spezifiziert. Der Rückgabewert ist gleich `True`, sofern die Kopie nicht vom Original abweicht und gleich `False`, wenn eine Abweichung vorliegt.

Im Rahmen dieses Testfalls wird überprüft, ob die implementierte Funktion diese Funktionalität zur Verfügung stellt. Zu diesem Zweck wird die Funktion im ersten Schritt mit übereinstimmenden Streckentopologien aufgerufen. Im zweiten Schritt wird die Funktion mit voneinander abweichenden Variablen aufgerufen.

### 6.4 Testskript

Um den im vorangegangenen Abschnitt beschriebenen Testfall zu realisieren, wird ein Testskript angefertigt. Dieses dient sowohl dem Aufruf der Funktion, als auch zur Bereitstellung der entsprechenden Variablen `streckentopologie` und `BV_Streckentopologie`.

Der Rückgabewert der Funktion wird schließlich durch eine einfache `if`-Abfrage mit dem zu erwartenden Rückgabewert verglichen. Tritt eine Abweichung auf, so wird dies als Fehler auf der Konsole ausgegeben.

Dies wird mit folgendem Test-Skript realisiert:

siehe 'Google Code → 04\_Test → 04.02\_Testskripts → 04.02.01\_Befehlsvalidierung\_checkStreckenTopologie'

### 6.5 Test-Referenz

Genutzt wird für diesen Testfall die im Modul vordefinierte Streckentopologie, aus der heraus ebenfalls die Kopie `BV_Streckentopologie` erzeugt wird. Um die Funktion mit voneinander abweichenden Variablen aufzurufen, wird diese Streckentopologie verändert. Aus Gründen der Übersichtlichkeit wird die Streckentopologie an dieser Stelle nicht aufgeführt. Die folgende

Tabelle zeigt die einzelnen Aufrufe der Funktion, wobei dargestellt ist, inwieweit die beiden Variablen voneinander abweichen.

Nr.:	Testfall:	Erwartetes Ergebnis:
1.	Streckentopologie = BV_Streckentopologie	True
2.	Streckentopologie <> BV_Streckentopologie: Abweichungen an einer Stelle, in einem Gleisabschnitt (Bsp.: Gleisabschnitt 2: next1)	False
3.	Streckentopologie <> BV_Streckentopologie: Abweichungen in jedem Gleisabschnitt (1 bis 9 jeweils next1)	False
4.	Streckentopologie <> BV_Streckentopologie: Abweichungen in jedem Gleisabschnitt (jeweils nr, next1, next2, prev1, prev2, nextSwitch, prevSwitch, nextSensor, prevSensor)	False

## 6.6 Test-Protokoll

Das Konsolen-Ergebnis wird in das Dokument 'Protokoll\_Test\_Befehlsvalidierung\_checkStreckenTopologie' kopiert und diese Datei im Ordner 'Google Code → Dokumente → 04\_Tests → 04.03\_Testprotokolle → 04.03.01\_Befehlsvalidierung' abgelegt.

---

## 7 Testfall 3 „sendSensorDaten“

### 7.1 Identifikation des Testobjektes

Siehe Kapitel 3

### 7.2 Test-Identifikation

Testname: Test\_Befehlsvalidierung\_sendSensorDaten

Verzeichnisse:

Testskripts: Google Code → Dokumente → 04\_Tests → 04.02\_Testskript → 04.02.01\_Befehlsvalidierung\_sendSensorDaten

Testprotokolle: Google Code → Dokumente → 04\_Tests → 04.03\_Testprotokolle → 04.03.01\_Befehlsvalidierung\_sendSensorDaten

### 7.3 Testfallbeschreibung

Dieser Testfall dient zur Überprüfung der lokalen Funktion `static boolean sendSensorDaten (void)` im Modul Befehlsvalidierung. Es handelt sich hierbei um eine lokale Funktion, die die Sensordaten vom S88 - Treiber an die Leitzentrale weiterleitet. Das bedeutet, die Sensordaten werden in den Shared-Memory zwischen Befehlsvalidierung und Leitzentrale geschrieben.

Es ist zu überprüfen, ob das im Modul-Design spezifizierte Verhalten auch dem Verhalten des Moduls entspricht. Solange die Sensordaten im Shared-Memory leer sind, liefert die Funktion `True` eins als Rückgabewert. Falls sich an dieser Stelle jedoch noch alte Sensordaten befinden, nimmt die Funktion den Rückgabewert `False` an.

Nach dem die Funktion erfolgreich ausgeführt wurde, muss der Shared-Memory zwischen Befehlsvalidierung und S88 - Treiber leer sein und der Shared-Memory zwischen Befehlsvalidierung und der Leitzentrale mit den neuen Sensordaten gefüllt sein.

### 7.4 Testskript

Um den im vorangegangenen Abschnitt beschriebenen Testfall zu realisieren, wird ein Testskript angefertigt. Aus diesem Testskript heraus wird die Funktion `static boolean sendSensorDaten (void)` unter Variation der unten aufgeführten Variablen aufgerufen.

Um zu überprüfen, ob die Funktionalität mit der Spezifikation übereinstimmt, werden die Variablen manuell durch das Testskript vorgegeben:

- S88\_BV\_sensordaten.Byte0
- S88\_BV\_sensordaten.Byte1
- BV\_LZ\_sensordaten.Byte0
- BV\_LZ\_sensordaten.Byte1

Entsprechend der manuell vorgegebenen Variablen kann somit der zu erwartende Rückgabewert mit dem tatsächlichen Rückgabewert über eine if - Abfrage verglichen werden. Stimmen beide miteinander überein, liegt kein Fehler vor. Andernfalls weicht die Funktionalität von der Spezifikation ab und es wird ein Fehler auf der Konsole ausgegeben. Darüber hinaus ist zu berücksichtigen, dass bei einem Rückgabewert False zusätzlich die Methode sendNachricht () aufgerufen wird. Um sicherzugehen, dass die auch dieser Aufruf erfolgt, wird neben einer Überprüfung des Rückgabewerts ebenfalls überprüft, ob eine Nachricht gesendet wurde.

Dies wird mit folgendem Test-Skript realisiert:

siehe 'Google Code → 04\_Test → 04.02\_Testskripts → 04.02.01\_Befehlsvalidierung\_sendSensorDaten'

## 7.5 Test Referenz

In der Tabelle sind einige Variationen der betrachteten Variablen sowie die dazugehörigen zu erwartenden Rückgabewerte der Funktion dargestellt. Im Rahmen dieses Testfalls ist die Funktion mit den Werten für die Variablen aufzurufen.

S88_BV_sensor daten.Byte0	S88_BV_sensor daten.Byte1	BV_LZ_sensord aten.Byte0	BV_LZ_sensordat en.Byte1	Erwarteter Rückgabewert:
Leer	Leer	-	-	True
Leer	nicht Leer	Leer	Leer	True
nicht Leer	Leer	Leer	Leer	True
Leer	nicht Leer	nicht Leer	Leer	False
nicht Leer	Leer	nicht Leer	Leer	False
nicht Leer	Leer	Leer	nicht Leer	False
Leer	nicht Leer	Leer	nicht Leer	False
Leer	nicht Leer	Leer	Leer	True
nicht Leer	Leer	Leer	Leer	True

## 7.6 Test-Protokoll

Das Konsolen-Ergebnis wird in das Dokument 'Protokoll\_Test\_Befehlsvalidierung\_sendSensorDaten' kopiert und diese Datei im Ordner 'Google Code → Dokumente → 04\_Tests → 04.03\_Testprotokolle → 04.03.01\_Befehlsvalidierung' abgelegt.

---

## 8 Testfall 4 „checkStreckenBefehl“

### 8.1 Identifikation des Testobjektes

Siehe Kapitel 3

### 8.2 Test-Identifikation

Testname: Test\_Befehlsvalidierung\_checkStreckenBefehl

Verzeichnisse:

Testskripts: Google Code → Dokumente → 04\_Tests → 04.02\_Testskript → 04.02.01\_Befehlsvalidierung\_checkStreckenBefehl

Testprotokolle: Google Code → Dokumente → 04\_Tests → 04.03\_Testprotokolle → 04.03.01\_Befehlsvalidierung\_checkStreckenBefehl

### 8.3 Testfallbeschreibung

Dieser Testfall dient zur Überprüfung der lokalen Funktion static boolean checkStreckenBefehl (void) im Modul Befehlsvalidierung. Es handelt sich hierbei um eine lokale Funktion, die zunächst die Streckenbefehle der Leitzentrale auf syntaktische Korrektheit überprüft und im nächsten Schritt sicherstellt, dass die Streckenbefehle nicht zu unsicheren Zuständen auf der Strecke führen.

Dieser Testfall dient dazu zu überprüfen, ob das im Modul-Design spezifizierte Verhalten auch dem Verhalten des Moduls entspricht. Das heißt, sofern ein Streckenbefehl im Shared-Memory zwischen Leitzentrale und Befehlsvalidierung ist und dieser gültig bzw. sicher ist, liefert die Funktion als Rückgabewert eine True. Ist der Streckbefehl jedoch ungültig oder führt zu einem unsicheren Zustand, so liefert die Funktion eine False als Rückgabewert.

### 8.4 Testskript

Um den im vorangegangenen Abschnitt beschriebenen Testfall zu realisieren, wird ein Testskript angefertigt. Dieses Testskript ruft die betrachtete Funktion unter Variation der unten aufgeführten Variablen auf. Der zu erwartende Rückgabewert wird dann mit dem tatsächlichen Rückgabewert verglichen, dies kann wiederum durch eine if - Abfrage erfolgen. Treten Abweichungen zwischen dem erwarteten und dem tatsächlichen Rückgabewert auf, wird dies als Fehler auf der Konsole ausgegeben.

- LZ\_BV\_streckenbefehl.Entkoppler
- LZ\_BV\_streckenbefehl.Weiche
- LZ\_BV\_streckenbefehl.Lok

---

Dies wird mit folgendem Test-Skript realisiert:

siehe 'Google Code → 04\_Test → 04.02\_Testskripts → 04.02.01\_Befehlsvalidierung\_checkStreckenBefehl'

### **8.5 Test-Protokoll**

Das Konsolen-Ergebnis wird in das Dokument 'Protokoll\_Test\_Befehlsvalidierung\_checkStreckenBefehl' kopiert und diese Datei im Ordner 'Google Code → Dokumente → 04\_Tests → 04.03\_Testprotokolle → 04.03.01\_Befehlsvalidierung' abgelegt.



---

## 9 Testfall 5 „sendStreckenBefehl“

### 9.1 Identifikation des Testobjektes

Siehe Kapitel 3

### 9.2 Test-Identifikation

Testname: Test\_Befehlsvalidierung\_sendStreckenBefehl

Verzeichnisse:

Testskripts: Google Code → Dokumente → 04\_Tests → 04.02\_Testskript → 04.02.01\_Befehlsvalidierung\_sendStreckenBefehl

Testprotokolle: Google Code → Dokumente → 04\_Tests → 04.03\_Testprotokolle → 04.03.01\_Befehlsvalidierung\_sendStreckenBefehl

### 9.3 Testfallbeschreibung

Dieser Testfall dient zur Überprüfung der lokalen Funktion `static void sendStreckenBefehl (void)` des Moduls Befehlsvalidierung. Es handelt sich hierbei um eine lokale Funktion, die den Streckenbefehl der Leitzentrale an die Ergebnisvalidierung sendet.

Es ist zu überprüfen, ob das im Modul-Design spezifizierte Verhalten auch dem Verhalten des Moduls entspricht. Nach erfolgreicher Ausführung der Funktion muss der Streckenbefehl im Shared-Memory zwischen Leitzentrale und Befehlsvalidierung geleert sowie die dazugehörige Bestätigung auf eins gesetzt werden. Des Weiteren muss im Shared-Memory zwischen Ergebnisvalidierung und Befehlsvalidierung der neue Streckenbefehl vorhanden sein.

### 9.4 Testskript

Um den im vorangegangenen Abschnitt beschriebenen Testfall zu realisieren, wird ein Testskript angefertigt. Dieses Testskript ruft die Funktion unter Variation der entsprechenden Variablen auf. Abhängig vom Wert der jeweiligen Variablen wird das Verhalten der Funktion beobachtet und mit dem erwarteten Verhalten verglichen. Zu betrachten sind die folgenden drei Variablen:

- LZ\_BV\_streckenbefehl.Entkoppler
- LZ\_BV\_streckenbefehl.Weiche
- LZ\_BV\_streckenbefehl.Lok

Dies wird mit folgendem Test-Skript realisiert:

siehe 'Google Code → 04\_Test → 04.02\_Testskripts → 04.02.01\_Befehlsvalidierung\_sendStreckenBefehl'

### 9.5 Test Referenz

In der Tabelle sind einige Variationen der betrachteten Variablen sowie das dazugehörige zu erwartende Verhalten der Funktion dargestellt. Im Rahmen dieses Testfalls ist die Funktion mit den Werten für die Variablen aufzurufen.

LZ_BV_streckenbefe hl.Entkoppler	LZ_BV_streckenbefe hl.Weiche	LZ_BV_streckenbefe hl.Lok	Verhalten:
Leer	Leer	Leer	keine Aktion ausgeführt
nicht Leer	Leer	Leer	LZ_BV_streckenbefe hl.Entkoppler; LZ_BV_streckenbefe hl.Weiche; LZ_BV_streckenbefe hl.Lok           leeren; BV_LZ_bestatigung auf eins setzen;  BV_EV_streckenbefe hl==LZ_BV_strecken befehl
Leer	nicht Leer	Leer	
Leer	Leer	nicht Leer	
nicht Leer	nicht Leer	nicht Leer	

### 9.6 Test-Protokoll

Das Konsolen-Ergebnis wird in das Dokument 'Protokoll\_Test\_Befehlsvalidierung\_sendStreckenBefehl' kopiert und diese Datei im Ordner 'Google Code → Dokumente → 04\_Tests → 04.03\_Testprotokolle → 04.03.01\_Befehlsvalidierung' abgelegt.

---

## 10 Testfall 6 „checkSensorDaten“

### 10.1 Identifikation des Testobjektes

Siehe Kapitel 3

### 10.2 Test-Identifikation

Testname: Test\_Befehlsvalidierung\_checkSensorDaten

Verzeichnisse:

Testskripts: Google Code → Dokumente → 04\_Tests → 04.02\_Testskript → 04.02.01\_Befehlsvalidierung\_checkSensorDaten

Testprotokolle: Google Code → Dokumente → 04\_Tests → 04.03\_Testprotokolle → 04.03.01\_Befehlsvalidierung\_checkSensorDaten

### 10.3 Testfallbeschreibung

Dieser Testfall dient zur Überprüfung der lokalen Funktion `static boolean checkSensorDaten (void)` des Moduls Befehlsvalidierung. Es handelt sich hierbei um eine lokale Funktion, die die Sensordaten des S88 - Treibers überprüft und das Streckenabbild entsprechend aktualisiert. Dies geschieht im Fall, dass die Daten richtig sind.

Es ist zu überprüfen, ob das im Modul-Design spezifizierte Verhalten auch dem Verhalten des Moduls entspricht. Sind die Sensordaten gültig und konsistent, liefert die Funktion als Rückgabewert eine `True`. Ist jedoch das Fehlerbit gesetzt, oder die Daten sind unlogisch, liefert die Funktion als Rückgabewert eine `False`.

### 10.4 Testskript

Um den im vorangegangenen Abschnitt beschriebenen Testfall zu realisieren, wird ein Testskript angefertigt. Aus diesem Testskript heraus wird die Funktion aufgerufen. Um sicherzustellen, dass die Funktionalität der Spezifikation im Modul-Design entspricht, müssen unter anderem die folgenden Variablen im Testskript gesetzt werden. Die aus der Wahl dieser Variablen resultierende Rückgabe der Funktion wird mit der zu erwartenden Rückgabe verglichen.

- `S88_BV_sensordaten.Fehler`
- `S88_BV_sensordaten.Byte0`
- `S88_BV_sensordaten.Byte1`

Im ersten Schritt wird die Variable `S88_BV_sensordaten.Fehler` (Fehler - Byte) auf einen von null verschiedenen Wert gesetzt. Mit Hilfe einer `if` - Abfrage wird dann überprüft, ob die Funktion tatsächlich `False` zurück liefert und die weitere Bearbeitung abbricht. Daran anschließend wird `S88_BV_sensordaten.Fehler` auf null gesetzt und die beiden Variablen `S88_BV_sensordaten.Byte0` und `S88_BV_sensordaten.Byte1` werden auf leer gesetzt. In diesem Fall muss die Funktion ein `True` zurückgeben, auch dies wird mit Hilfe einer `if` -

---

Abfrage überprüft. Tritt ein von der Spezifikation abweichendes Verhalten auf, wird dies auf der Konsole ausgegeben.

Daran anschließend werden die Variablen so initialisiert, dass die for - Schleife im Anschluss an die zwei if - Abfragen im Modul ausgeführt wird. Diese dient der Überprüfung der Streckenbefehle und muss deshalb entsprechend der Spezifikation arbeiten, damit es nicht zu unsicheren Zuständen auf den Gleisen kommt. Folgende Initialwerte werden exemplarisch zum Testen genutzt:

- S88\_BV\_sensordaten.Fehler = Leer
- S88\_BV\_sensordaten.Byte0 <> Leer
- S88\_BV\_sensordaten.Byte1 <> Leer

Die Funktion checkSensorDaten (void) greift an dieser Stelle auf andere lokale Hilfsfunktionen des Moduls Befehlsvalidierung zurück. Es ist somit vorab zu überprüfen, ob die Funktionen zugNebenSensor () und sensorNachbarn () der Spezifikation entsprechend arbeiten.

Liefert ein Aufruf von zugNebenSensor () als Rückgabewert False, liefert auch die Funktion checkSensorDaten ein False zurück. Analoges Verhalten zeigt sich bei einem Aufruf von sensorNachbarn.

Dies wird mit folgendem Test-Skript realisiert:

siehe 'Google Code → 04\_Test → 04.02\_Testskripts → 04.02.01\_Befehlsvalidierung\_checkSensorDaten'

### 10.5 Test-Protokoll

Das Konsolen-Ergebnis wird in das Dokument 'Protokoll\_Test\_Befehlsvalidierung\_checkSensorDaten' kopiert und diese Datei im Ordner 'Google Code → Dokumente → 04\_Tests → 04.03\_Testprotokolle → 04.03.01\_Befehlsvalidierung' abgelegt.

---

## 11 Testfall 8 „zugNebenSensor“

### 11.1 Identifikation des Testobjektes

Siehe Kapitel 3

### 11.2 Test-Identifikation

Testname: Test\_Befehlsvalidierung\_zugNebenSensor

Verzeichnisse:

Testskripts: Google Code → Dokumente → 04\_Tests → 04.02\_Testskript → 04.02.01\_Befehlsvalidierung\_zugNebenSensor

Testprotokolle: Google Code → Dokumente → 04\_Tests → 04.03\_Testprotokolle → 04.03.01\_Befehlsvalidierung\_zugNebenSensor

### 11.3 Testfallbeschreibung

Dieser Testfall dient zur Überprüfung der lokalen Funktion `static boolean zugNebenSensor(byte sensorNr, byte *zugNr, byte *richtung)` des Moduls Befehlsvalidierung. Es handelt sich hierbei um eine lokale Funktion, die überprüft, ob sich Züge auf benachbarten Abschnitten befinden.

Es ist zu überprüfen, ob das spezifizierte Verhalten auch dem Verhalten des Moduls entspricht. Die Rückgabewerte der Funktion können entweder den Wert `False` oder `True` annehmen. Dies hängt jeweils von der Position des jeweiligen Zugs sowie der Position des angesprochenen Sensors ab. Jeder Sensor besitzt maximal drei Nachbarabschnitte. Befindet sich einer der Züge innerhalb dieser Nachbarabschnitte, liefert die Funktion eine `True` als Rückgabewert. Andernfalls wird eine `False` zurückgegeben.

### 11.4 Testskript

Um den im vorangegangenen Abschnitt beschriebenen Testfall zu realisieren, wird ein Testskript angefertigt. Aus diesem Testskript heraus wird die Funktion aufgerufen. Der Funktion werden unterschiedliche Sensornummern übergeben, sowie unterschiedliche Positionen der Züge. In Abhängigkeit von den Eingangswerten wird der Rückgabewert der Funktion mit dem erwarteten Rückgabewert verglichen. Treten Abweichungen auf, werden diese als Fehler auf der Konsole ausgegeben.

Dies wird mit folgendem Test-Skript realisiert:

siehe 'Google Code → 04\_Test → 04.02\_Testskripts → 04.02.01\_Befehlsvalidierung\_zugNebenSensor'

### 11.5 Test-Protokoll

Das Konsolen-Ergebnis wird in das Dokument 'Protokoll\_Test\_Befehlsvalidierung\_zugNebenSensor' kopiert und diese Datei im Ordner 'Google Code → Dokumente → 04\_Tests → 04.03\_Testprotokolle → 04.03.01\_Befehlsvalidierung' abgelegt.

---

## 12 Testfall 9 „sensorNachbarn“

### 12.1 Identifikation des Testobjektes

Siehe Kapitel 3

### 12.2 Test-Identifikation

Testname: Test\_Befehlsvalidierung\_sensorNachbarn

Verzeichnisse:

Testskripts: Google Code → Dokumente → 04\_Tests → 04.02\_Testskript → 04.02.01\_Befehlsvalidierung\_sensorNachbarn

Testprotokolle: Google Code → Dokumente → 04\_Tests → 04.03\_Testprotokolle → 04.03.01\_Befehlsvalidierung\_sensorNachbarn

### 12.3 Testfallbeschreibung

Dieser Testfall dient zur Überprüfung der lokalen Funktion `static boolean sensorNachbarn (byte sensorNr, byte *nextAbs, byte *prevAbs, byte *nSwitch, byte *pSwitch)` des Moduls Befehlsvalidierung. Diese lokale Funktion dient dazu, die Nachbarabschnitte eines Sensors zu bestimmen. Der Rückgabewert beträgt entweder `False` oder `True`.

### 12.4 Testskript

Um den im vorangegangenen Abschnitt beschriebenen Testfall zu realisieren, wird ein Testskript angefertigt. Aus diesem Testskript heraus wird die Funktion aufgerufen.

Zu variieren ist dabei der Übergabeparameter `byte sensorNr`. Bei vorher bekannter Sensornummer können die Rückgabewerte der Funktion für den jeweils nächsten bzw. vorangegangenen Abschnitt mit den erwarteten Werten überprüft werden.

Dies wird mit folgendem Test-Skript realisiert:

siehe 'Google Code → 04\_Test → 04.02\_Testskripts → 04.02.01\_Befehlsvalidierung\_sensorNachbarn'

### 12.5 Test-Referenz

In der Tabelle sind einige Variationen der betrachteten Variablen sowie das dazugehörige zu erwartende Verhalten der Funktion dargestellt. Im Rahmen dieses Testfalls ist die Funktion mit den Werten für die Variablen aufzurufen.

---

Sensornummer:	prevAbs:	nextAbs:	nSwitch:	pSwitch:	Erwarteter Rückgabewert:
1	1	0	a	-	True
7	5	6	-	a	True
10	0	7		-	True
13	8	9	-	-	True

### 12.6 Test-Protokoll

Das Konsolen-Ergebnis wird in das Dokument 'Protokoll\_Test\_Befehlsvalidierung\_sensorNachbarn' kopiert und diese Datei im Ordner 'Google Code → Dokumente → 04\_Tests → 04.03\_Testprotokolle → 04.03.01\_Befehlsvalidierung' abgelegt.

---

## 13 Testfall 10 „sendNachricht“

### 13.1 Identifikation des Testobjektes

Siehe Kapitel 3

### 13.2 Test-Identifikation

Testname: Test\_Befehlsvalidierung\_sendNachricht

Verzeichnisse:

Testskripts: Google Code → Dokumente → 04\_Tests → 04.02\_Testskript → 04.02.01\_Befehlsvalidierung\_sendNachricht

Testprotokolle: Google Code → Dokumente → 04\_Tests → 04.03\_Testprotokolle → 04.03.01\_Befehlsvalidierung\_sendNachricht

### 13.3 Testfallbeschreibung

Dieser Testfall dient zur Überprüfung der lokalen Funktion static sendNachricht (Zustand Zustand, Fehler fehler) des Moduls Befehlsvalidierung. Diese lokale Funktion dient dazu, die Nachrichten an das Auditing System zu schicken. Hierfür werden die aus sechs Byte bestehenden Nachrichten zu Beginn erstellt, bevor die Funktion sendMsg (MODUL\_BV, nachricht) aufgerufen wird.

Es ist zu überprüfen, ob das spezifizierte Verhalten auch dem Verhalten des Moduls entspricht, das heißt ob die Nachrichten tatsächlich erstellt bzw. gesendet werden.

### 13.4 Testskript

Um den im vorangegangenen Abschnitt beschriebenen Testfall zu realisieren, wird ein Testskript angefertigt. Um sicherzustellen, dass die Funktion den Anforderungen entsprechend arbeitet, wird die Funktion aus dem Testskript heraus aufgerufen. Des Weiteren werden die Werte für Zustand und Fehler im Testskript festgelegt. Die gesendete Nachricht kann daraufhin mit der zu erwartenden Nachricht verglichen werden. Treten Abweichungen auf, werden diese auf der Konsole als Fehlermeldungen ausgegeben.

Dies wird mit folgendem Test-Skript realisiert:

siehe 'Google Code → 04\_Test → 04.02\_Testskripts → 04.02.01\_Befehlsvalidierung\_sendNachricht'

### 13.5 Test-Protokoll

Das Konsolen-Ergebnis wird in das Dokument 'Protokoll\_Test\_Befehlsvalidierung\_sendNachricht' kopiert und diese Datei im Ordner 'Google Code → Dokumente → 04\_Tests → 04.03\_Testprotokolle → 04.03.01\_Befehlsvalidierung' abgelegt.



---

## 14 Testfall 11 „checkKritischerZustand“

### 14.1 Identifikation des Testobjektes

Siehe Kapitel 3

### 14.2 Test-Identifikation

Testname: Test\_Befehlsvalidierung\_ checkKritischerZustand

Verzeichnisse:

Testskripts: Google Code → Dokumente → 04\_Tests → 04.02\_Testskript → 04.02.01\_ Befehlsvalidierung\_ checkKritischerZustand

Testprotokolle: Google Code → Dokumente → 04\_Tests → 04.03\_Testprotokolle → 04.03.01\_ Befehlsvalidierung\_ checkKritischerZustand

### 14.3 Testfallbeschreibung

Dieser Testfall dient zur Überprüfung der lokalen Funktion `static boolean checkKritischerZustand (void)` des Moduls `Befehlsvalidierung`. Diese lokale Funktion überprüft, ob auf den Gleisanlagen ein kritischer Zustand entstehen könnte. Der Rückgabewert der Funktion kann die Werte `True` bzw. `False` annehmen und entspricht dem Wert der booleschen Variablen `kritisch` innerhalb der Funktion.

Es ist zu überprüfen, ob das spezifizierte Verhalten auch dem Verhalten des Moduls entspricht, das heißt, ob kritische Zustände richtig erkannt werden und der Rückgabewert entsprechend angepasst wird.

### 14.4 Testskript

Um den im vorangegangenen Abschnitt beschriebenen Testfall zu realisieren, wird ein Testskript angefertigt. Aus diesem Testskript heraus wird die Funktion aufgerufen. Festgelegt werden dabei Werte für `zugPosition[]`, `zugGeschwindigkeit[]` und `zugRichtung[]`. Diese werden während der Bearbeitung der Funktion ausgewertet und dienen als Grundlage zur Überprüfung auf kritische Zustände.

Bei Testen sind somit mehrere Fälle zu unterscheiden. Im ersten Fall werden der Funktion Werte übergeben, die nicht zu einem kritischen Zustand führen. Der Rückgabewert muss für diesen Fall `False` sein (`kritisch = False`). Dies wird mit Hilfe einer `if` - Abfrage im Testskript überprüft. Im zweiten Fall werden Werte übergeben, die offensichtlich zu einem kritischen Zustand führen würden. Auch hier wird der Rückgabewert (`True`) durch eine `if` - Abfrage überprüft.

Treten Abweichungen von erwarteten Ergebnis auf, werden diese als Fehlermeldungen auf der Konsole ausgegeben.

Dies wird mit folgendem Test-Skript realisiert:

siehe 'Google Code → 04\_Test → 04.02\_Testskripts → 04.02.01\_Befehlsvalidierung\_ checkKritischerZustand'

---

#### **14.5 Test-Protokoll**

Das Konsolen-Ergebnis wird in das Dokument 'Protokoll\_Test\_Befehlsvalidierung\_checkKritischerZustand' kopiert und diese Datei im Ordner 'Google Code → Dokumente → 04\_Tests → 04.03\_Testprotokolle → 04.03.01\_Befehlsvalidierung' abgelegt.

---

## 15 Testfall 12 „weicheRichtig“

### 15.1 Identifikation des Testobjektes

Siehe Kapitel 3

### 15.2 Test-Identifikation

Testname: Test\_Befehlsvalidierung\_ weicheRichtig

Verzeichnisse:

Testskripts: Google Code → Dokumente → 04\_Tests → 04.02\_Testskript → 04.02.01\_Befehlsvalidierung\_ weicheRichtig

Testprotokolle: Google Code → Dokumente → 04\_Tests → 04.03\_Testprotokolle → 04.03.01\_Befehlsvalidierung\_ weicheRichtig

### 15.3 Testfallbeschreibung

Dieser Testfall dient zur Überprüfung der lokalen Funktion `static boolean weicheRichtig (byte zugPos, byte richtung, byte ziel, byte weiche)`. Diese lokale Funktion überprüft, ob sich die Weichen in der richtigen Position für die Überfahrt der Züge befinden.

Mit Hilfe dieses Testfalls ist sicherzustellen, dass das Verhalten dem der Spezifikation entspricht. Befindet sich eine Weiche nicht in der richtigen Position muss die Funktion `False` zurückliefern, andernfalls `True`.

### 15.4 Testskript

Um den im vorangegangenen Abschnitt beschriebenen Testfall zu realisieren, wird ein Testskript angefertigt. Aus diesem Testskript heraus wird die Funktion aufgerufen. Übergeben werden dabei unterschiedliche Werte für `zugPos`, `richtung`, `ziel` und `weiche`. Die Übergabeparameter bilden die Grundlage für die in der Funktion erfolgende Überprüfung. Beim Testen sind zwei Fälle zu unterscheiden. Im ersten Fall werden der Funktion korrekte Werte übergeben, das heißt, die Weiche befindet sich in der richtigen Stellung. Im zweiten Fall werden fehlerhafte Werte übergeben. Der Rückgabewert wird jeweils mit dem erwarteten Wert verglichen. Treten Abweichungen auf, werden diese als Fehler auf der Konsole ausgegeben.

Dies wird mit folgendem Test-Skript realisiert:

siehe 'Google Code → 04\_Test → 04.02\_Testskripts → 04.02.01\_Befehlsvalidierung\_ weicheRichtig'

### 15.5 Test-Protokoll

Das Konsolen-Ergebnis wird in das Dokument 'Protokoll\_Test\_Befehlsvalidierung\_ weicheRichtig' kopiert und diese Datei im Ordner 'Google Code → Dokumente → 04\_Tests → 04.03\_Testprotokolle → 04.03.01\_Befehlsvalidierung' abgelegt.

---

## 16 Testfall 13 „zugFährtaufWeicheZu“

### 16.1 Identifikation des Testobjektes

Siehe Kapitel 3

### 16.2 Test-Identifikation

Testname: Test\_Befehlsvalidierung\_ zugFährtaufWeicheZu

Verzeichnisse:

Testskripts: Google Code → Dokumente → 04\_Tests → 04.02\_Testskript → 04.02.01\_Befehlsvalidierung\_ zugFährtaufWeicheZu

Testprotokolle: Google Code → Dokumente → 04\_Tests → 04.03\_Testprotokolle → 04.03.01\_Befehlsvalidierung\_ zugFährtaufWeicheZu

### 16.3 Testfallbeschreibung

Dieser Testfall dient zur Überprüfung der lokalen Funktion `static boolean zugFaehrtAufWeicheZu (byte weicheNr)` des Moduls Befehlsvalidierung.

### 16.4 Testskript

Um den im vorangegangenen Abschnitt beschriebenen Testfall zu realisieren, wird ein Testskript angefertigt. Aus diesem Testskript heraus wird die Funktion aufgerufen, wobei dabei die Weichennummer variiert wird. Abhängig von der Position, der Richtung und der Weichennummer wird der Rückgabewert mit dem zu erwartenden Rückgabewert überprüft. Tritt hierbei eine Abweichung auf, wird diese auf der Konsole ausgegeben.

Dies wird mit folgendem Test-Skript realisiert:

siehe 'Google Code → 04\_Test → 04.02\_Testskripts → 04.02.01\_Befehlsvalidierung\_ zugFährtaufWeicheZu'

### 16.5 Test-Protokoll

Das Konsolen-Ergebnis wird in das Dokument 'Protokoll\_Test\_Befehlsvalidierung\_ zugFährtaufWeicheZu' kopiert und diese Datei im Ordner 'Google Code → Dokumente → 04\_Tests → 04.03\_Testprotokolle → 04.03.01\_Befehlsvalidierung' abgelegt.

---

## 17 Testfall 14 „ZielGleisundWeiche“

### 17.1 Identifikation des Testobjektes

Siehe Kapitel 3

### 17.2 Test-Identifikation

Testname: Test\_Befehlsvalidierung\_ZielGleisundWeiche

Verzeichnisse:

Testskripts: Google Code → Dokumente → 04\_Tests → 04.02\_Testskript → 04.02.01\_Befehlsvalidierung\_ZielGleisundWeiche

Testprotokolle: Google Code → Dokumente → 04\_Tests → 04.03\_Testprotokolle → 04.03.01\_Befehlsvalidierung\_ZielGleisundWeiche

### 17.3 Testfallbeschreibung

Dieser Testfall dient zur Überprüfung der lokalen Funktion `static void zielGleisUndWeiche (byte zugPos, byte richtung, byte *ziel, byte *weiche)`. Die Funktion dient dazu, zu einer gegebenen Zugposition sowie einer Richtung, in der sich der Zug bewegt, das Zielgleis und die nächste Weiche zu bestimmen.

### 17.4 Testskript

Um den im vorangegangenen Abschnitt beschriebenen Testfall zu realisieren, wird ein Testskript angefertigt. Aus diesem Testskript heraus wird die Funktion aufgerufen. Übergeben werden dabei unterschiedliche Zugpositionen bzw. Richtungen. Die daraus ermittelten Ziele sind mit dem erwarteten Ergebnis zu überprüfen.

Dies wird mit folgendem Test-Skript realisiert:

siehe 'Google Code → 04\_Test → 04.02\_Testskripts → 04.02.01\_Befehlsvalidierung\_ZielGleisundWeiche'

### 17.5 Test-Protokoll

Das Konsolen-Ergebnis wird in das Dokument 'Protokoll\_Test\_Befehlsvalidierung\_ZielGleisundWeiche' kopiert und diese Datei im Ordner 'Google Code → Dokumente → 04\_Tests → 04.03\_Testprotokolle → 04.03.01\_Befehlsvalidierung' abgelegt.

---

## 18 Testfall 15 „void workBV“

### 18.1 Identifikation des Testobjektes

Siehe Kapitel 3

### 18.2 Test-Identifikation

Testname: Test\_Befehlsvalidierung\_workBV

Verzeichnisse:

Testskripts: Google Code → Dokumente → 04\_Tests → 04.02\_Testskript → 04.02.01\_Befehlsvalidierung\_workBV

Testprotokolle: Google Code → Dokumente → 04\_Tests → 04.03\_Testprotokolle → 04.03.01\_Befehlsvalidierung\_workBV

### 18.3 Testfallbeschreibung

Dieser Testfall dient zur Überprüfung Funktion void workBV (void) der Befehlsvalidierung, welche von der Betriebsmittelverwaltung aufgerufen und die gesamte Funktionalität der Befehlsvalidierung zur Verfügung stellt.

Es ist zu überprüfen, dass ein Aufruf der void workBV (void) Schnittstelle die vorher spezifizierte Funktionalität tatsächlich zur Verfügung stellt. Die einzelnen innerhalb dieser Funktion aufzurufenen lokalen Hilfsfunktionen wurden bereits in den vorangegangenen beschriebenen Testfällen auf deren einwandfreie Funktion hin überprüft. Nun gilt es sicherzustellen, dass die einzelnen Funktion wirklich aufgerufen werden und entsprechend der Spezifikation zusammenarbeiten.

### 18.4 Testskript

Um den im vorangegangenen Abschnitt beschriebenen Testfall zu realisieren, wird ein Testskript angefertigt.

Aufgebaut ist die Funktion void workBV(void) aus einer switch - case Abfrage mit vier Blöcken sowie einem default - Block. Die Unterscheidung zwischen den einzelnen Blöcken erfolgt anhand der Variablen nextState.

- nextState 0: Wenn keine neuen Sensordaten eingetroffen, werden keine Aktionen ausgeführt.
- nextState 1: Gleisbild auf kritische Zustände prüfen
- nextState 2: Streckenbefehl überprüfen und (wenn OK) an Ergebnisvalidierung senden
- nextState 3: Wird ausgeführt, wenn Kopien der Streckentopologie manipuliert wurden

Der Wert der Variablen nextState wird vor der Ausführung der Funktion gesetzt. Zum Testen der Funktion wird dies durch das Testskript vorgenommen. Dieses setzt nextState der Reihe nach auf null, eins, zwei und drei. Somit kann der Reihe nach überprüft werden, ob die Anweisungen im jeweiligen case - Block ausgeführt bzw. richtig ausgeführt werden. Nach der

Ausführung aller Anweisungen in einem case - Block wird überprüft, ob die Funktion entsprechend der Spezifikation reagiert hat. Dies lässt sich an dem neu zugewiesenen Wert für nextState oder an einem Aufruf von emergency\_off() feststellen. Weichen erwartetes und tatsächliches Ergebnis voneinander ab, wird ein Fehler auf der Konsole ausgegeben.

Um den im vorangegangenen Abschnitt beschriebenen Testfall zu realisieren, wird ein Testskript angefertigt.

Dies wird mit folgendem Test-Skript realisiert:

siehe 'Google Code → 04\_Test → 04.02\_Testskripts → 04.02.01\_Befehlsvalidierung\_workBV'

### 18.5 Test-Referenz

In der folgenden Tabelle sind die Werte der Variablen dargestellt, mit denen die einzelnen case - Blöcke aufgerufen werden sollen. Hierdurch wird sichergestellt, dass alle Ausweisungen überprüft werden.

Nr.:	nextState	Variablen - Belegung:	Ergebnis:
1	0	S88_BV_sensordaten.Byte0 == Leer && S88_BV_sensordaten.Byte1== Leer	nextState = 2
2	0	S88_BV_sensordaten.Byte0 <> Leer && S88_BV_sensordaten.Byte1== Leer; checkSensorDaten() == False    sendSensorDaten == False	Aufruf von emergency_off ()
3	0	S88_BV_sensordaten.Byte0 <> Leer && S88_BV_sensordaten.Byte1== Leer; checkSensorDaten() <> False    sendSensorDaten <> False	nextState = 1
4	1	checkKritischerZustand() == False; criticalStateCounter > MAX_KRITISCH	Aufruf von emergency_off()
5	1	checkKritischerZustand() <> False;	nextState = 2; criticalStateCounter = 0;
6	2	checkStreckenBefehl() == True	nextState = 0
7	3	checkStreckentopologie() == False	Aufruf von emergency_off()
8	3	checkStreckentopologie() <> False	nextState = 2

---

**18.6 Test-Protokoll**

Das Konsolen-Ergebnis wird in das Dokument 'Protokoll\_Test\_Befehlsvalidierung\_workBV' kopiert und diese Datei im Ordner 'Google Code → Dokumente → 04\_Tests → 04.03\_Testprotokolle → 04.03.01\_Befehlsvalidierung' abgelegt.



---

## 19 Auswertung

Die Auswertung der Testfälle wird im Anschluss an die Durchführung erstellt.