Testspezifikation-Ergebnisvalidierung

Für das studentische Projekt *Sichere Eisenbahnsteuerung*

Datum	31.05.2010
Quelle	Dokumente → 04_Test → 04.01_Testspezifikation
Autor	Felix Geber
Version	0.0
Status	zum Review freigegeben

1 Historie

Version	Datum	Autor	Bemerkung
0.0	31.05.2010	Felix Geber	Initialisierung der Testspezifikation; Anlegen von Testfall 1 bis Testfall 6

2 Inhaltsverzeichnis

3 Identifikation des Testobjekts

Es wird der Programmcode zum Softwaremodul „Ergebnisvalidierung“ getestet:

- Ergebnisvlaidierung.c (Version 2.0, Repository-Nr. 204)
- Ergebnisvlaidierung.h (Version 1.0, Repository-Nr. 204)

Die Ergebnisvalidierung (EV) hat die Aufgabe, die von der Anwendung erzeugten Streckenbefehle auf Korrektheit zu prüfen. Das Modul Ergebnisvalidierung befindet sich in der Sicherheitsschicht und steht in direkter Kommunikation mit fünf Modulen:

- Die EV bekommt die Streckenbefehle von der *Befehlsvalidierung (BV)*.
- Der *SSC-Treiber* vergleicht die Streckenbefehle der beiden Mikrocontroller miteinander. Das Ergebnis des Vergleichs wird der EV über den Shared-Memory mitgeteilt.
- Meldet der SSC-Treiber einen Fehler zurück, wird eine Meldung an das *Auditing-System (AS)* geschickt. Es werden ebenfalls Warnungen und Informationen an das AS gesandt.
- Bei einer Fehlermeldung wird weiterhin ein Not-Aus über den *Not-Aus-Treiber* ausgelöst.
- Meldet das Modul SSC-Treiber hingegen keinen Fehler zurück, werden die Streckenbefehle an den *RS232-Treiber* gesendet.

Die EV wird von der Betriebsmittelverwaltung (BMV) initialisiert und die Methode *void workEV(void)* in einer Endlosschleife aufgerufen.

In den folgenden Testfällen werden einige Variablen mit dem Wert 'X' gekennzeichnet. Hierbei verdeutlicht das 'X', dass der Zustand der Variable egal ist (z.B. bei einem Parameter des Typs boolean: TRUE oder FALSE).

4 Testziele

Der Test des Software-Moduls 'Ergebnisvalidierung' soll sicherstellen, dass keine fehlerhaften Streckenbefehle an den RS232-Treiber gesandt werden. Die Kommunikation zwischen den anderen vier Modulen müssen ebenfalls geprüft werden. Genauer gesagt, muss geprüft werden, ob die EV die Streckenbefehle der Befehlsvalidierung bekommt. Weiterhin muss das Auditing-System Informationen, Warnungen und Fehler sicher zugeschickt bekommen. Die Kommunikation zwischen SSC-Treiber und EV muss sichergestellt und das Auslösen des Not-Aus-Signals muss über den Not-Aus-Treiber garantiert werden.

Dies dient dem Gesamtziel, eine gemäß Pflichtenheft (Kapitel 6) fehlerfreie Fahraufgabe, also eine ohne das Erreichen unsicher Zustände, auszuführen.

5Testfall 1 „Kommunikationsprüfung mit RS232- und SSC-Treiber“

5.1 Identifikation des Testobjektes

siehe Kapitel 3

5.2 Test-Identifikation

Testname: Test_Ergebnisvalidierung_Kommunikationsprüfung

Verzeichnisse:

Testskripts: Google Code → Dokumente → 04_Tests → 04.02_Testskript → 04.02.03_Ergebnisvalidierung

Testprotokolle: Google Code → Dokumente → 04_Tests → 04.03_Testprotokolle → 04.03.03_Ergebnisvalidierung

5.3 Testfallbeschreibung

Das Modul EV prüft beim Durchlauf der *void workEV(void)* zuerst, ob es Kommunikationsprobleme mit dem RS232-Treiber oder dem SSC-Treiber gab. Dies geschieht durch die Methode *static boolean checkForCommunicationErrors(void)*. Lag in den beiden Treiber-Modulen kein Fehler vor, wird FALSE zurückgegeben, anderenfalls TRUE.

In diesem Testfall soll die Kommunikationsabfrage geprüft werden. Ziel ist es, eine 100%-ige Quellcode-Abdeckung zu realisieren.

5.4 Testskript

Der Quellcode der Methode *static boolean checkForCommunicationErrors(void)* besteht grundsätzlich aus zwei Kommunikationsüberprüfungen:

- Einerseits wird geprüft, ob der Fehleranteil des Shared-Memory vom RS232-Treiber bzw. vom SSC-Treiber zurückgesetzt wurde. Hat der RS232-Treiber oder der SSC-Treiber noch einen Fehler, wird eine Fehlermeldung an das Auditing-System AS geschickt, ein Not-Aus über den Not-Aus-Treiber eingeleitet und die Methode *static boolean checkForCommunicationErrors(void)* liefert ein TRUE zurück.
- Andererseits wird kontrolliert, ob beim letzten Modulaufruf der Streckenbefehl in den Shared-Memory zum RS232-Treiber bzw. SSC-Treiber geschrieben wurde. Es darf maximal dreimal zu keiner Streckenbefehlsschreibung in den Shared-Memory kommen. Wurde dreimal nacheinander nicht in diesen reingeschrieben und wird im aktuellen Methodendurchlauf der Streckenbefehl nicht resetet, wird ein Fehler an das Auditing-System gesendet, ein Not-Aus eingeleitet und TRUE zurückgegeben. Wurde hingegen nur ein- oder zweimal nicht in den Shared-Memory geschrieben, geht nur eine Warnung raus. Ist bei beiden Treibern der Zähler kleiner drei, wird die Methode *static boolean checkForCommunicationErrors(void)* mit dem Rückgabewert FALSE

verlassen. Wenn der Zähler Null ist, wird die komplette Bedingung umgangen. Nur wenn beide Zähler (also der vom RS232-Treiber und vom SSC-Treiber) mit Null belegt sind, liefert *static boolean checkForCommunicationErrors(void)* ein FALSE zurück, ohne das Senden einer Nachricht.

Bei der ersten Kommunikationsüberprüfung muss sichergestellt werden, dass die Nachricht mit dem richtigen Informationsgehalt an das Auditing-System AS geschickt, ein Not-Aus eingeleitet, und TRUE zurückgeliefert wird.

In der zweiten Prüfung muss eine komplette Codeabdeckung erfolgen. Die Zähler beider Treiber müssen Werte zwischen '0' und '4' zugewiesen bekommen und die Streckenbefehle müssen einmal resetet sein, und einmal nicht.

Gibt die Methode *static boolean checkForCommunicationErrors(void)* TRUE zurück, muss geprüft werden, ob *void workEV(void)* verlassen wird.

Dies wird mit folgendem Test-Skript realisiert:

siehe 'Google Code → 04_Test → 04.02_Testskripts → 04.02.03_Ergebnisvalidierung'

5.5 Testreferenz

Während des Testdurchlaufs werden die in Tabelle 1 dargestellten Schritte ausgeführt. Im Quellcode werden zuerst die Zustände des RS232-Treibers geprüft und anschließend die des SSC-Treibers. Die Kommunikationsprüfung KP 2 kann nur erreicht werden, wenn *RS232_EV_streckenbefehl.Fehler*=0 und *SSC_EV_streckenbefehl.Fehler*=0 sind (siehe Teilprüfung TP 1).

KP	TP	Zustand	Erwartete Ausgabe
1	1	RS232_EV_streckenbefehl.Fehler=0, SSC_EV_streckenbefehl.Fehler=0	Es wird in der Kommunikationsprüfung KP 2 fortgefahren.
1	2	RS232_EV_streckenbefehl.Fehler=0, SSC_EV_streckenbefehl.Fehler=1	• Nachricht an AS: (4,3,0,X,0) • Not-Aus-Signal auslösen • Rückgabe: TRUE
1	3	RS232_EV_streckenbefehl.Fehler=1, SSC_EV_streckenbefehl.Fehler=0	• Nachricht an AS: (5,3,0,X,0) • Not-Aus-Signal auslösen • Rückgabe: TRUE

KP	TP	Zustand	Erwartete Ausgabe
1	4	RS232_EV_streckenbefehl.Fehler=1, SSC_EV_streckenbefehl.Fehler=1	• Nachricht an AS: (5,3,0,X,0) • Not-Aus-Signal auslösen • Rückgabe: TRUE • es darf kein weitere Not-Aus eingeleitet werden bzw. die Nachricht (4,3,0,X,0) an das AS gesendet werden, da die Methode vorher verlassen werden musste
2	1	counterRS232=0, counterSSC=0	Rückgabewert: FALSE
2	2	counterRS232=0, counterSSC=1...2, EV_SSC_streckenbefehl != {LEER,LEER,LEER,0}	• Nachricht an AS: (2,2,1...2,0,0) • counterSSC von 1 auf 2 bzw. von 2 auf 3 • Rückgabewert: FALSE
2	3	counterRS232=0, counterSSC=1...2, EV_SSC_streckenbefehl = {LEER,LEER,LEER,0}	• EV_SSC_streckenbefehl=oldSSC? • oldSSC={LEER,LEER,LEER,0}? • counterSSC=0? • Rückgabewert: FALSE
2	4	counterRS232=0, counterSSC=3, EV_SSC_streckenbefehl != {LEER,LEER,LEER,0}	• Nachricht an AS: (2,3,4,0,0) • Not-Aus-Signal auslösen • Rückgabewert: TRUE
2	5	counterRS232=0, counterSSC=3, EV_SSC_streckenbefehl = {LEER,LEER,LEER,0}	• EV_SSC_streckenbefehl=oldSSC? • oldSSC={LEER,LEER,LEER,0}? • counterSSC=0? • Rückgabewert: FALSE
2	6	counterRS232=0, counterSSC=4, EV_SSC_streckenbefehl != {LEER,LEER,LEER,0}	Fehler, da MAXSSCCOUNTER+1=5, und dies ist für das AS eine nicht definierte Nachricht

KP	TP	Zustand	Erwartete Ausgabe
2	7	counterRS232=0, counterSSC=4, EV_SSC_streckenbefehl = {LEER,LEER,LEER,0}	Dieser Zustand darf nicht erreicht werden, da counterSSC<=3
2	8-14	counterRS232=1...2, counterSSC=X, EV_RS232_streckenbefehl != {LEER,LEER,LEER,0}	• Nachricht an AS: (3,2,1...2,0,0) • counterRS232 von 1 auf 2 bzw. von 2 auf 3 • weiter zur Überprüfung des SSC-Treibers für die KP 2 / TP 1 bis TP 7
2	15-21	counterRS232=1...2, counterSSC=X, EV_RS232_streckenbefehl = {LEER,LEER,LEER,0}	• EV_RS232_streckenbefehl=oldRS232? • oldRS232={LEER,LEER,LEER,0}? • counterRS232=0? • weiter zur Überprüfung des SSC-Treibers für die KP 2 / TP 1 bis TP 7
2	22	counterRS232=3, counterSSC=X, EV_RS232_streckenbefehl != {LEER,LEER,LEER,0}	• Nachricht an AS: (3,3,4,0,0) • Not-Aus-Signal auslösen • Rückgabewert: TRUE Die Methode *static boolean checkForCommunicationErrors(void)* muss verlassen werden, wodurch der Zustand von counterSSC egal ist.
2	23-29	counterRS232=3, counterSSC=X, EV_RS232_streckenbefehl = {LEER,LEER,LEER,0}	• EV_RS232_streckenbefehl=oldRS232? • oldRS232={LEER,LEER,LEER,0}? • counterRS232=0? • weiter zur Überprüfung des SSC-Treibers für die KP 2 / TP 1 bis TP 7

Tabelle 1: Erwartete Ausgabe für den Testfall 1

5.6 Test-Protokoll

Das Konsolen-Ergebnis wird in das Dokument 'Protokoll_Test_Ergebnisvalidierung' kopiert und diese Datei im Ordner 'Google Code → Dokumente → 04_Tests → 04.03_Testprotokolle → 04.03.03_Ergebnisvalidierung' abgelegt.

6 Testfall 2 „Prüfung des internen Streckenbefehls“

6.1 Identifikation des Testobjektes

siehe Kapitel 3

6.2 Test-Identifikation

Testname: Test_Ergebnisvalidierung_Streckenbefehl-Intern

Verzeichnisse:

Testskripts: Google Code → Dokumente → 04_Tests → 04.02_Testskript → 04.02.03_Ergebnisvalidierung

Testprotokolle: Google Code → Dokumente → 04_Tests → 04.03_Testprotokolle → 04.03.03_Ergebnisvalidierung

6.3 Testfallbeschreibung

Im Anschluss an die Kommunikationsprüfung wird der interne Streckenbefehl, also der aus dem Shared-Memory von der Befehlsvalidierung BV, überprüft und in den Shared-Memory zum SSC-Treiber geschrieben.Die Überprüfung findet in der Form statt, indem man kontrolliert, ob ein neuer Streckenbefehl von der BV in den Shared-Memory geschrieben wurde.

6.4 Testskript

Es wird getestet, ob der Aufruf der Funktion *static void processInternalStreckenbefehl(void)* mit unterschiedlich übergebenen Variablen stets richtige Aktionen veranlasst. Involvierte Variablen sind hierbei:

- BV_EV_streckenbefehl (Streckenbefehl)
- oldSSC (Streckenbefehl)
- EV_SSC_streckenbefehl (Streckenbefehl)

Die oben aufgeführten Variablen müssen unterschiedlich miteinander kombiniert werden, um wiederum eine 100%-ige Codeabdeckung zu gewährleisten.

Dies wird mit folgendem Test-Skript realisiert:

siehe 'Google Code → 04_Test → 04.02_Testskripts → 04.02.03_Ergebnisvalidierung'

6.5 Testreferenz

Während des Testdurchlaufs werden die in Tabelle 2 dargestellten Schritte ausgeführt.

TP	Zustand	Erwartete Ausgabe
1	BV_EV_streckenbefehl= {LEER,LEER,LEER,0}, oldSSC ={X,X,X,X}, EV_SSC_streckenbefehl ={X,X,X,X}	• return
2	BV_EV_streckenbefehl !={LEER,LEER,LEER,0}, oldSSC !={LEER,LEER,LEER,0}, EV_SSC_streckenbefehl ={X,X,X,X}	• internerStreckenbefehl =BV_EV_streckenbefehl? • BV_EV_streckenbefehl ={LEER,LEER,LEER,0}? • isBVNew=TRUE? • Nachricht an AS: (2,3,internerStreckenbefehl.Lok, internerStreckenbefehl.Weiche, internerStreckenbefehl.Entkoppler) • Not-Aus-Signal auslösen • return
3	BV_EV_streckenbefehl != {LEER,LEER,LEER,0}, oldSSC ={LEER,LEER,LEER,0}, EV_SSC_streckenbefehl ={LEER,LEER,LEER,0}	• internerStreckenbefehl =BV_EV_streckenbefehl? • BV_EV_streckenbefehl ={LEER,LEER,LEER,0}? • isBVNew=TRUE? • Nachricht an AS: (2,1,internerStreckenbefehl.Lok, internerStreckenbefehl.Weiche, internerStreckenbefehl.Entkoppler) • EV_SSC_streckenbefehl =internerStreckenbefehl? • return

TP	Zustand	Erwartete Ausgabe
4	BV_EV_streckenbefehl ={LEER,LEER,LEER,0}, oldSSC ={LEER,LEER,LEER,0}, EV_SSC_streckenbefehl !={LEER,LEER,LEER,0}	• internerStreckenbefehl =BV_EV_streckenbefehl? • BV_EV_streckenbefehl ={LEER,LEER,LEER,0}? • isBVNew=TRUE? • oldSSC=internerStreckenbefehl? • Nachricht an AS: (2,2,0,0,0) • counterSSC=1? • return

Tabelle 2: Erwartete Ausgabe für den Testfall 2

6.6 Test-Protokoll

Das Konsolen-Ergebnis wird in das Dokument 'Protokoll_Test_Ergebnisvalidierung' kopiert und diese Datei im Ordner 'Google Code → Dokumente → 04_Tests → 04.03_Testprotokolle → 04.03.03_Ergebnisvalidierung' abgelegt.

7 Testfall 3 „Prüfung des externen Streckenbefehls“

7.1 Identifikation des Testobjektes

siehe Kapitel 3

7.2 Test-Identifikation

Testname: Test_Ergebnisvalidierung_Streckenbefehl-Extern

Verzeichnisse:

Testskripts: Google Code → Dokumente → 04_Tests → 04.02_Testskript → 04.02.03_Ergebnisvalidierung

Testprotokolle: Google Code → Dokumente → 04_Tests → 04.03_Testprotokolle → 04.02.03_Ergebnisvalidierung

7.3 Testfallbeschreibung

In diesem Testfall wird die Methode *static boolean processExternalStreckenbefehl(void)* auf Korrektheit überprüft.Die Methode ist dafür zuständig, zu überprüfen, ob ein neuer Streckenbefehl vom SSC-Treiber vorhanden ist. Ist dies der Fall, wird der Streckenbefehl intern abgespeichert und der Speicher resetet, also auf {LEER,LEER,LEER,0} gesetzt.

Die Methode gibt TRUE zurück, falls der externe oder/und der interne Streckenbefehl neu ist/sind. Ist keiner der beiden Streckenbefehle neu, wird FALSE zurückgegeben.

7.4 Testskript

Es wird getestet, ob der Aufruf der Funktion *static boolean processExternalStreckenbefehl(void)* mit unterschiedlich übergebenen Variablen stets richtige Rückgabewerte liefert. Involvierte Variablen sind hierbei:

- *SSC_EV_streckenbefehl* (Streckenbefehl)
- *isBVNew* (boolean)

Die beiden oben aufgeführten Variablen müssen unterschiedlich miteinander kombiniert werden, um eine 100%-ige Codeabdeckung zu gewährleisten.

Dies wird mit folgendem Test-Skript realisiert:

siehe 'Google Code → 04_Test → 04.02_Testskripts → 04.02.03_Ergebnisvalidierung'

7.5 Testreferenz

Während des Testdurchlaufs werden die in Tabelle 3 dargestellten Rückgabewerte erwartet:

TP	Zustand	Erwartete Ausgabe
1	SSC_EV_streckenbefehl ={LEER,LEER,LEER,0}, isBVNew=X	• externerStreckenbefehl =SSC_EV_streckenbefehl • SSC_EV_streckenbefehl ={LEER,LEER,LEER,0} • isSSCNew=TRUE • Rückgabewert: TRUE
2	SSC_EV_streckenbefehl !={LEER,LEER,LEER,0}, isBVNew=FALSE	• Rückgabewert: FALSE
3	SSC_EV_streckenbefehl !={LEER,LEER,LEER,0}, isBVNew=TRUE	• Rückgabewert: TRUE

Tabelle 3: Erwartete Ausgabe für den Testfall 3

7.6 Test-Protokoll

Das Konsolen-Ergebnis wird in das Dokument 'Protokoll_Test_Ergebnisvalidierung' kopiert und diese Datei im Ordner 'Google Code → Dokumente → 04_Tests → 04.03_Testprotokolle → 04.03.03_Ergebnisvalidierung' abgelegt.

8 Testfall 4 „Prüfung des Streckenbefehlvergleichs“

8.1 Identifikation des Testobjektes

siehe Kapitel 3

8.2 Test-Identifikation

Testname: Test_Ergebnisvalidierung_StreckenbefehlVergleich

Verzeichnisse:

Testskripts: Google Code → Dokumente → 04_Tests → 04.02_Testskript → 04.02.03_Ergebnisvalidierung

Testprotokolle: Google Code → Dokumente → 04_Tests → 04.03_Testprotokolle → 04.02.03_Ergebnisvalidierung

8.3 Testfallbeschreibung

Gegenstand dieses Testfalls ist die Prüfung des korrekten Vergleichs des internen mit dem externen Streckenbefehl. Dies geschieht über die Methode *static boolean streckenbefehleEqual(Streckenbefehl *track1, Streckenbefehl *track2)*. *Streckenbefehl *track1* ist hierbei der interne Streckenbefehl und *Streckenbefehl *track2* der externe Streckenbefehl. Das Modul *void workEV(void)* wird bei Unstimmigkeiten der beiden Streckenbefehle, also wenn *static boolean streckenbefehleEqual(Streckenbefehl *track1, Streckenbefehl *track2) FALSE liefert,* verlassen. Sind beide Streckenbefehle gleich, wird TRUE zurückgegeben.

8.4 Testskript

Es wird getestet, ob der Aufruf der Funktion *static boolean streckenbefehleEqual(Streckenbefehl *track1, Streckenbefehl *track2)* mit unterschiedlich übergebenen Variablen stets richtige Rückgabewerte liefert. Involvierte Variablen sind hierbei:

- *isBVNew* (boolean)
- *isSSCNew* (boolean)
- *internerStreckenbefehl* (Streckenbefehl)
- *externerStreckenbefehl* (Streckenbefehl)
- *streckenbefehleUngleich* (Byte)

Die oben aufgeführten Variablen müssen unterschiedlich miteinander kombiniert werden, um eine 100%-ige Codeabdeckung zu gewährleisten.

Dies wird mit folgendem Test-Skript realisiert:

siehe 'Google Code → 04_Test → 04.02_Testskripts → 04.02.03_Ergebnisvalidierung'

8.5 Testreferenz

Während des Testdurchlaufs werden die in Tabelle 4 dargestellten Rückgabewerte erwartet.

TP	Zustand	Erwartete Ausgabe
1	isBVNew=FALSE also internerStreckenbefehl ={LEER,LEER,LEER,0} isSSCNew=X streckenbefehleUngleich=X	• Rückgabewert: FALSE
2	isBVNew=X isSSCNew=FALSE also externerStreckenbefehl ={LEER,LEER,LEER,0} streckenbefehleUngleich=X	• Rückgabewert: FALSE
3	isBVNew=TRUE isSSCNew=TRUE internerStreckenbefehl =externerStreckenbefehl !={LEER,LEER,LEER,0} streckenbefehleUngleich=X	• streckenbefehleUngleich=0? • Rückgabewert: TRUE
4	isBVNew=TRUE internerStreckenbefehl !={LEER,LEER,LEER,0} isSSCNew=TRUE externerStreckenbefehl !={LEER,LEER,LEER,0} internerStreckenbefehl !=externerStreckenbefehl streckenbefehleUngleich =0...2	• Nachricht an AS: (1,2,0...2,0,0) • streckenbefehleUngleich von 0 auf 1, 1 auf 2, oder 2 auf 3 • Rückgabewert: FALSE

5	isBVNew=TRUE internerStreckenbefehl !={LEER,LEER,LEER,0} isSSCNew=TRUE externerStreckenbefehl !={LEER,LEER,LEER,0} internerStreckenbefehl !=externerStreckenbefehl streckenbefehleUngleich =3...4	• Nachricht an AS: (1,3,4...5,0,0) • Not-Aus-Signal auslösen • Rückgabewert: FALSE • der Zustand streckenbefehleUngleich=4 sollte nicht erreicht werden können, da die Variable vorher zurückgesetzt wird

Tabelle 4: Erwartete Ausgabe für den Testfall 4

8.6 Test-Protokoll

Das Konsolen-Ergebnis wird in das Dokument 'Protokoll_Test_Ergebnisvalidierung' kopiert und diese Datei im Ordner 'Google Code → Dokumente → 04_Tests → 04.03_Testprotokolle → 04.03.03_Ergebnisvalidierung' abgelegt.

9 Testfall 5 „Prüfen des Sendevorgangs an den RS232-Treiber“

9.1 Identifikation des Testobjektes

siehe Kapitel 3

9.2 Test-Identifikation

Testname: Test_Ergebnisvalidierung_StreckenbefehlVergleich

Verzeichnisse:

Testskripts: Google Code → Dokumente → 04_Tests → 04.02_Testskript → 04.02.03_Ergebnisvalidierung

Testprotokolle: Google Code → Dokumente → 04_Tests → 04.03_Testprotokolle → 04.02.03_Ergebnisvalidierung

9.3 Testfallbeschreibung

In diesem Testfall soll die Korrektheit des Sendens an den RS232-Treiber kontrolliert werden.

Bei dem zum Testfall zugehörigen Quellcode wird zuerst geprüft, ob ein alter Streckenbefehl nicht gesendet werden konnte und ob der Shared-Memory zum RS232-Treiber zurückgesetzt ist. Nach diesen Kriterien wird entweder der Streckenbefehl an den RS232-Treiber gesendet, eine Warnung an das AS geschickt oder ein Fehler erzeugt, der das Auslösen eines Not-Aus-Signal nach sich zieht.

9.4 Testskript

Wie unter 9.3 kurz beschrieben, wird zuerst geprüft, ob *oldRS232* resetet ist. Ist dies nicht der Fall wird eine Fehlermeldung an das AS gesendet und der Not-Aus-Treiber angesprochen. Anderenfalls kommt die Abfrage, ob *EV_RS232_streckenbefehl* ebenfalls resetet ist. Ist die gerade genannte Bedingung erfüllt, wird eine Information an das AS geschickt und der Streckenbefehl an den RS232-Treiber übertragen. Ist die Bedingung nicht erfüllt, wird eine Warnung erzeugt und der *counterRS232* von '0' auf '1' erhöht.

Involvierte Variablen sind also:

- *oldRS232* (Streckenbefehl)
- *EV_RS232_streckenbefehl* (Streckenbefehl)

Wird *oldRS232* resetet, trägt also die Werte {LEER,LEER,LEER,0}, wird in der Methode *static boolean checkForCommunicationErrors(void)* ebenfalls der *counterRS232* auf '0' gesetzt.

Die oben aufgeführten Variablen müssen unterschiedlich miteinander kombiniert werden, um eine 100%-ige Codeabdeckung zu gewährleisten.

Dies wird mit folgendem Test-Skript realisiert:

siehe 'Google Code → 04_Test → 04.02_Testskripts → 04.02.03_Ergebnisvalidierung'

9.5 Testreferenz

Während des Testdurchlaufs werden die in Tabelle 5 dargestellten Rückgabewerte erwartet.

TP	Zustand	Erwartete Ausgabe
1	oldRS232 !={LEER,LEER,LEER,0} EV_RS232_streckenbefehl ={X,X,X,X}	• Nachricht an AS: (3,3,internerStreckenbefehl.Lok,internerStreckenbefehl.Weiche,internerStreckenbefehl.Entkoppler) • Not-Aus-Signal auslösen
2	oldRS232 ={LEER,LEER,LEER,0} EV_RS232_streckenbefehl !={X,X,X,X}	• oldRS232=internerStreckenbefehl? • Nachricht an AS: (3,2,0,0,0) • counterRS232=1? • return
3	oldRS232 ={LEER,LEER,LEER,0} EV_RS232_streckenbefehl ={X,X,X,X}	• Nachricht an AS: (3,1,internerStreckenbefehl.Lok,internerStreckenbefehl.Weiche,internerStreckenbefehl.Entkoppler) • EV_RS232_streckenbefehl =internerStreckenbefehl?

Tabelle 5: Erwartete Ausgabe für den Testfall 5

9.6 Test-Protokoll

Das Konsolen-Ergebnis wird in das Dokument 'Protokoll_Test_Ergebnisvalidierung' kopiert und diese Datei im Ordner 'Google Code → Dokumente → 04_Tests → 04.03_Testprotokolle → 04.03.03_Ergebnisvalidierung' abgelegt.

10 Testfall 6 „Prüfen des Sendevorgangs an den RS232-Treiber“

10.1 Identifikation des Testobjektes

siehe Kapitel 3

10.2 Test-Identifikation

Testname: Test_ErgebnisvalidierungStreckenbefehlVergleich

Verzeichnisse:

Testskripts: Google Code → Dokumente → 04_Tests → 04.02_Testskript → 04.02.03_Ergebnisvalidierung

Testprotokolle: Google Code → Dokumente → 04_Tests → 04.03_Testprotokolle → 04.02.03_Ergebnisvalidierung

10.3 Testfallbeschreibung

In dem letzten Testfall werden die internen Variablen zurückgesetzt.

10.4 Testskript

Es müssen die unten aufgezählten Variablen auf ihre korrekte Wertebelegung geprüft werden.

- *counterSSC* = 0 (Byte)
- *counterRS232* = 0 (Byte)
- *streckenbefehleUngleich* = 0 (Byte)
- *internerStreckenbefehl* = {LEER,LEER,LEER,0} (Streckenbefehl)
- *externerStreckenbefehl* = {LEER,LEER,LEER,0} (Streckenbefehl)
- *EV_nachricht* = {0,0,0,0,0} (Byte)
- *oldSSC* = {LEER,LEER,LEER,0} (Streckenbefehl)
- *oldRS232* = {LEER,LEER,LEER,0} (Streckenbefehl)

Dies wird mit folgendem Test-Skript realisiert:

siehe 'Google Code → 04_Test → 04.02_Testskripts → 04.02.03_Ergebnisvalidierung'

10.5 Testreferenz

Während des Testdurchlaufs werden die in Tabelle 6 dargestellten Rückgabewerte erwartet. Die in Spalte 2 aufgelisteten Zustände werden vor der Methode *void initEV(void)* definiert.

Nach Durchlauf von *void initEV(void)* müssen die Variablen den in Spalte 3 dargestellten Wert angenommen haben.

TP	Zustand	Erwartete Ausgabe
1	counterSSC=0...7 (die Zahlen von '0' bis '7' werden alle durchprobiert werden)	• *counterSSC*=0
2	counterRS232=0...7 (die Zahlen von '0' bis '7' werden alle durchprobiert werden)	• *counterRS232*=0
3	streckenbefehleUngleich =0...7 (die Zahlen von '0' bis '7' werden alle durchprobiert werden)	• *streckenbefehleUngleich*=0
4	internerStreckenbefehl ={0:5:255,0:5:255,0:5:255, 0:5:255} (die Zahlen von '0' bis '255' werden in 5-er-Schritten durchprobiert)	• internerStreckenbefehl={255,255,255,0}
5	externerStreckenbefehl ={0:5:255,0:5:255,0:5:255,0:5: 255} (die Zahlen von '0' bis '255' werden in 5-er-Schritten durchprobiert)	• externerStreckenbefehl={255,255,255,0}
6	EV_nachricht={0...7,0...7,0...7 ,0...7,0...7} (die Zahlen von '0' bis '7' werden alle durchprobiert werden)	• EV_nachricht={0,0,0,0,0}
7	oldSSC ={0:5:255,0:5:255,0:5:255,0:5: 255} (die Zahlen von '0' bis '255' werden in 5-er-Schritten durchprobiert)	• oldSSC={255,255,255,0}

8	oldRS232 ={0:5:255,0:5:255,0:5:255,0:5:255} (die Zahlen von '0' bis '255' werden in 5-er-Schritten durchprobiert)	• oldRS232={255,255,255,0}

Tabelle 6: Erwartete Ausgabe für den Testfall 6

Der Ausdruck X:Y:Z bedeutet, dass man sich schrittweise dem Wert Z nähret durch das Erhöhen von X um Y.

10.6 Test-Protokoll

Das Konsolen-Ergebnis wird in das Dokument 'Protokoll_Test_Ergebnisvalidierung' kopiert und diese Datei im Ordner 'Google Code → Dokumente → 04_Tests → 04.03_Testprotokolle → 04.03.03_Ergebnisvalidierung' abgelegt.

11 Auswertung

wird nach Testdurchführung erstellt