Diplomarbeit

Erstellen einer Lerneinheit über den Interbus-S für die Virtuelle Hochschule Karlsruhe (einschl. einer Java-Visualisierung)

Bearbeiter:	Manuel Müller
Kurs:	TEL97AT
Betreuer:	Dipl.-Ing. Stefan Braun
Betreuer BA:	Prof. Dr. G. Richter

Inhalt

1 Einleitung

1.1 Einleitung

Diese Diplomarbeit befasst sich mit einer Lerneinheit über den Interbus-S. Diese Einheit wird im Rahmen der Virtuellen Hochschule Karlsruhe (Vikar) eingesetzt werden.

Vikar ist ein Projekt, zu dem sich alle Karlsruher Hochschulen zusammengeschlossen haben, um den Studierenden die Möglichkeit zu geben, hochschulübergreifende Lerninhalte über das Internet multimedial zu erfahren. Weiterhin soll die Möglichkeit gegeben werden Themen, die während der Vorlesungen nur angeschnitten werden können, selbstständig zu vertiefen.
Durch dieses Gemeinschaftsprojekt lässt sich das vorhandene Wissenspotential bündeln und zentral zur Verfügung stellen.

Die Arbeit hat sich in zwei wesentliche Teile gegliedert. Zum einen wurden mehrere Intetnet-Seiten mit Text- und Grafikinhalten erstellt, zum anderen wurden die Vorgänge beim Interbus-S auch visualisiert. Die Visualisierung ist als JAVA-Applet realisiert worden.

Auf den folgenden Seiten wird nun detailliert die Vorgehensweise beschreiben, mit deren Hilfe diese Lerneinheit fertiggestellt wurde.

An dieser Stelle möchte ich noch Herrn Professor Doktor G. Richter danken, der mich bei der Erstellung dieser Diplomarbeit betreut und unterstützt hat.
Ein weiterer Dank gilt dem Interbus-Club Deutschland, der mir unbürokratisch Informationen und Grafiken für diese Diplomarbeit bereitgestellt hat.

1.2 Der Interbus-S

Dieser Abschnitt soll einen kurzen Überblick über den Interbus geben. Genauere Informationen finden sich in der Lerneinheit zu dieser Diplomarbeit.

1.2.1 Eine kurze Einführung

- Der Interbus ist ein sehr effizienter Automatisierungsbus, der auf der Feldebene zu Einsatz kommt.
- Der Aufbau des Interbus ist ringförmig.
- Die Datenübertragung erfolgt seriell mit Hilfe eines Summenrahmen-Verfahrens.
- Durch den Aufbau und das Übertragungsverfahren entfallen alle Formen der Adressierungsleitungen.
- Der Interbus konfiguriert sich nach dem Einschalten automatisch durch einen Initialisierungs-Zyklus.
- Die Datensicherung erfolgt über ein CRC-Prüfsummen-Verfahren.

2 Pflichtenheft

Die folgenden Punkte sollen im Rahmen dieser Diplomarbeit bearbeitet und realisiert werden:

- Es soll eine gut strukturierte Lerneinheit für das Projekt der Virtuellen Hochschule Karlsruhe erstellt werden.
- Die Struktur soll sich an der der Herren Koyne und Gundert orientieren, die bereits eine Lerneinheit für den Profibus fertig gestellt haben.
- Diese Lerneinheit ist nahtlos in die vorgegebene Lernumgebung (Companion) einzufügen.
- Aus der Lerneinheit sollen sowohl die Entwicklung, die Struktur als auch der logische Aufbau des Interbus-S ersichtlich sein.
- Das Verhalten der einzelnen Busteilnehmer im Normal- und im Fehlerfall soll beschrieben und visuell dargestellt werden.
  Dazu gehören:
  - die Zugriffsstrategie
  - die Datenübertragung
  - die Möglichkeiten der Fehlererkennung und -lokalisierung
- Die visuelle Darstellung soll mit Hilfe der Programmiersprache JAVA erreicht werden, da dies die flexibelste Lösung bietet.

3 Das Lernsystem Companion

3.1 Allgemeines

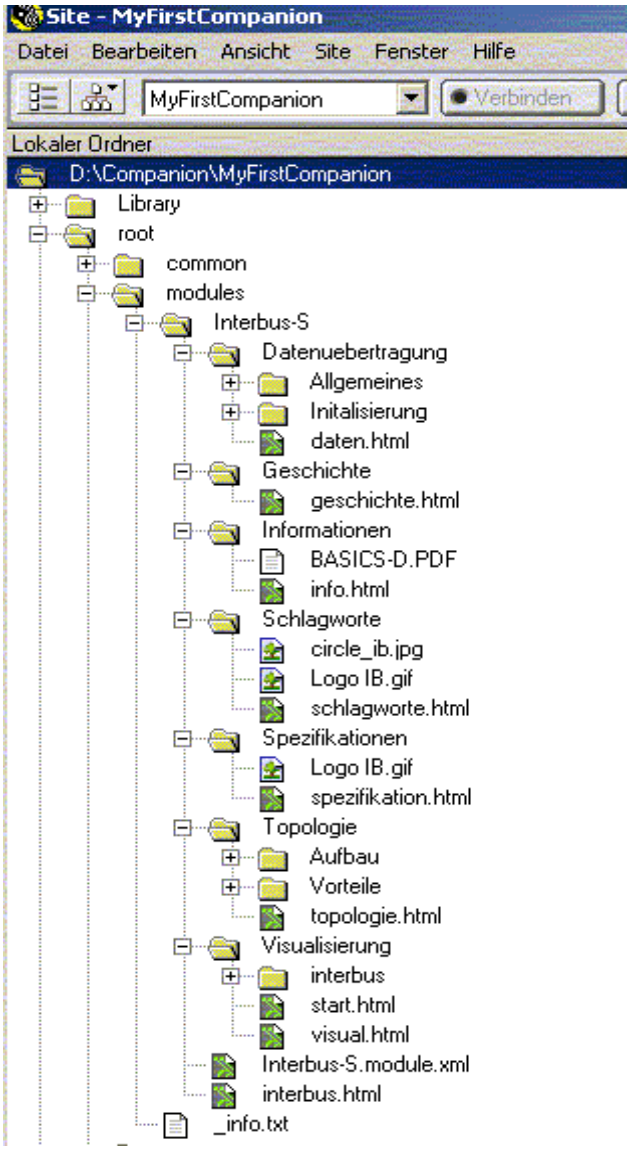
Companion ist ein Autorensystem, das an der Universität Karlsruhe entwickelt wurde. Es gibt den Autoren einige Vorlagen an die Hand, die dann mit Hilfe des Macromedia Dreamweavers bearbeitet werden können. Dazu wurden eigens Erweiterungen entwickelt, die man in den Dreamweaver integrieren kann. Dies ermöglicht, dass eine Vielzahl von Autoren ihre Lerneinheiten in eine einheitliche Benutzeroberfläche einfügen können.
Die Lernkapitel der einzelnen Autoren werden als Module realisiert, die sich dann beliebig miteinander verknüpfen lassen. So gibt es z.B. auch die Möglichkeit mehrere Module zu einer Tour zusammen zu fassen.

3.2 Einrichtung von Companion

Um ein leeres Basissystem auf dem eigenen Rechner zu installieren, muss folgender Maßen vorgegangen werden:
Zunächst muss der Macromedia Dreamweaver installiert werden. Sehr verbreitet und optimal von Companion unterstützt ist hierbei die Version 2. Für die Erstellung dieser Diplomarbeit wurde die Version 3 eingesetzt und es konnten keine Probleme im Zusammenspiel mit den Companion-Erweiterungen festgestellt werden.
Wurde dieser Schritt erfolgreich abgeschlossen, folgt die Installation des Companion Base Systems.
Anschließend müssen noch die Companion-Erweiterungen für den Dreamweaver installiert werden und schon kann damit begonnen werden, ein Modul zu erstellen.

4 Erstellen der Lerneinheit

4.1 Die Struktur des Interbus-S-Moduls



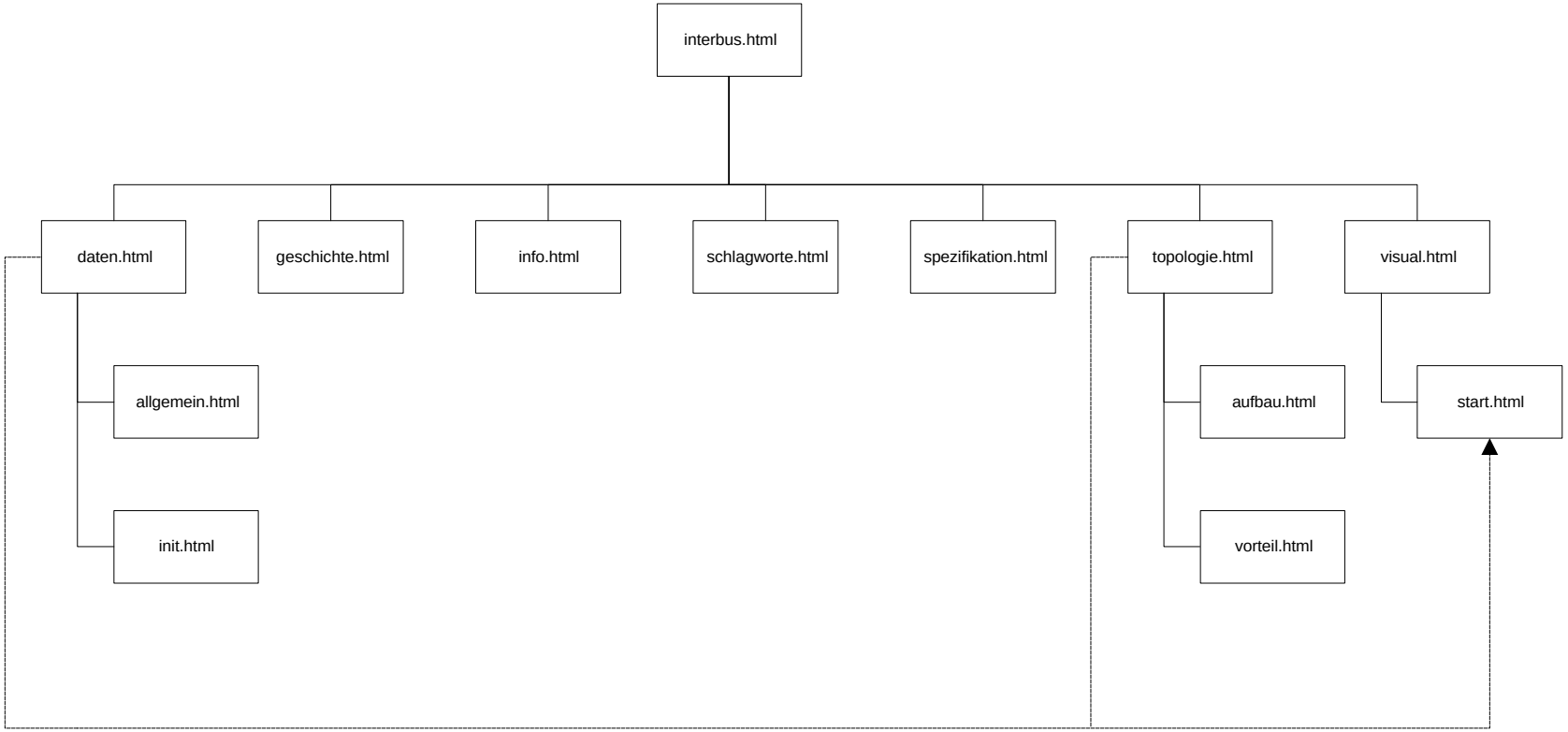
Grafik 1: Die Struktur des Lernmoduls

Der Aufbau des Moduls wird aus der nebenstehenden Grafik ersichtlich.
Die Untergliederung dieser Lerneinheit umfasst die folgenden Unterkapitel:

- Datenübertragung
  - Allgemeines
  - Initialisierung
- Geschichte
- Informationen
- Schlagworte
- Spezifikationen
- Topologie
  - Aufbau
  - Vorteile
- Visualisierung

Zusätzlich zu diesen Kapiteln gibt es noch ein übergreifendes Glossar.

4.2 Die Dokumentenstruktur



Grafik 2: Die Dokumentenstruktur

4.3 Ein Inhaltsdokument

Um ein Inhaltsdokument zu erstellen muss im Dreamweaver so vorgegangen werden: Im Menü „Datei“ den Befehl „Neu von Vorlage“ wählen. Im darauf erscheinenden Fenster „Inhaltsdokument“ auswählen und dies mit „OK“ bestätigen.
Jetzt öffnet sich die Vorlage, die schon eine Reihe von Kommentaren zur Benutzung enthält.

Der erste Schritt sollte sein, die Seite in einem passenden Verzeichnis zu speichern. Wichtig hierbei ist, dass die Dateien die Endung **.html** tragen müssen.
Danach müssen noch die Eigenschaften des Dokuments über „Ändern ✍ Document Properties“ angepasst werden.

An dieser Stelle soll nicht näher auf die Erstellung einer Seite eingegangen werden, sondern es wird auf das Companion-Handbuch verwiesen, das unter http://vikar.ira.uka.de/intern/downloads/companion/companion.1.2/index.html heruntergeladen werden kann.

In diesem Handbuch wird ausführlich erläutert, wie die einzelnen Dokumente wie Inhaltsverzeichnis, Inhaltsdokument oder Glossar erstellet werden können.

4.4 Hinweise

Folgende Hinweise sollten beachtet werden, wenn einem einige Probleme ersparen bleiben sollen, die aus dem Umgang mit dem Dreamweaver oder dem Netscape Navigator resultieren.

- ✍ Die Namen der Inhaltsdokumente und der Links innerhalb eines Moduls dürfen keine Umlaute enthalten, da dies Probleme mit dem Navigator von Netscape mit sich bringt. Dies gilt übrigens auch für Verweise auf Glossareinträge.
- ✍ Nach dem Start von Dreamweaver sollten einige Sekunden vergehen bevor mit dem Arbeiten begonnen wird, da sonst die Companion-Erweiterungen nicht richtig initialisiert werden und sich die Fehlermeldungen beim Bearbeiten von Dokumenten häufen.
- ✍ Damit die Seiten in der Lernumgebung richtig dargestellt werden, muss noch die Datei *SourceFormat.profile* im Ordner „Configuration“ von Dreamweaver gelöscht und die dort vorhandene Datei *SourceFormat.profile.new* in *SourceFormat.profile* umbenannt werden.
- ✍ Äußerst wichtig ist es auch, nach jedem neu angelegten Dokument und jedem neu eingefügten Link den Befehl „Befehle ✍ Update Dokument Database“ auszuführen, damit die Verknüpfungen richtig verwaltet werden.

5 Beschreibung der Kapitel

Gestartet wird die Lerneinheit durch einen Klick auf die Datei *index.html* im Ordner *MyFirstCompanion*.

5.1 Das Kapitel „Geschichte“

In diesem Kapitel wird ein kurzer Überblick über die Geschichte des Interbus-S gegeben. Dabei wird sowohl auf die allgemeine Entwicklung von Bussystemen als auch die Historie des Interbus-S im Speziellen eingegangen.
Hier kann der Lernende auch nachlesen, welche Normen dem Interbus-S zu Grunde liegen.
Die Entwicklung der Bussysteme wird in einer guten Vorlesung über die Automatisierungssysteme immer sehr ausführlich erörtert werden. Deshalb wurden hier nur die wirklich wichtigen Meilensteine in der Entstehung des Interbus dargestellt.

5.2 Das Kapitel „Schlagworte“

In diesem Abschnitt kann sich der Studierende über die wichtigsten Daten des Interbus informieren. Sie sind in Stichworten in Form einer Liste dargestellt. Diese Art der Darstellung wurde gewählt, um dem Lernenden die Möglichkeit zu bieten, sich schnell zu informieren, ohne vom Wesentlichen abgelenkt zu werden.

5.3 Das Kapitel „Spezifikationen“

Hier können die wichtigsten Spezifikationen des Interbus-S nachgelesen werden. Die Informationen sind in einer übersichtlichen Tabelle zusammengestellt. Hauptsächlich wurde diese Seite erstellt, um kompatibel zu bleiben gegenüber den bereits erstellten Lernmodulen im Kapitel *Automatisierungssysteme*. Trotzdem können auch aus dieser Tabelle noch einige zusätzliche Informationen zum Interbus erhalten werden.

5.4 Das Kapitel „Topologie“

In diesem Kapitel wir der Aufbau des Busses beschrieben. Es wird erklärt, wie die Stationen miteinander verbunden sind und welche Vorteile die Ringstruktur beim Interbus-S mit sich bringt. Der besseren Übersicht wegen ist dieses Kapitel demnach auch in die 2 Unterkapitel *„Die Struktur des Interbus-S“* und *„Die Vorteile der Ringstruktur“* unterteilt worden.
Zum besseren Verständnis bietet sich die Möglichkeit eine Simulation aufzurufen, die die Vorgänge im Bus darstellt.

5.5 Das Kapitel „Datenübertragung“

Dieses Kapitel dient dazu zu erfahren, wie die Datenübertragung beim Interbus-S funktioniert. Um die beiden möglichen Betriebsfälle Normalbetrieb und Störung auch logisch von einander zu trennen, ist das Kapitel in die beiden Bereiche „Initialisierung“ und „Allgemeines“ untergliedert. Auch von hier aus lässt sich die Visualisierung aufrufen.

5.5.1 Das Unterkapitel „Allgemeines“

Da die allgemeinen Informationen zur Datenübertragung relativ umfangreich sind, wurden die Daten wiederum in 5 Abschnitte unterteilt. Diese sind:

- **Allgemeines**
  Hier ist zu erfahren, wie das Übertragungsprotokoll des Interbus aufgebaut ist.
- **Der Aufbau eines Interbus-Teilnehmers**
  Dieser Abschnitt beschreibt den inneren Aufbau einer Station. Er dient dazu die einzelnen Elemente der Buskommunikation in einer Station kennen zu lernen.
- **Die Funktion des Summenrahmens**
  Um einen Überblick über die Funktion des Summenrahmens zu bekommen, kann man sich diesen Abschnitt durchlesen. So lernt man die Art der Datenübertragung beim Interbus besser kennen.
- **Die Sicherung der Daten**
  Hier wird beschrieben, wie beim Interbus Übertragungsfehler erkannt werden können. Es wird gezeigt, wie die Überprüfung der gebildeten Prüfsumme erfolgt.
- **Die Effizienz des Interbus-S Protokolls**
  Schließlich wird noch auf die Effizienz des Protokolls im Vergleich zu nachrichtenorientierten Protokollen eingegangen.

Über die Links am Beginn des Dokuments können die einzelnen Überschriften direkt angesprungen werden. Mit Hilfe des kleine blauen Pfeils am Ende jedes Abschnitts kann wieder der Beginn des Dokuments erreicht werden.

5.5.2 Das Unterkapitel „Initialisierung“

Hier wird beschrieben, wie die automatische Konfiguration des Busses beim Einschalten funktioniert. Dieses Unterkapitel wurde aufgenommen, da diese Form der Konfiguration eine Besonderheit des Interbus ist.

5.6 Das Kapitel „Informationen“

Hier werden dem Studierenden einige Quellen an die Hand gegeben, wo er sich weiter führende Informationen besorgen kann. Weiterhin wird noch eine pdf-Datei angeboten, die vom Interbusclub herausgegeben wurde und die wichtigsten Basisinformationen zum Interbus liefert.

5.7 Das Kapitel „Visualisierung“

Auf dieser Seite erwartet den Studenten ein interaktives JAVA-Applet, das ihm die Vorgänge beim Interbus-S anschaulich näher bringen soll. Auf dieses Applet wird in den nächsten Kapiteln dieser Diplomarbeit noch näher eingegangen.

6 Die Visualisierung

6.1 Allgemeines

Die Visualisierung der Datenübertragung beim Interbus-S wurde als JAVA-Applet erstellt. Da der Autor zu diesem Zweck erst JAVA gelernt hat, ist daraus keine perfekte Darstellung der Vorgänge geworden – vielmehr musste ein Kompromiss gefunden werden, zwischen dem, was programmiert werden sollte, und dem was in der zur Verfügung stehenden Zeit in die Tat umzusetzen war.
Dennoch lassen sich die Abläufe gut verstehen. Natürlich hätte mit einer Programmiersprache wie JAVA erheblich mehr Interaktivität eingebaut werden können.
Der Umfang an Funktionen und Klassen, der von SUN zu JAVA mitgeliefert wird, ist aber zu umfangreich, um ihn innerhalb von 3 Monaten überschauen zu können. Dies trifft noch vielmehr zu, da die Programmierung nicht die einzige Aufgabe dieser Diplomarbeit dargestellt hat.

Hinweis: Zur optimalen Darstellung des Applets wird die Verwendung des Internet Explorers 5.x empfohlen. Dieser Browser stellt das Applet auf jeden Fall richtig dar, während es bei anderen zu Fehldarstellungen kommen kann.

6.2 Einrichten der Programmierumgebung

Die Programmierumgebung lässt sich recht einfach einrichten. Zuerst wird das Programm VisualAge for Java 2 von IBM auf einem Rechner installiert. Nach dem Start wählt man im Auswahlmenü „Create Applet“ und folgt den Anweisungen des Assistenten. Schon hat man das Grundgerüst seines Applets und kann anfangen zu Programmieren.
Anfängern, die noch nie mit VisualAge gearbeitet haben, wird empfohlen, das Tutorial, welches im Paket enthalten ist, durch zu arbeiten.

6.2.1 Das Verwenden einer IDE

Eine IDE (integrated development environment) ist zwar sehr praktisch, da viele Code-Zeilen nicht selbst geschrieben werden müssen. Trotzdem muss erwähnt werden, dass das Verwenden einer solchen IDE nicht gerade dazu beiträgt, einen übersichtlichen und gut strukturierten Code zu erhalten. Vor Allem die Namensgebung der Klassen und Methoden aus den automatisch erstellten Programmteilen ist oft eher verwirrend. Teilweise kann dadurch auch das Konzept der Ungarischen Notation nicht eingehalten werden.

6.2.2 Eigenschaften von VisualAge

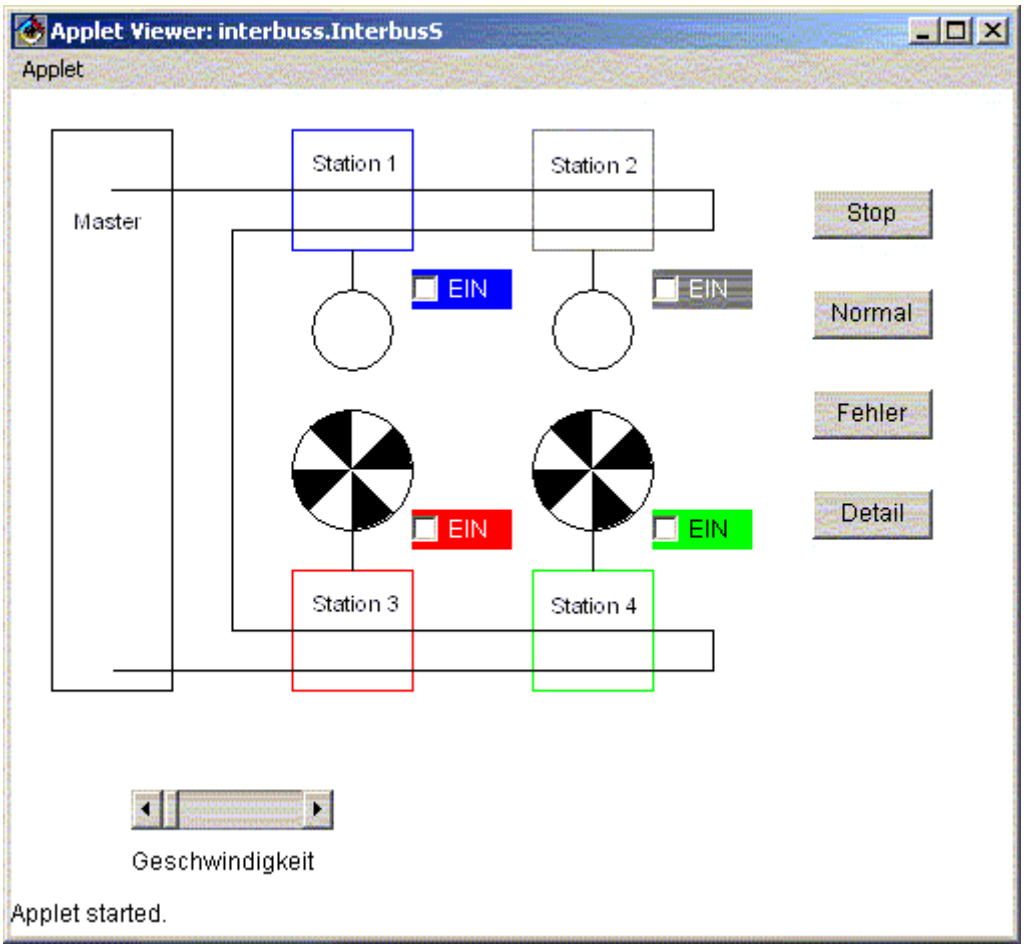
Folgende Punkte sollten bei der Arbeit mit VisualAge im Hinterkopf behalten werden:

- Der Source-Code wird nicht in einer Datei, sondern in einer Datenbank gespeichert. Soll also der ganze zusammenhängende Code betrachtet werden können, so muss er zunächst in eine Datei exportiert werden. Durch diese Form der Verwaltung ergibt sich aber auch noch ein 2. Problem: Jede einzelne Variable bekommt im sogenannten Repository einen eigenen Eintrag und wird auch im Verzeichnisbaum einzeln aufgeführt. Was für Klassen und Methoden noch ganz angenehm ist wird in diesem Umfang aber schnell unübersichtlich.
- Das Tool bietet eine integrierte Versionsverwaltung. Diese ist sehr hilfreich, wenn umfangreichere Änderungen wieder rückgängig gemacht werden sollen.
- Viele JAVA-Beans (Buttons, Checkboxen usw.) lassen sich visuell programmieren. **Wichtig:** Die so erstellten Code-Teile werden nach jeder visuellen Programmiertätigkeit neu geschrieben. Wenn innerhalb dieser automatisch generierten Funktionen eigener Code benötigt wird, so muss dieser zwischen den „user code“-Tags eingefügt werden, damit er nicht überschrieben wird. Diese Tags werden in den meisten Fällen auch automatisch erzeugt.

7 Der grafische Aufbau des Applets

7.1 Die Ansicht im Normalfall

Der folgende Screenshot zeigt, wie das fertige Applet im Normalfall aussieht.



Grafik 3: Das Applet im Normalfall

7.1.1 Beschreibung für den Normalfall

Hier werden nun kurz die einzelnen Elemente der Normalansicht beschrieben.

Das Rechteck „Master“

Dieses Rechteck soll den Master des Busses symbolisieren. Sein Datenpaket (das Loopbackwort) wird seiner Rahmenfarbe entsprechend in schwarz dargestellt.

Die Rechtecke „Station x“

Sie stellen symbolisch 4 Bus-Teilnehmer dar. Ihre Datenpakete sind auch entsprechend ihrer Rahmenfarbe gefärbt, damit sie leichter zu unterscheiden sind.

Die runden Elemente

Diese Objekte stellen Aktoren dar, die im Verlauf der Simulation animiert werden können. Die kleineren Kreise symbolisieren LEDs und die größeren Motoren.

Die Buttons

Mit Hilfe der Buttons kann ausgewählt werden, welcher Modus dargestellt wird. Dabei stehen die folgenden Möglichkeiten zur Verfügung:

Tabelle 1: Die Funktion der Buttons

Stop	Dieser Button erzeugt das stehende Bild, wie es der Screenshot zeigt. Gleichzeitig werden alle Variablen auf ihre Initialisierungswerte zurückgesetzt.
Normal	Dieser Button erlaubt die Ansicht des Busses während des Normalbetriebs.
Fehler	Mit diesem Button kann man die Ansicht auswählen, in der ein Ausfall der 2. Station simuliert wird.
Detail	Um eine einzelne Station groß heraus zu zeichnen, wählt man diesen Button. Das erlaubt, sich die Vorgänge in einer Station während des Normalbetriebs anzusehen.

Die Checkboxen

Die Checkboxen erlauben es dem Betrachter auszuwählen, welche der Aktoren animiert werden sollen. Dazu sind sie farblich an die Rahmen der einzelnen Stationen angepasst.

Die Scrollbar

Mit Hilfe der Scrollbar kann die Geschwindigkeit der Animation verändert werden. Mögliche Werte sind von 50 ms bis 1000 ms. Die Unterteilung erfolgt in 10er-Schritten.

7.1.2 Die Koordinaten der Elemente

Falls nachträglich die Ansicht des Applets verändern werden soll, so helfen diese Tabellen, aus denen die Koordinaten der einzelnen Elemente entnommen werden können. Die Größe des gesamten Applets ist übrigens BxH: 500x400 und der Ursprung des Koordinaten-Systems liegt in der linken oberen Ecke

Die Rechtecke und Kreise

Tabelle 2: Die Koordinaten der Rechtecke (Normalfall)

Objekt	x-Koordinate	y-Koordinate	Breite	Höhe
Master	20	20	60	260
Blaue Station	140	20	60	60
Graue Station	260	20	60	60
Rote Station	140	240	60	60
Grüne Station	260	240	60	60
Stop-Button	400	50	60	25
Normal-Button	400	100	60	25
Fehler-Button	400	150	60	25
Detail-Button	400	200	60	25
Blaue Box	200	90	50	20
Graue Box	320	90	50	20
Rote Box	200	210	50	20
Grüne Box	320	210	50	20
Die Scrollbar	60	350	101	20
Linke LED	150	100	40	40
Rechte LED	270	100	40	40
Linker Motor	140	160	60	60
Rechter Motor	260	160	60	60

Die einzelnen Segmente der Motoren werden wie folgt bestimmt:

(x-Koordinate des vollen Kreises, y-Koordinate d.v.K., Breite d.v.K., Höhe d.v.K., Startwinkel, Endwinkel)

Die Winkel werden gegen den Uhrzeigersinn positiv gezählt. 0° entspricht der 3-Uhr-Stellung eines Uhrzeigers.

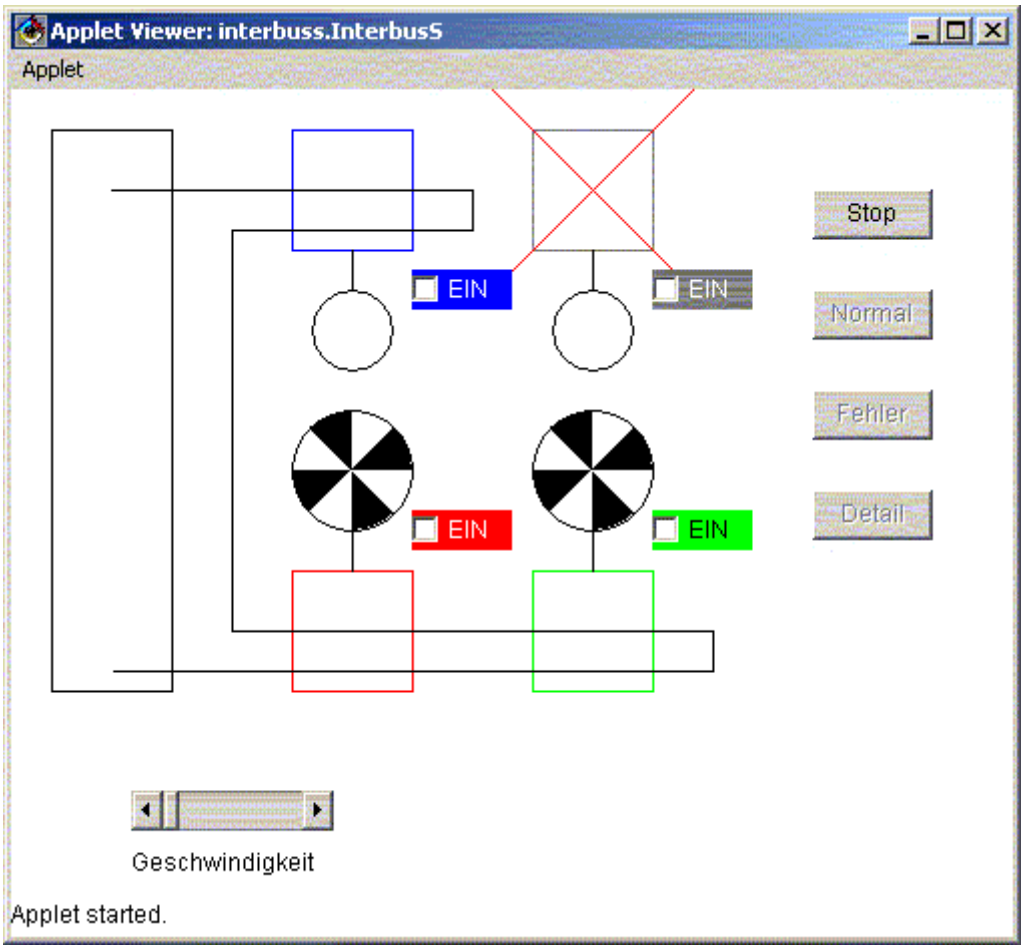
Die Linien

Tabelle 3: Die Koordinaten der Linien (Normalfall)

Von	Nach	Punkte x,y
Master	Station 1	50,50 / 170,50
Station 1	Station 2	170,50 / 290,50
Station 2	Station 3	290,50 / 350,50 / 350,70 / 110,70 / 110,270 / 170,270
Station 3	Station 4	170,270 / 290,270
Station 4	Master	290,270 / 350,270 / 350,290 / 50,290
Station 1	Linke LED	170,80 / 170,100
Station 2	Rechte LED	290,80 / 290,100
Station 3	Linker Motor	170,220 / 170,240
Station 4	Rechter Motor	290,220 / 290,240

7.2 Die Ansicht im Fehlerfall

So wird das Applet im Fehlerfall dargestellt:



Grafik 4: Das Applet im Fehlerfall

7.2.1 Beschreibung für den Fehlerfall

Für den Fehlerfall gilt im Prinzip die Beschreibung des Normalfalls. Lediglich in 2 Punkten unterscheiden sich die Ansichten:

- Die 2. Station wird als defekt angenommen und nimmt nicht am Busgeschehen teil.
- Die Checkbox für die Animation der rechten LED lässt sich nicht aktivieren.

7.2.2 Die Koordinaten der Elemente

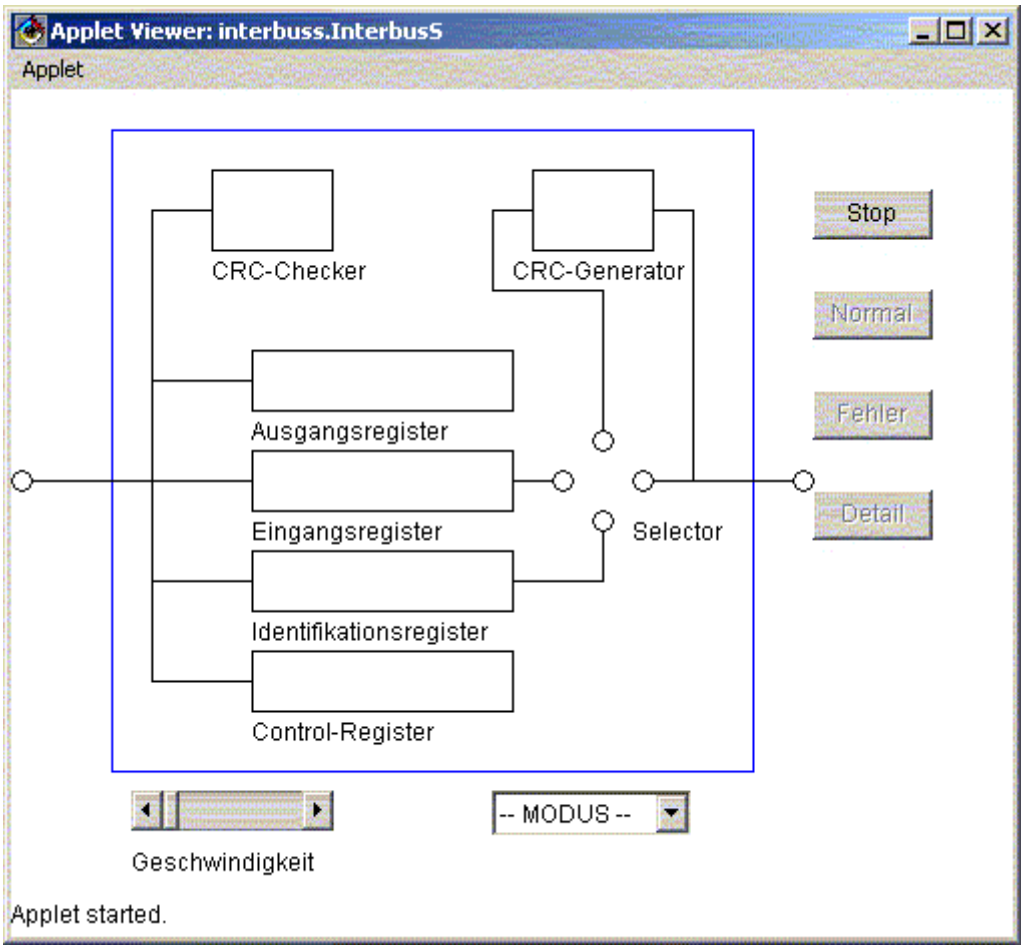
Die folgenden Tabellen zeigen die Koordinaten der Elemente, die nicht mit dem Normalfall überein stimmen.

Tabelle 4: Die Koordinaten der Elemente (Fehlerfall)

Objekt	Punkte x,y
Linie von Station 1 nach Station 3	170,50 / 230,50 / 230,70 / 110,70 / 110,270 / 170,270
1. Gerade des roten „X“	240,0 / 340,100
2. Gerade des roten „X“	240,100 / 340,0

7.3 Die Detailansicht

Das folgende Bild sieht der Anwender, wenn er die Detailansicht wählt:



Grafik 5: Das Applet der Detailansicht

7.3.1 Beschreibung für die Detailansicht

Das Bild sollte eigentlich selbsterklärend sein. Deshalb wird hier nur auf die Choice-List hingewiesen, mit deren Hilfe man in der Detailansicht 3 Modi auswählen kann:

- Initialisierungs-Modus
- Daten-Modus
- CRC-Check-Modus

Nach der Auswahl des Modus startet die Simulation automatisch.

7.3.2 Die Modi der Detailansicht

Tabelle 5: Die Modi der Detailansicht

Modus	Funktion
Initialisierung	Zeigt die Vorgänge während des Initialisierungs-Vorgangs.
Datenmodus	Stellt den Datenverkehr im Normalfall dar.
CRC-Check	Zeigt die Check-Sequenz, die der Übertragungs-Sicherheit dient.

7.3.3 Die Koordinaten der Elemente

Wie schon bei den beiden vorhergehenden Ansichten werden hier die Koordinaten der einzelnen Elemente aufgelistet.

Die Rechtecke und Kreise

Tabelle 6: Die Koordinaten der Elemente (Detailansicht)

Objekt	x-Koordinate	y-Koordinate	Breite	Höhe
Blauer Rahmen	50	20	320	320
1. Kreis	0	190	10	10
2. Kreis	270	190	10	10
3. Kreis	290	170	10	10
4. Kreis	290	210	10	10
5. Kreis	310	190	10	10
6. Kreis	390	190	10	10
CRC-Checker	100	40	60	40
CRC-Generator	260	60	40	60
Ausgangsregister	120	130	130	30
Eingangsregister	120	180	130	30
Identifikationsregister	120	230	130	30
Control-Register	120	280	130	30
Choice-List	240	350	100	40

An den Koordinaten der Buttons und der Scrollbar hat sich nichts verändert, die Checkboxen wurden für diese Ansicht ausgeblendet.

Die Linien

Tabelle 7: Die Koordinaten der Linien (Detailansicht)

Objekt	Punkte x,y
1. Linie	10,195 / 70,195
2. Linie	70,60 / 100,60
3. Linie	70,145 / 120,145
4. Linie	70,195 / 120,195
5. Linie	70,245 / 120,245
6. Linie	70,295 / 120,295
7. Linie	250,195 / 270,195
8. Linie	320,195 / 390,195
9. Linie	70,60 / 70,295
10. Linie	260,60 / 240,60 / 240,100 / 295,100 / 295,170
11. Linie	320,60 / 340,60 / 340,195
12. Linie	250,245 / 295,245 / 295,220

Die Texte

Tabelle 8: Die Koordinaten der Texte (Detailansicht)

Text	x-Koordinate	y-Koordinate
Ausgangsregister	120	175
Eingangsregister	120	225
Identifikationsregister	120	275
Control-Register	120	325
CRC-Checker	100	95
CRC-Generator	250	95
Selector	310	225

8 Die Klassen und Methoden

8.1 Übersicht

In diesem Kapitel werden alle Klassen und Methoden ausführlich dargestellt. Dabei wurde bewusst darauf verzichtet, einige der automatisch erstellten Methoden ausführlicher zu beschreiben, da diese je nach Programmiertool sehr differieren können.
Außerdem soll an dieser Stelle auch noch auf die automatisch erstellte Dokumentation im Ordner *javadoc* auf der zur Diplomarbeit mitgelieferten CD hingewiesen werden. Diese Dokumentation entspricht den Vorgaben von SUN, die aus diesem Grund einen javadoc-Generator in ihre Entwicklungsumgebung mit implementiert hat.

8.1.1 Zusammenfassung der Methoden

Tabelle 9: Übersicht über die verwendeten Methoden

Methode
checkStat2_ItemStateChanged
detailAuswahl_ItemStateChanged
xxxbutton_MouseClicked
doAnimation
drawText
paint
paint<Text>(1)

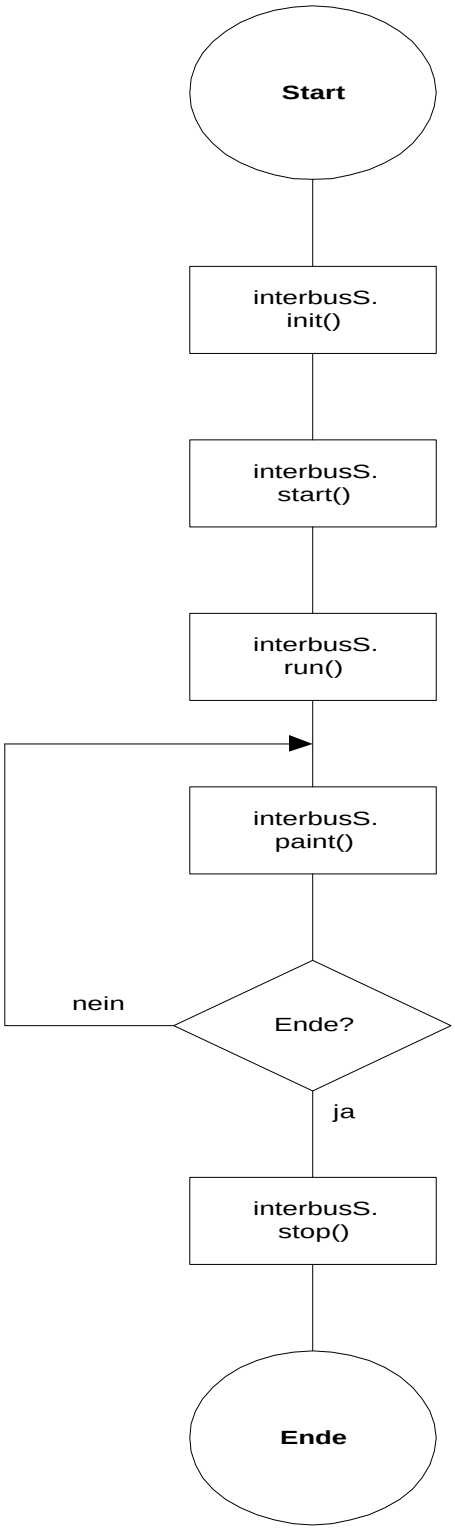
(1)paint<Text> steht für alle Methoden, die mit paint beginnen.

8.2 Die Klasse *InterbusS*

Diese Klasse beinhaltet das gesamte Applet. In ihrem Inneren liegen also alle Methoden, die notwendig sind, um das Applet darzustellen. Ebenso werden hier die Paket globalen Variablen deklariert und initialisiert.
Sie ist direkt von der Klasse *Applet* (java.applet.Applet) abgeleitet. Außerdem werden die beiden Schnittstellen *MouseListener* und *Runnable* eingebunden.

Der *MouseListener* ermöglicht es, auf Aktionen mit der Mouse zu reagieren.
Die Schnittstelle *Runnable* wird benötigt, damit das Applet in einem eigenen Thread ablaufen kann.

8.2.1 Der Programmablaufplan



Grafik 6: PAB der Klasse InterbusS

8.3 Die Klasse *ivjEventHandler*

Hier haben wir das erste Beispiel eines Code-Abschnitts, der automatisch durch VisualAge erzeugt wurde. Es handelt sich hierbei um eine innere Klasse, die alle Events abfängt und verarbeitet. Im vorliegenden Applet reagiert diese Klasse nur auf die Mausklicks auf die Buttons oder auf den sich ändernden Status der Checkboxen, der Scrollbar und der Choice-List.

8.4 Die Methode *checkStat2_ItemStateChanged*

Diese Methode wurde unter der visuellen Programmieroberfläche mit Hilfe des Assistenten eingebunden. Sie sorgt dafür, dass die graue Checkbox im Fehlerfall nicht aktiviert werden kann. Dies wird erreicht, indem der Status der Checkbox immer wieder auf inaktiv gesetzt, sobald sie aktiviert wird:

Als Pseudocode:

```
wenn Fehlerfall{
   wenn Checkbox aktiviert{
      deaktiviere Checkbox
   }
}
```

8.5 Die Methoden *xxxButton_MouseClicked*

In diesen Methoden stehen die Aktionen, die ausgeführt werden sollen, wenn ein Button mit der Maus gedrückt wurde. Dies sind in diesem Applet:

- Buttons aktivieren oder deaktivieren
- Elemente Ausblenden oder Einblenden
  Beispiel: Die Checkboxen werden in der Detailansicht nicht benötigt und deshalb ausgeblendet, dafür wird aber die Choice-List eingeblendet
- Zählvariablen und Merkervariablen auf Startwerte setzen
- Die Variable für die Modus-Auswahl auf den entsprechenden Wert setzen
- Im Falle des Stop-Buttons wird zusätzlich noch der Status der Checkboxen auf inaktiv gesetzt.

8.6 Die Methode *doAnimation*

Mit dieser Methode werden die 4 Aktoren animiert, wenn sie denn dazu ausgewählt sind. Dazu wechselt sie immer 2 verschiedene Ansichten der Aktoren, was den Eindruck einer Bewegung schaffen soll.
Die LEDs werden dabei nur eingeschaltet, da es bei einer hohen Simulationsgeschwindigkeit mit blinkenden LEDs sonst zu einem lästigen Flackern des Applets gekommen wäre.

Als Pseudocode:

```
Wenn linke LED animiert{
   Färbe linke LED
}
Wenn rechte LED animiert{
   Färbe rechte LED
}
Wenn 1.Ansicht{
   Wenn linker Motor animiert{
      Zeichne die 1. Ansicht des Motors
   }
   Wenn rechter Motor animiert{
      Zeichne die 1. Ansicht des Motors
   }
}
Sonst{
   Wenn linker Motor animiert{
      Zeichne die 2. Ansicht des Motors
   }
   Wenn rechter Motor animiert{
      Zeichne die 2. Ansicht des Motors
   }
}
```

8.7 Die Methode *drawText*

Diese Methode zeichnet die Beschriftung der Komponenten in der Detailansicht. Die zu Grunde liegende Funktion stammt aus der Klasse java.awt.Graphics und sieht folgendermaßen aus: **drawString(„Text“, x-Koord., y-Koord.)**

8.8 Die Methode *getAppletInfo*

Hier handelt es sich wiederum um eine von VisualAge erstellte Methode, die Informationen über das Applet wie Titel, Autor und Erstellungsdatum liefert.

8.9 Die Methoden *getxxx*

Diese automatisch generierten Methoden setzen die Parameter für die interaktiven Elemente des Applets (Buttons, Checkboxen, Scrollbar und Choice-List).
Die wichtigsten Parameter sind:

- Name des Objekts
- Koordinaten des Objekts
- Größe des Objekts
- Label des Objekts
- Bei Choice-Lists: Elemente der Liste

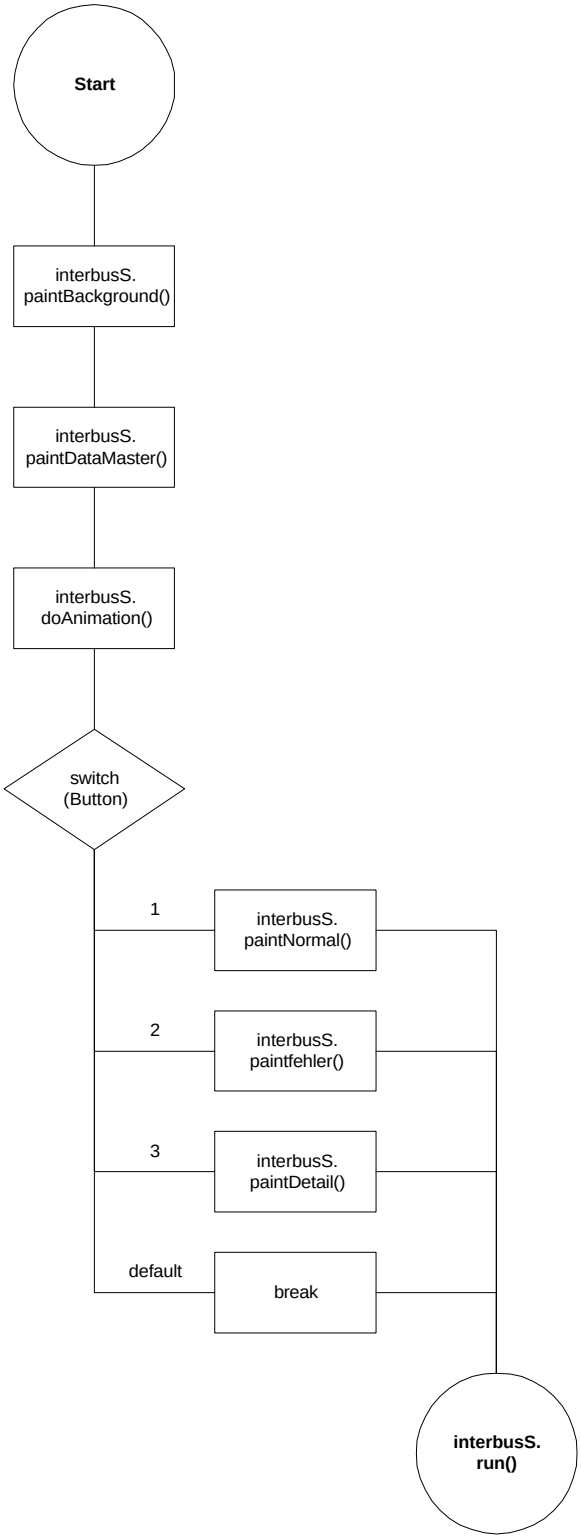
8.10 Die Methode *init*

Diese Methode wird automatisch einmal abgearbeitet, wenn das Applet aufgerufen wird. Sie bestimmt den Namen und die Größe des Applets und fügt die Java-Beans, wie Buttons oder Checkboxen, zum Applet hinzu.
Hier wurde die Hintergrundfarbe des Applets explizit auf weiß gesetzt, da dies nicht bei jedem Browser Standard ist.
Außerdem wird noch über **ivjDetailAuswahl.setVisible(false)** die Choice-List versteckt, da die in der Gesamtansicht des Busses nicht benötigt wird.

8.11 Die Methode *paint*

Sie wird zyklisch aufgerufen und zeichnet das Applet. Hier ruft sie die Methoden **paintBackground, paintDataMaster, doAnimation** und je nach Modus eine der Methoden **paintNormal, paintFehler oder paintDetail** auf.

8.11.1 Der Programmablaufplan



Grafik 7: PAB der Methode paint

8.12 Die Methode *paintBackground*

Diese Methode zeichnet alle statischen Elemente des Applets. Dies sind im einzelnen:

- Die Stationen des Busses
- Die Aktoren im Ruhezustand
- Die Verbindungslinien zwischen den Stationen
- Die Komponenten der Detailansicht

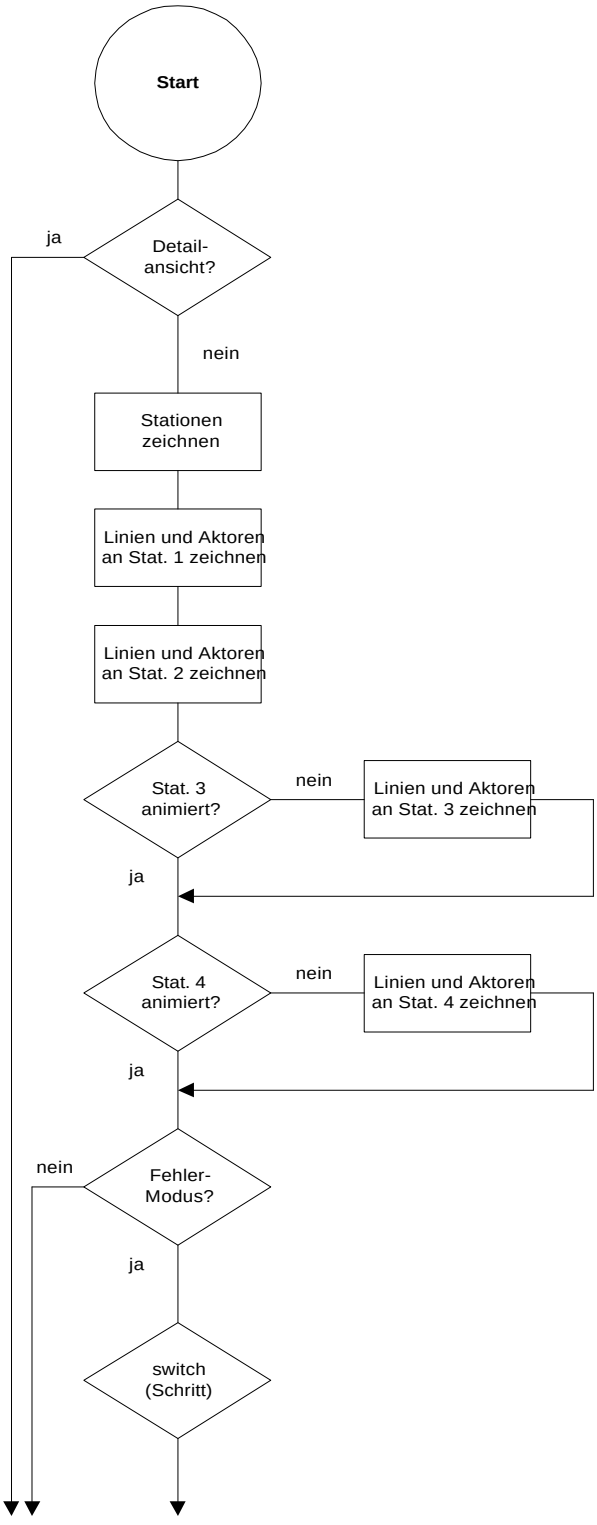
Dabei werden die 3 Fälle Normalzustand, Fehlerfall und Detailansicht unterschieden.

Hier sollen nun noch ein paar häufig auftretende Funktionen beschreiben werden, die verwendet werden um dass Applet zu zeichnen.

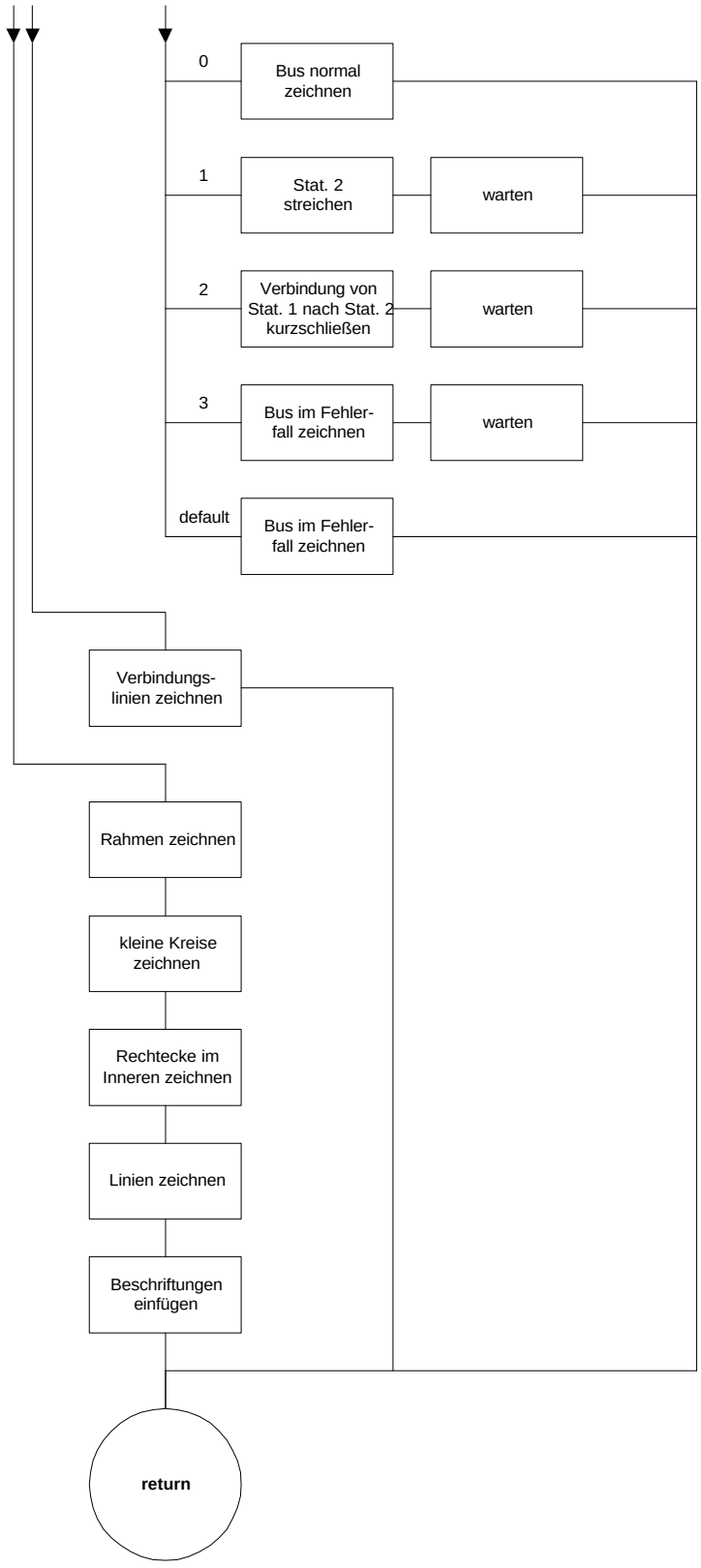
Tabelle 10: Häufig verwendete Funktionen

Funktion	Beschreibung
setColor(Color.<Farbe>)	Setzt die aktuelle Zeichenfarbe auf <Farbe> Einige Standardfarben sind vordefiniert und lassen sich über ihre englische Bezeichnung aufrufen, z.B. Color.blue
drawRect(x, y, width, height)	Zeichnet ein Rechteck mit Ursprung (x,y), der Breite width und der Höhe height.
fillRect(x, y, width, height)	Zeichnet ein Rechteck mit Ursprung (x,y), der Breite width und der Höhe height und füllt es mit der aktuell gewählten Farbe.
drawLine(x1, y1, x2, y2)	Zeichnet eine Linie von (x1,y1) nach (x2,y2)
drawPolyline(xFeld, yFeld, AnzPunkte)	Zeichnet eine Polygonlinie deren Punkte aus den entsprechenden Paaren der Felder xFeld und yFeld bestehen. AnzPunkte gibt an aus wie vielen Punkten die Linie aufgebaut ist
drawOval(x, y, width, height)	Zeichnet eine Oval. x und y beschreiben den linken oberen Punkt eines Rechtecks, dass das Oval umrahmt. width und height geben die Breite und die Höhe des Ovals an. Sind diese Werte gleich, so erhält man einen Kreis.
fillOval(x, y, width, height)	Füllt das angegebene Oval mit der aktuellen Zeichenfarbe.
drawArc(x, y, width, height, arc1, arc2)	Zeichnet einen Kreisausschnitt. x, y, width, height erfüllen die gleiche Funktion wie bei drawOval. arc1 gibt den Start- und arc2 den Endwinkel des Kreisausschnittes an. Die Winkel werden gegen den Uhrzeigersinn positiv gezählt. 0° entspricht der 3-Uhr-Stellung eines Uhrzeigers.

8.12.1 Der Programmablaufplan



Grafik 8: PAB der Methode paintBackground (1)

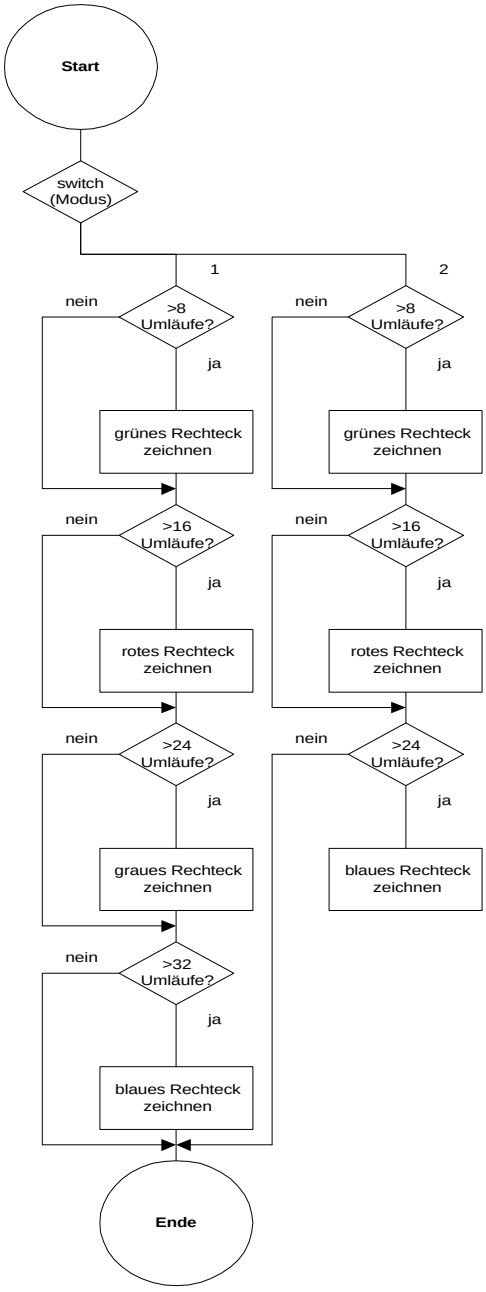


Grafik 9: PAB der Methode paintBackground (2)

8.13 Die Methode *paintDataMaster*

Abhängig von der Anzahl der Durchläufe, zeichnet diese Methode die farbigen Datenpakete, die der Master während des aktiven Busbetriebs zwischenspeichert. Innerhalb der Methode wird zwischen Normal- und Fehlerfall unterschieden.

8.13.1 Der Programmablaufplan



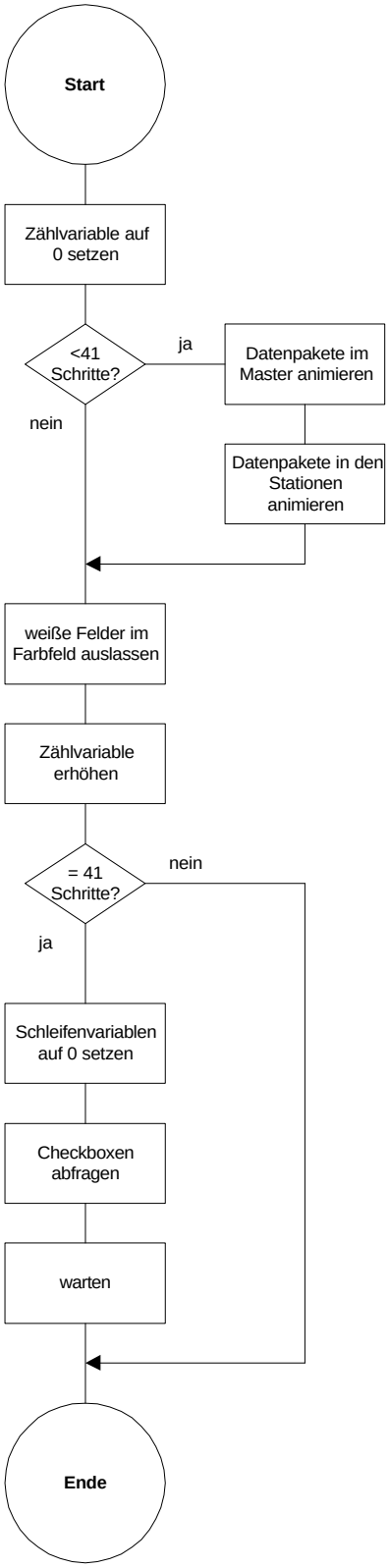
Grafik 10: PAB der Methode paintDataMaster

8.14 Die Methode *paintDetail, paintNormal und paintFehler*

In diesen 3 Methoden werden die Datenpakete animiert. Während dies in den Methoden *paintNormal* und *paintFehler* nur ein zyklisches abarbeiten der Befehle ist, so wird in der Methode *paintDetail* noch zwischen den 3 Modi Initialisierung, Datenmodus und CRC-Check unterschieden.
Ein Aufspalten der Animation in diese 3 Teilpakete wurde gewählt, da sie von einander unabhängig sind und somit auch in anderen Simulationen eingesetzt werden könnten.

8.14.1 Programmablaufplan *paintNormal, paintFehler*

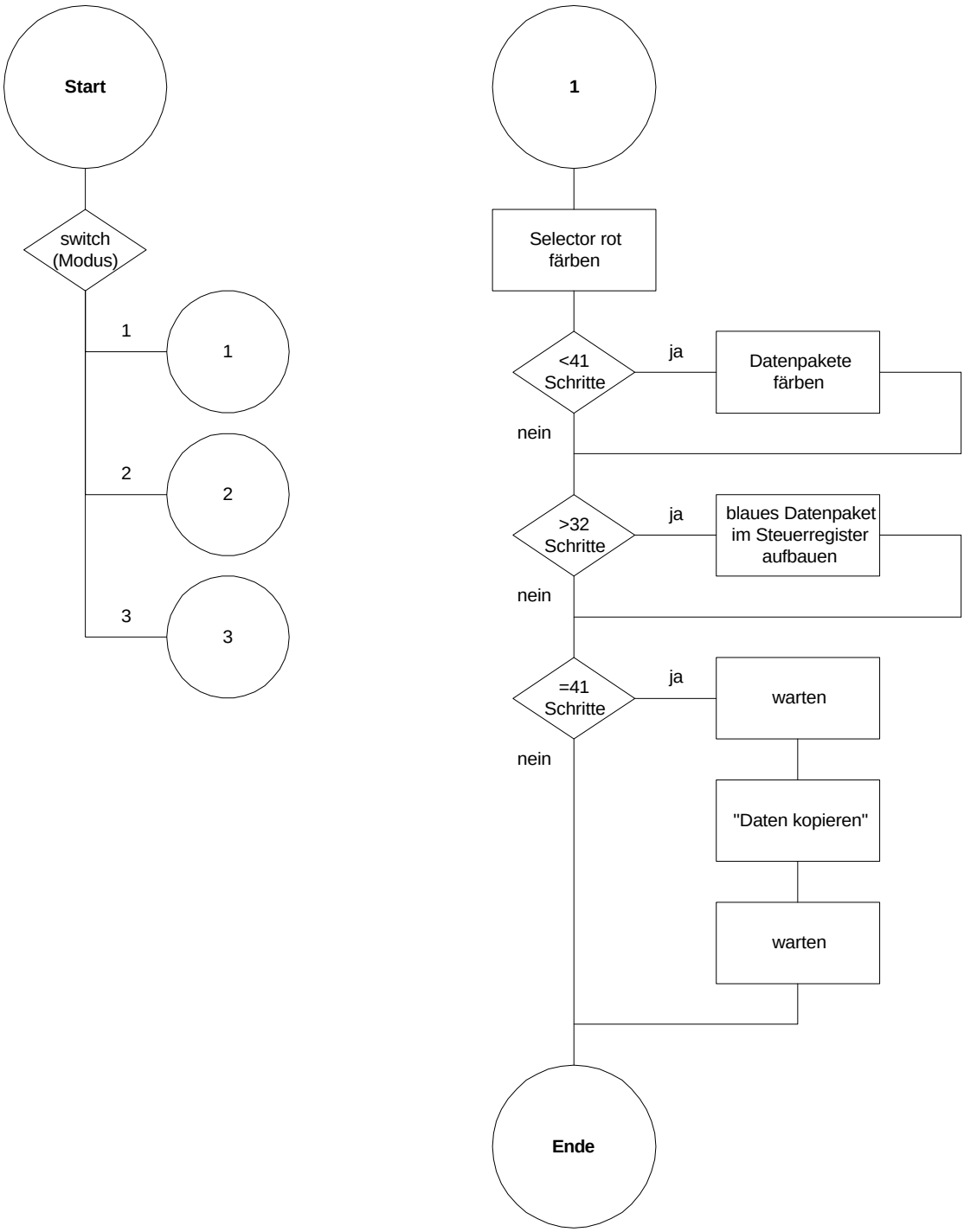
Hinweis: Der PAP der Methode paintFehler entspricht im Aufbau genau dem der Methode paintNormal. Sie unterscheiden sich nur in den Farbfeldern auf die zurück gegriffen werden.



Grafik 11: PAB der Methoden paintNormal und paintFehler

8.14.2 Programmablaufplan *paintDetail*

Da sich die 3 Schritte in ihrem Aufbau sehr ähneln, wurde hier beispielhaft nur der Initialisierungsmodus genauer herausgezeichnet.



Grafik 12: PAB der Methode paintDetail

9 Verwendete Algorithmen

9.1 Bewegung der Datenpakete

Um die Datenpakete zu bewegen, habe wurden zunächst umfangreiche Felder definiert, in denen die jeweilige Reihenfolge der Farben hinterlegt ist.
Innerhalb dieser Felder kann man einen Farbindex die gewünschte Farbe ausgewählt werden.

Hinweis: Ein Datenpaket ist aus 8 „Datenhäppchen“ aufgebaut.

Tabelle 11: Die Funktionen der Farbfelder

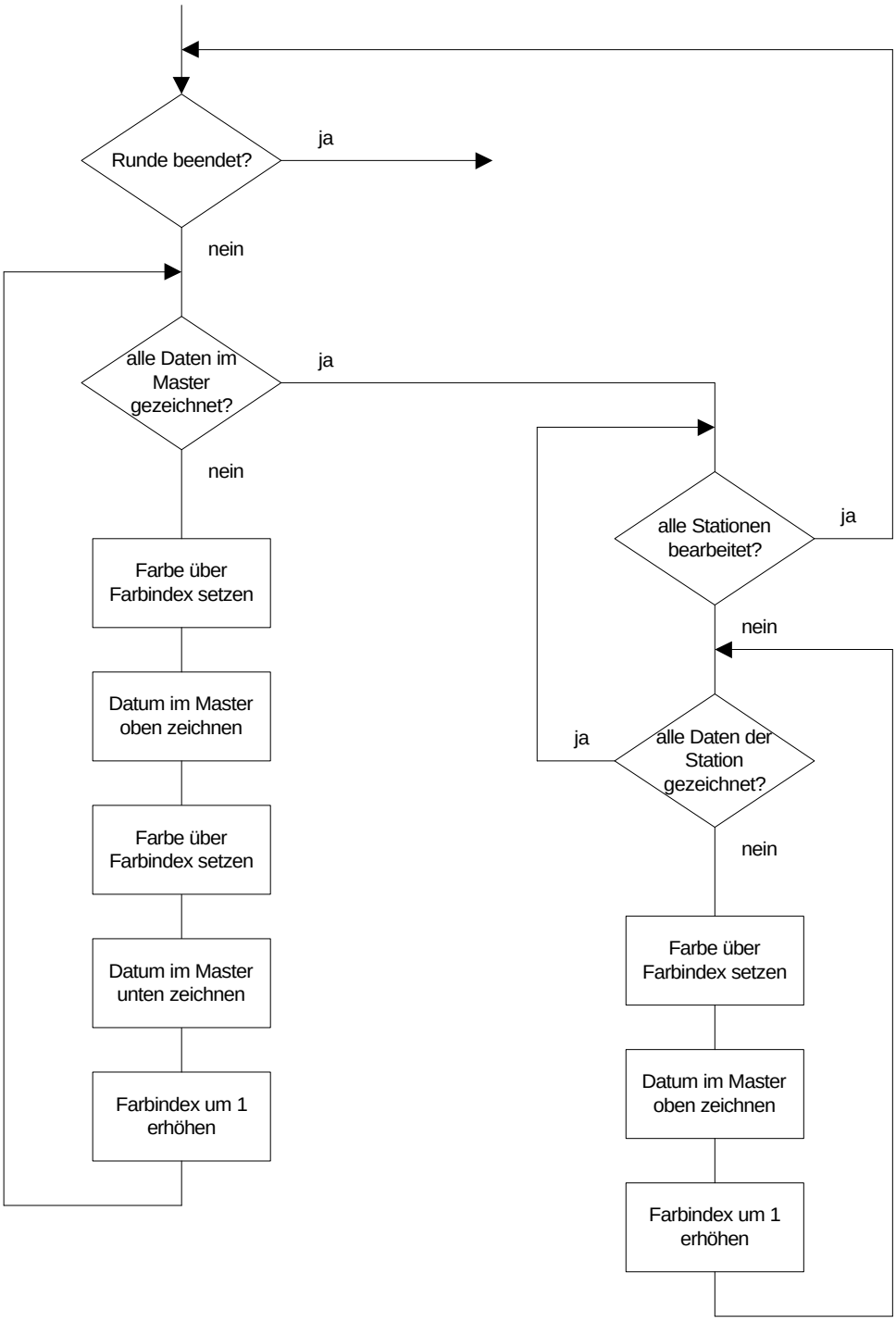
Feldname	Funktion
colStation	Farben für die Darstellung der Datenpakete in den Stationen im Normalmodus
colMaster	Farben für die Darstellung der Datenpakete im Master im Normalmodus
colStationFehler	Farben für die Darstellung der Datenpakete in den Stationen im Fehlermodus
colMasterFehler	Farben für die Darstellung der Datenpakete im Master im Fehlermodus
colAusgang	Farben für die Übernahme des richtig farbigen Datenpakets in die Register
colCRC	Farben für die Darstellung der CRC-Pakete in der Detailansicht
colIdent	Farben für die Darstellung der Datenpakete in den Registern in der Detailansicht

Der Algorithmus sieht eigentlich in allen Fällen von der Struktur her gleich aus. Lediglich in der Detailansicht entfällt das Zeichnen der Daten im Master.

Ablauf:

Zunächst wird überprüft, ob die Datenpakete einmal um den Bus gewandert sind. Ist dies nicht der Fall, so werden zunächst die Datenpakete im Master in der jeweils aktuellen Farbe gezeichnet und der Farbindex des Masters um 1 erhöht. Danach werden die Datenpakete der Stationen, eins nach dem anderen, aus den jeweils 8 Daten in den entsprechenden Farben aufgebaut. Schließlich wird noch der Farbindex der Stationen um 1 inkrementiert.

9.1.1 Der Ablaufplan



Grafik 13: PAB des Animations-Algorithmus

10 Verzeichnisse

10.1 Abbildungsverzeichnis

10.2 Tabellenverzeichnis

11 Der Source-Code

Bei einer Diplomarbeit, die sich zum größten Teil mit der Erstellung von Software beschäftigt, darf natürlich auch der Source-Code nicht fehlen.
Um die Übersichtlichkeit noch ein wenig zu verbessern, wurde er direkt aus einem Tool ausgedruckt, welches das Syntax-Highlighting beherrscht. Deshalb fehlen ab der nächsten Seite auch die Seitenzahlen sowie die Kopf- und Fußzeilen.