Modul-Design Ergebnisvalidierung

Für das studentische Projekt *Sichere Eisenbahnsteuerung*

Datum	26.05.2010
Quelle	Dokumente\02_Design\02.02_Moduldesign
Autoren	F. Geber
Version	2.0
Status	freigegeben

1Historie

Version	Datum	Autor	Bemerkung
0.1	24.11.09	Philip Weber	Initial.
0.2	07.12.09	Philip Weber	Inhalt.
0.3	08.12.09	Philip Weber	Kapitel '6 Dynamisches Verhalten'
0.4	21.12.09	Philip Weber	- Funktionen entfernt. (read_error, ...) - Kapitel '6 Dynamisches Verhalten' erweitert.
0.5	22.12.09	Philip Weber	Funktionen workEV() und initEV() wurden hinzugefügt.
0.6	22.12.09	Philip Weber	Typ *byte[3]* durch den Typ *Streckenbefehl* ersetzt.
0.7	22.12.09	Philip Weber	*SSC_EV_failure* und *RS232_EV_failure* in den Typ *Streckenbefehl* integriert
0.8	31.12.09	Philip Weber	- Referenz auf 'RS232 Treiber – Modul-Design' - Kapitel 8: 'compareStreckenbefehle' entfernt - Kapitel 8: Variablen angepasst - Kapitel 12 komplett neu erstellt
0.9	13.01.10	Philip Weber	- Kapitel 15: *externerStreckenbefehl* durch *internerStreckenbefehl* ersetzt - Punkte 2, 3, 5, 6, 7, 9, 10 aus dem Review eingearbeitet
1.0	02.02.10	Philip Weber	- Kapitel 'Error: Reference source not found Error: Reference source not found' eingefügt.
1.1	04.02.10	Philip Weber	- Kapitel 8 und 10: Variablen *isBVNew, isSSCNew* und *EV_nachricht[6]* eingefügt - Kapitel 9: Schnittstelle zum „Software Watchdog“-Modul entfernt - Kapitel 14, 15 und 15: Verarbeitung von *isBVNew* und *isSSCNew* eingefügt. - Der Korrektheit wegen wurde *0xFFh* durch {*0xFFh, 0xFFh, 0xFFh, 0x00h*} ersetzt. - Abbildungen 4, 5 und 6 eingefügt. - Abbildungen 2, 3 angepasst.
1.2	07.02.10	Philip Weber	- Kapitel 14 und 17 verbessert. - Abbildung 3 korrigiert und verbessert.
2.0	26.05.10	F. Geber	Layout-Anpassungen, Umwandlung der Kreuzung in eine Vereinigung in Abbildung 5 (vor „Streckenbefehle

			vergleichen“)

2Inhaltsverzeichnis

Inhaltsverzeichnis

3Einleitung

Das Modul „Ergebnisvalidierung“ dient der Überprüfung der von der Anwendung erzeugten Streckenbefehle auf Korrektheit. Hierzu werden diese über die SSC-Schnittstelle zwischen den Mikrocontrollern ausgetauscht und verglichen. Unterscheiden sich diese voneinander, wird ein Not-Aus-Signal erzeugt und eine Statusmeldung mit den abweichenden Werten an das Auditing-System zur Protokollierung gesendet.

4Referenzierte Dokumente

- Dokumente\02_Design\02.01_Subsystemdesign\Softwaredesign
- Dokumente\02_Design\02.02_Moduldesign\Modul-Design_Betriebsmittelverwaltung
- Dokumente\02_Design\02.02_Moduldesign\Modul-Design_RS232 Treiber
- Dokumente\02_Design\02.02_Moduldesign\Modul-Design_SSC Treiber
- Dokumente\02_Design\02.02_Moduldesign\Modul-Design_Befehlsvalidierung

5Architektur

5.1Funktionen

Tabelle 1: void initEV(void)

Name:	**initEV**
Beschreibung:	Dient zum Initialisieren des Moduls.
Vorbedingung:	keine
Parameter:	keine
Rückgabe:	keine
Nachbedingung:	Es wurden die intern verwendeten Variablen zurückgesetzt.
Fehlerfall:	keine

Tabelle 2: void workEV(void)

Name:	**workEV**
Beschreibung:	Diese Methode wird von der Befehlsvalidierung bei jedem Durchlauf aufgerufen.
Vorbedingung:	Das Modul wurde initialisiert.
Parameter:	keine
Rückgabe:	keine
Nachbedingung:	keine
Fehlerfall:	emergency_off() wird aufgerufen und der Fehler wird protokolliert.

5.2Static Variablen

Nur vom Modul Ergebnisvalidierung verwendet:

- **byte counterSSC**

Zähler, falls der Shared Memory zum SSC-Treiber nicht {*0xFFh, 0xFFh, 0xFFh, 0x00h*} ist.

- **byte counterRS232**

Zähler, falls der Shared Memory zum RS232-Treiber nicht {*0xFFh, 0xFFh, 0xFFh, 0x00h*} ist.

- **byte streckenbefehleUngleich**

Zähler, falls die Streckenbefehle einander nicht entsprechen.

•**byte EV_nachricht[6]**

Byte-Array zur Übermittlung von Fehlern und Nachrichten an das Auditing-System

•**boolean isBVNew**

Zeigt an, ob ein neuer Streckenbefehl von der Befehlsvalidierung vorhanden ist

•**boolean isSSCNew**

Zeigt an, ob ein neuer Streckenbefehl vom SSC-Treiber vorhanden ist

•**Streckenbefehl internerStreckenbefehl**

Dient zum Zwischenspeichern des Streckenbefehls des eigenen Mikrocontrollers.

•**Streckenbefehl externerStreckenbefehl**

Dient zum Zwischenspeichern des Streckenbefehls des anderen Mikrocontrollers.

5.3Globale Variablen

Vom Modul Befehlsvalidierung und Ergebnisvalidierung verwendet:

•**Streckenbefehl BV_EV_streckenbefehl**

Streckenbefehl von der Befehlsvalidierung.

Vom Modul SSC-Treiber und Ergebnisvalidierung verwendet:

•**Streckenbefehl EV_SSC_streckenbefehl**

Streckenbefehl zum SSC-Treiber.

•**Streckenbefehl SSC_EV_streckenbefehl**

Streckenbefehl vom SSC-Treiber.

Vom Modul RS232-Treiber und Ergebnisvalidierung verwendet:

•**Streckenbefehl EV_RS232_streckenbefehl**

Streckenbefehl zum RS232-Treiber.

5.4Abhängigkeiten von anderen Modulen

–Befehlsvalidierung

Sendet die in der Leitzentrale erzeugten Streckenbefehle an die Ergebnisvalidierung.

Zur Übergabe dient die globale Variable *BV_EV_streckenbefehl*

–SSC-Treiber

1. Austausch von Streckenbefehlen zum SSC-Treiber: *EV_SSC_streckenbefehl*

2. Austausch von Streckenbefehlen vom SSC-Treiber: *SSC_EV_streckenbefehl*

2. Bei der Kommunikation entstandene Fehler: *SSC_EV_streckenbefehl.Fehler*

–RS232-Treiber

1. Austausch von Streckenbefehlen zum RS232-Treiber: *EV_RS232_streckenbefehl*

2. Bei der Kommunikation entstandene Fehler: *RS232_EV_streckenbefehl.Fehler*

Die Kommunikation zwischen den genannten Modulen und der Ergebnisvalidierung erfolgt über einen Shared Memory.

–Auditing-System

Dem Auditing-System werden die einzelnen Arbeitsschritte zur Protokollierung mitgeteilt.

5.5Schnittstellenbeschreibung

Es gibt eine Schnittstelle, die von außen aufgerufen wird.

–void workEV()

Diese Methode wird aus dem Hauptprogramm (Betriebsmittelverwaltung.c) aufgerufen, um die Validierung der Streckenbefehle einzuleiten.

–void initEV()

Diese Methode dient zum Initialisieren des Moduls. Es werden folgende Variablen initialisiert:

–`counterSSC` = *0*

–`counterRS232` = *0*

–`internerStreckenbefehl` = {*0xFFh, 0xFFh, 0xFFh, 0x00h*}

–`isBVNew` = *FALSE*

–`isSSCNew` = *FALSE*

–`externerStreckenbefehl` = {*0xFFh, 0xFFh, 0xFFh, 0x00h*}

–`streckenbefehleUngleich` = *0*

–`EV_nachricht` = *0* (Alle sechs Bytes)

5.6Nachrichtenkodierung

Die Nachrichten, die an das Auditing-System gesendet werden, müssen kodiert werden, da jede Nachricht nur sechs Byte groß sein darf. Im Folgenden werden die einzeln kodierten Nachrichten in einer Tabelle zur besseren Übersicht zusammengefasst. Das 'X' steht dabei für einen beliebigen Wert (z.B. den Wert einer Zählvariablen).

Tabelle 3: Nachrichtenkodierung an das Auditing-System

Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Bedeutung
1	2	X	0	0	0	**Warnung:** Anzahl aufeinander folgender unterschiedlicher Streckenbefehle
1	3	4	0	0	0	**Fehler:** Streckenbefehle ungleich
2	2	X	0	0	0	**Warnung:** Anzahl vergeblicher Versuche den Streckenbefehl an den SSC-Treiber zu senden
2	3	4	0	0	0	**Fehler:** Streckenbefehl konnte nicht an den SSC-Treiber gesendet werden
3	2	X	0	0	0	**Warnung:** Anzahl vergeblicher Versuche den Streckenbefehl an den RS232-Treiber zu senden
3	3	4	0	0	0	**Fehler:** Streckenbefehl konnte nicht an den RS232-Treiber gesendet werden
4	3	0	X	0	0	**Fehler:** Fehlermeldung vom SSC-Treiber kommend
5	3	0	X	0	0	**Fehler:** Fehlermeldung vom RS232-Treiber kommend
6	1	X	X	X	0	**Info:** Streckenbefehl der an den SSC-Treiber gesendet wurde
7	1	X	X	X	0	**Info:** Streckenbefehl der an den RS232-Treiber gesendet wurde.

6Dynamisches Verhalten

6.1Übersicht

Beim Aufruf der Funktion `workEV()` durch das Hauptprogramm wird die Ergebnisvalidierung gestartet. Bei jedem Moduldurchlauf wird als erstes überprüft, ob bei der Verarbeitung der Streckenbefehle bei einem der Treiber (RS232 oder SSC) ein Fehler aufgetreten ist. Danach wird überprüft, ob alle Streckenbefehle beim letzten Modulaufruf gesendet werden konnten. Im Anschluss daran wird der, von der Befehlsvalidierung kommende, interne Streckenbefehl überprüft und anschließend an den anderen Mikrocontroller gesendet. Daraufhin wird der vom zweiten Mikrocontroller stammende Streckenbefehl ausgelesen. Diese beiden werden miteinander verglichen und falls diese gleich sind, wird zum Schluss der interne Streckenbefehl an den RS232-Treiber gesendet, damit dieser ihn an die Strecke senden kann.

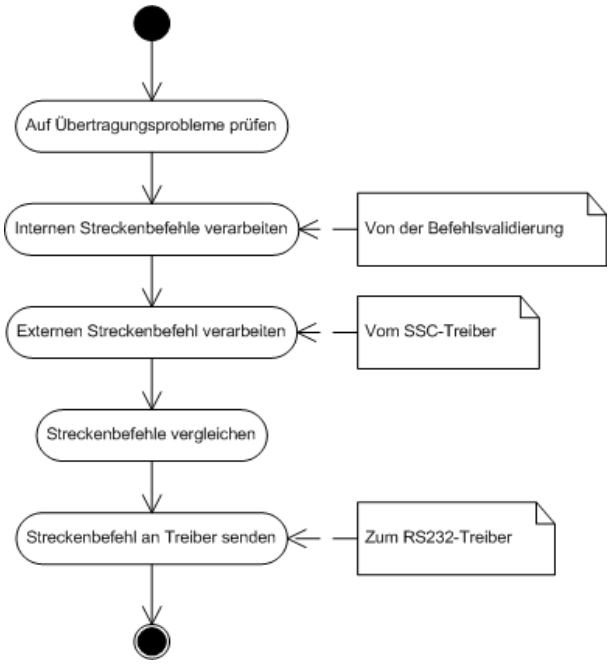

Abbildung 1: Ablauf der Ergebnisvalidierung

6.2Auf Übertragungsprobleme prüfen

Beim Aufruf der Ergebnisvalidierung wird zuerst überprüft ob Probleme bei der Übertragung aufgetreten sind. Sind bei der Verarbeitung der Streckenbefehle innerhalb der Module *RS232-Treiber* oder *SSC-Treiber* Fehler aufgetreten, schreiben diese einen Fehlercode in den entsprechenden Bereich des Shared Memory:

- `RS232_EV_Streckenbefehl.Fehler` bzw.
- `SSC_EV_Streckenbefehl.Fehler`

Weist eine der beiden Fehlervariablen einen anderen Wert als *0x00h* (Kein Fehler, siehe 'RS232 Treiber – Modul-Design' Kapitel 6.2 *Erzeugte Fehler*) auf, ist davon auszugehen, dass

ein Fehler bei der Kommunikation mit dem anderen Mikrocontroller (SSC) oder mit der MultiMAUS (RS232) aufgetreten ist. Der Fehler wird daraufhin an das Auditing-System übermittelt und ein Not-Aus eingeleitet.

Erzeugt die Überprüfung der Fehlercodes keinen Not-Aus, wird überprüft ob die Ergebnisvalidierung bei vorherigen Durchläufen die Streckenbefehle an den *RS232*- und *SSC-Treiber* senden konnte.

Hierzu wird zunächst überprüft ob die Zählvariablen `counterRS232` einen Wert größer *'0'* besitzt.

•Ist er gleich *'0'*, wird mit der nächsten Überprüfung fortgefahren.

•Ist er größer *'0'*, wird überprüft, ob der entsprechende Shared Memory Bereich (`EV_RS232_streckenbefehl`) zurückgesetzt (== {*0xFFh, 0xFFh, 0xFFh, 0x00h*}) wurde.

◦Falls ja, wird die Variable `internerStreckenbefehl` in den Shared Memory geschrieben und die Variable `counterRS232` auf *'0'* zurückgesetzt.

◦Falls `EV_RS232_streckenbefehl` ungleich {*0xFFh, 0xFFh, 0xFFh, 0x00h*} und `counterRS232` größer gleich *'3'* ist wird ein Not-Aus eingeleitet und der Fall dem Auditing-System (`sendNachricht(3,3,4,00)`, siehe Tabelle 3) mitgeteilt.

◦Falls `EV_RS232_streckenbefehl` ungleich {*0xFFh, 0xFFh, 0xFFh, 0x00h*} und `counterRS232` kleiner *'3'* ist, wird die Zählvariable `counterRS232` inkrementiert, der Fall dem Auditing-System (`sendNachricht(3,2,X,00)`, siehe Tabelle 3) mitgeteilt und mit der nächsten Überprüfung (`counterSSC`) fortgefahren.

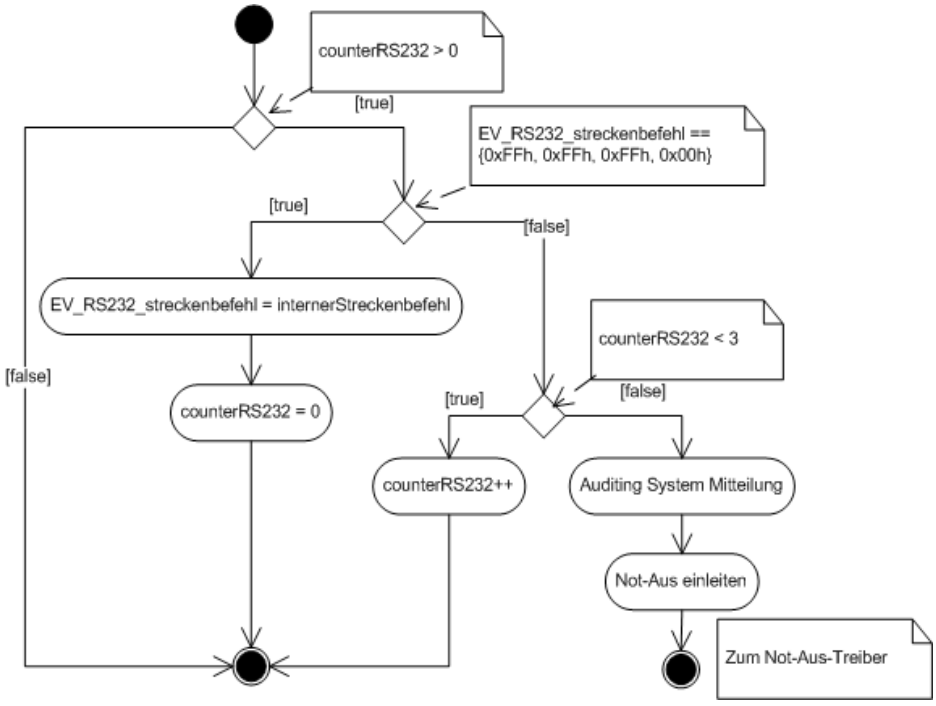

Abbildung 2: Überprüfung der Zählvariablen `counterRS232`

Wenn bei der Überprüfung der Zählvariable `counterRS232` kein Not-Aus eingeleitet wurde, wird die Überprüfung (siehe Abbildung 2) für den *SSC-Treiber* wiederholt. Hierfür müssen in Abbildung 2 `EV_RS232_streckenbefehl` durch `EV_SSC_streckenbefehl` und `counterRS232` durch `counterSSC` ersetzt werden.

6.3Internen Streckenbefehl verarbeiten

Es wird überprüft, ob sich ein neuer Streckenbefehl (!= {*0xFFh, 0xFFh, 0xFFh, 0x00h*}) im Shared Memory zur Befehlsvalidierung `BV_EV_streckenbefehl` befindet.

- Ist dies nicht der Fall, wird mit dem nächsten Schritt („Externen Streckenbefehl verarbeiten") fortgefahren.
- Wenn ja, wird `BV_EV_streckenbefehl` zur späteren Verwendung in die Variable `internerStreckenbefehl` geschrieben und anschließend zurückgesetzt (= {*0xFFh, 0xFFh, 0xFFh, 0x00h*}). Zusätzlich wird die Variable `isBVNew` auf *TRUE* gesetzt.

Falls das Ergebnis der ersten Überprüfung positiv war, wird überprüft ob in einem vorherigen Durchlauf ein Streckenbefehl nicht gesendet werden konnte (`isStreckenbefehlResetted(&oldSSC)` bzw. `counterSSC == 0`).

- Falls noch ein zu sendender Streckenbefehl aussteht wird eine Nachricht an das Auditing-System gesendet, ein Not-Aus eingeleitet und das Modul frühzeitig verlassen.
- Falls nicht, wird versucht den Streckenbefehl an den SSC-Treiber zu übergeben. Hierfür wird überprüft, ob der Shared Memory zum SSC-Treiber `EV_SSC_streckenbefehl` zurückgesetzt, also {*0xFFh, 0xFFh, 0xFFh, 0x00h*} ist.
- Trifft dies zu, wird die Variable `internerStreckenbefehl` in diesen geschrieben.
- Falls nicht, wird die Zählvariable `counterSSC` auf *'1'* gesetzt.

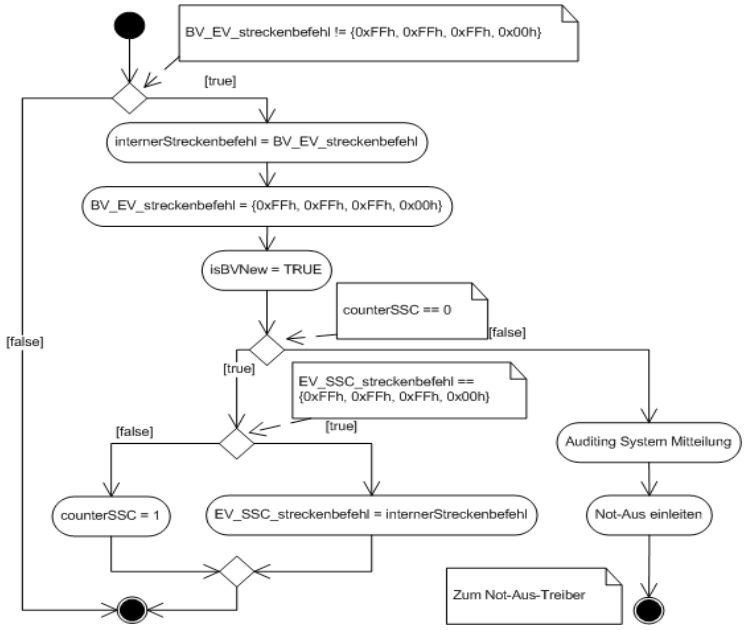

Abbildung 3: Internen Streckenbefehl verarbeiten

6.4Externen Streckenbefehl verarbeiten

In diesem Arbeitsschritt wird überprüft, ob ein neuer Streckenbefehl (!= {*0xFFh, 0xFFh, 0xFFh, 0x00h*}) vom SSC-Treiber in den Shared Memory zwischen diesem und der Ergebnisvalidierung (`SSC_EV_streckenbefehl`) geschrieben wurde.

•Falls ja, wird dieser in die Variable `externerStreckenbefehl` geschrieben und der Shared Memory auf den Wert {*0xFFh, 0xFFh, 0xFFh, 0x00h*} zurückgesetzt. Dies wird durchgeführt, um dem SSC-Treiber zu signalisieren, dass der Streckenbefehl ausgelesen wurde und ein neuer in den Shared Memory geschrieben werden kann. Zusätzlich wird die Variable `isSSCNew` auf *TRUE* gesetzt. Danach wird mit dem nächsten Schritt („Streckenbefehle vergleichen“) fortgefahren.

•Falls sich kein neuer Streckenbefehl im Shared Memory befindet, hängt das weitere Vorgehen davon ab, ob die Variable `internerStreckenbefehl` einen Streckenbefehl ungleich {*0xFFh, 0xFFh, 0xFFh, 0x00h*} besitzt. Der Performance wegen, wird hier die boolesche Variable `isBVNew` und nicht `internerStreckenbefehl` überprüft.

◦ Falls ja, wird mit dem nächsten Schritt („Streckenbefehle vergleichen“) fortgefahren.

◦ Falls nicht, wird das Modul frühzeitig verlassen.

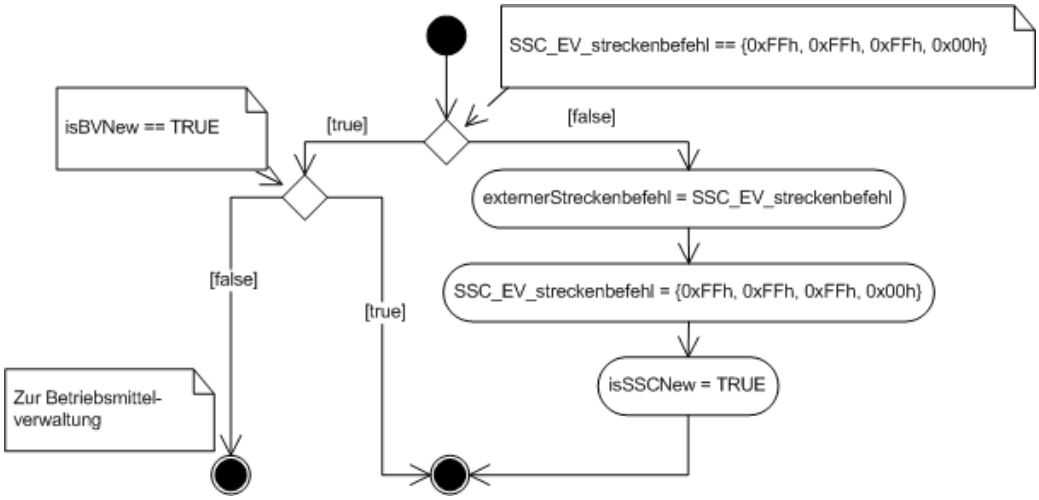

Abbildung 4: Externen Streckenbefehl verarbeiten

6.5Streckenbefehle vergleichen

Es wird überprüft, ob neue Streckenbefehle in die Variablen `internerStreckenbefehl` und `externerStreckenbefehl` geschrieben wurden. Hierfür werden die Variablen `isBVNew` und `isSSCNew` überprüft. *TRUE* bedeutet, dass ein neuer Streckenbefehl vorhanden ist. Falls beide neu sind, wird die Zählvariable `streckenbefehleUngleich` auf *'0'* zurückgesetzt. Im Anschluss daran wird der eigentliche Streckenbefehlvergleich durchgeführt (siehe Kapitel 16).

Ist wenigstens einer der beiden Streckenbefehle nicht neu wird überprüft, ob sich Streckenbefehle (!= {*0xFFh, 0xFFh, 0xFFh, 0x00h*}), egal ob neu oder alt, in den Variablen `internerStreckenbefehl` und `externerStreckenbefehl` befinden. Falls ja, werden diese miteinander verglichen (Kapitel 16).

Sind weder neue noch alte Streckenbefehle vorhanden wird das Modul frühzeitig verlassen.

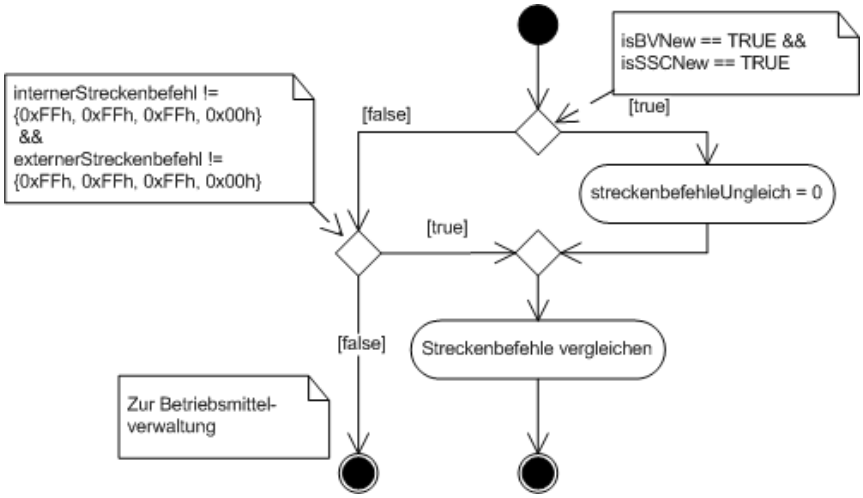

Abbildung 5: Grundsätzlicher Ablauf des Vergleichs

Vergleich durchführen:

Die beiden Streckenbefehle werden verglichen (siehe Abbildung 6).

•Entsprechen diese einander wird die Variable `streckenbefehleUngleich` auf den Wert *'0'* gesetzt und mit dem nächsten Schritt (Kapitel 17) fortgefahren.

•Sind sie ungleich wird die Zählvariable `streckenbefehleUngleich` inkrementiert, eine Warnung an das Auditing System gesendet und das Modul frühzeitig verlassen.

Dies wird solange wiederholt (bei jedem Modulaufruf ein Mal) bis die Variable `streckenbefehleUngleich` den Wert *'3'* erreicht hat. Danach wird dies dem Auditing System mitgeteilt und ein Not-Aus eingeleitet.

Durch diesen Mechanismus wird sichergestellt, dass zwei ungleiche Streckenbefehle nicht automatisch zu einem Not-Aus führen, da der Fall eintreten könnte, dass die beiden Mikrocontroller die Sensoren nicht zum selben Zeitpunkt auslesen, wodurch Unterschiede im Gleisbild entstehen können, die aber mit dem nächsten Auslesen der Sensorwerte wieder ausgeglichen sein sollten.

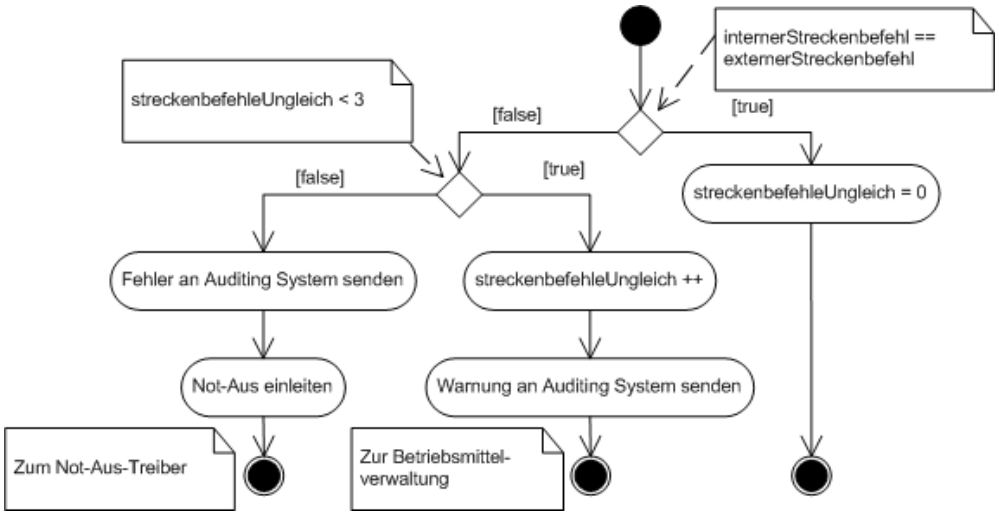

Abbildung 6: Streckenbefehle vergleichen

6.6Streckenbefehl an Treiber senden

Zuerst wird überprüft ob ein Streckenbefehl bei einem vorherigen Durchlauf nicht gesendet werden konnte (`counterRS232` > *0*).

•Falls noch ein zu sendender Streckenbefehl aussteht wird ein Nachricht an das Auditing System gesendet und ein Not-Aus eingeleitet und das Modul frühzeitig verlassen.

•Falls nicht wird überprüft, ob der Shared Memory zum RS232-Treiber `EV_RS232_streckenbefehl` durch das Treiber-Modul zurückgesetzt wurde (== {*0xFFh, 0xFFh, 0xFFh, 0x00h*}). Dies bedeutet, dass der Treiber zumindest mit der Verarbeitung des letzten Streckenbefehls begonnen hat und einen neuen Streckenbefehl entgegennehmen kann.

•Ist der Shared Memory szurückgesetzt, wird die Variable `internerStreckenbefehl` in die Shared Memory Variable `EV_RS232_streckenbefehl` geschrieben. Im Anschluss daran werden die Variablen `streckenbefehleUngleich`, `counterSSC`, `counterRS232`, `internerStreckenbefehl` und `externerStreckenbefehl` zurückgesetzt.

•Wurde der Shared Memory nicht zurückgesetzt wird eineNachricht an das Auditing System gesendet und die Zählvariable `counterRS232` auf *'1'* gesetzt.