Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Bachelorthesis

Joannis Geroliolios

Entwicklung einer Testsoftware zur Ansteuerung eines Zusatzrestgeldspeichers mittels seriellem Kommunikationsprotokoll

Fakultät Technik und Informatik
Department Informations- und Elektrotechnik

Faculty of Engineering and Computer Science
Department of Information and Electrical Engineering

Joannis Geroliolios

Entwicklung einer Testsoftware zur Ansteuerung eines Zusatzrestgeldspeichers mittels seriellem Kommunikationsprotokoll

Bachelorthesis eingereicht im Rahmen der Bachelorprüfung
im Studiengang Informations- und Elektrotechnik
am Department Informations- und Elektrotechnik
der Fakultät Technik und Informatik
der Hochschule für Angewandte Wissenschaften Hamburg

Betreuender Prüfer : Prof. Dr.-Ing. Robert Fitz
Zweitgutachter : Prof. Dr. Robert Heß

Abgegeben am 23. Mai 2013

Joannis Geroliolios

Thema der Bachelorthesis

Entwicklung einer Testsoftware zur Ansteuerung eines Zusatzrestgeld-speichers mittels seriellem Kommunikationsprotokoll

Stichworte

Fahrausweisautomat, Bargeldverarbeitung, Zusatzrestgeldspeicher, Serial Compact Hopper, RS-232, ccTalk, DES-Algorithmus

Kurzzusammenfassung

Diese Arbeit beschreibt den Entwurf und die Implementierung einer Test-software für den Zusatzrestgeldspeicher Serial Compact Hopper MK2. Die Testsoftware wurde mittels UML modelliert und in Visual Basic implementiert. Die Kommunikation zwischen der Testsoftware und dem Zusatzrestgeld-speicher verläuft mittels des seriellen Protokolls ccTalk. Das ccTalk-Protokoll ist eine modifizierte Form der seriellen RS-232-Schnittstelle, welche andere elektrische Eigenschaften besitzt. Aus diesem Grund war es notwendig einen RS232-zu-ccTalk-Adapter anzufertigen, welcher als Schnittstelle zwischen der seriellen Schnittstelle des Rechners und dem Zusatzrestgeldspeicher dient.

Joannis Geroliolios

Title of the paper

Development of a test software to control a supplementary coin box via serial communication protocol

Keywords

Ticket vending machine, cash processing, supplementary coin box, Serial Compact Hopper, RS-232, ccTalk, DES-algorithm

Abstract

This thesis describes the design and implementation of a test software for the supplementary coin box Serial Compact Hopper MK2. The test software was designed in UML and written in Visual Basic. The communication between the application and the supplementary coin box operates via the serial protocol ccTalk. The ccTalk protocol is a modified form of the serial RS-232 interface which has different electrical properties. For this purpose it was necessary to manufacture an RS232-to-ccTalk-adapter as an interface between the serial port of the computer and the supplementary coin box.

Inhaltsverzeichnis

Tabellenverzeichnis

Bilderverzeichnis

Abkürzungsverzeichnis

ACK	Acknowledgement
AG	Aktiengesellschaft
ARCNET	Attached Resource Computer Network
ASCII	American Standard Code for Information Interchange
BNK	Banknotenkasse
BNV	Banknotenverarbeitung
ccTalk	coin controls Talk
ct	Cent
DEC	Decimal
DES	Data Encryption Standard
EEPROM	Electrically Erasable Programmable Read Only Memory
FIFO	First In First Out
GmbH	Gesellschaft mit beschränkter Haftung
GND	Ground
HEX	Hexadecimal
HVV	Hamburger Verkehrsverbund
LCD	Liquid Crystal Display
LIFO	Last In First Out
PC	Personal Computer
RGS	Restgeldspeicher
RxD	Receive Data
SCH2	Serial Compact Hopper MK2
SUH	Serial Universal Hopper
TxD	Transmit Data
USB	Universal Serial Bus
ZRS	Zusatzrestgeldspeicher
XGA	Extended Graphics Array

1 Einführung

1.1 Einleitung

Der öffentliche Personennahverkehr leistet einen wichtigen Beitrag zur Infrastruktur und zur Lebensqualität und gehört zur staatlichen Daseinsfürsorge. Gerade in Metropolregionen, wie z. B. Hamburg, wird der öffentliche Personennahverkehr von vielen Menschen täglich genutzt. Ein funktionierendes und gut ausgebautes Bus- und Bahnnetz ist daher unerlässlich. Zur Benutzung des öffentlichen Personennahverkehrs gehört nicht nur das tatsächliche Fahren mit der Bahn oder dem Bus, sondern zunächst das Lösen einer Fahrkarte am Fahrausweisautomaten. Damit der Fahrgast die gewünschte Fahrkarte erhält, bedarf es verschiedener Funktionen im Fahrausweisautomaten. So muss der Automat z. B. in der Lage sein, Münzen anzunehmen und Wechselgeld in richtiger Höhe auszugeben. Die Wechselgeldausgabe erfolgt durch eine bestimmte Komponente des Automaten. Diese wird als Zusatzrestgeldspeicher, im nachfolgenden auch Hopper genannt, bezeichnet. In den Automaten der Hamburger HOCHBAHN AG werden die Hopper der Fa. Money Controls™ mit der Produktbezeichnung *Serial Compact Hopper MK2* verwendet. Diese Arbeit beschreibt den Entwurf und die Implementierung einer Testsoftware zur Ansteuerung eines Zusatzrestgeldspeichers mittels seriellem Kommunikationsprotokoll. Die Testsoftware wurde mittels UML modelliert und in Visual Basic implementiert. Die Kommunikation zwischen der Testsoftware und dem Zusatzrestgeldspeicher verläuft mittels des seriellen Protokolls *ccTalk*.

1.2 Ausgangssituation

1.2.1 Fahrausweisautomat

Die HOCHBAHN AG ist ein Hamburger Verkehrsunternehmen und das zweitgrößte Nahverkehrsunternehmen Deutschlands sowie der größte Partner im Hamburger Verkehrsverbund (HVV). Sie verfügt über eine Vielzahl unterschiedlicher Fahrausweisautomatentypen des Herstellers Höft & Wessel AG. Die Fahrausweisautomaten, im Folgenden Automat genannt, lassen sich zunächst in die Bereiche stationär und mobil unterteilen. Der stationäre Bereich entspricht U-Bahn- und Bushaltestellen. Hier ist die *almex.station* im Einsatz. Der mobile Bereich wird zunächst in zwei weitere Bereiche, Busse der HOCHBAHN und Fähren der HADAG[1], unterteilt. In den Bussen sind für den mobilen Ticketverkauf, direkt beim Fahrerarbeitsplatz, die *almex.optima* integriert. Auf den Fähren der HADAG sind die *almex.compact* im Einsatz, welche als eine kompaktere Lösung der *almex.station* angesehen werden können. Die *almex.station* sowie die *almex.compact* haben die gleiche Funktionsweise. Allerdings wurden aufgrund der geringeren Größe der *almex.compact* zum Teil andere Baugruppen eingesetzt als bei der *almex.station*.
Im nächsten Abschnitt wird auf die Funktionsweise, d. h. die Bargeldverarbeitung des Typs *almex.compact* eingegangen, welcher im Bild 1.1 zu sehen ist, da in diesem der *Serial Compact Hopper MK2* eingesetzt wird.

Bild 1.1: Automat *almex.compact* der Fa. Höft & Wessel AG. Entnommen aus [8]

[1] Vollständiger Firmenname: HADAG Seetouristik und Fährdienst AG. Die HADAG ist ein Fährbetreiber im Hamburger Hafen und ein Tochterunternehmen der HOCHBAHN AG.

1.2.2 Bargeldverarbeitung

Die Bargeldverarbeitung des Automaten steuert die Münz- und Notenannahme sowie die Ausgabe von Wechselgeld. Der innere Aufbau des Automatentyps *almex.compact* ist in Bild 1.2 schematisch dargestellt. Die Funktion der einzelnen Komponenten wird in Tabelle 1.1 beschrieben.

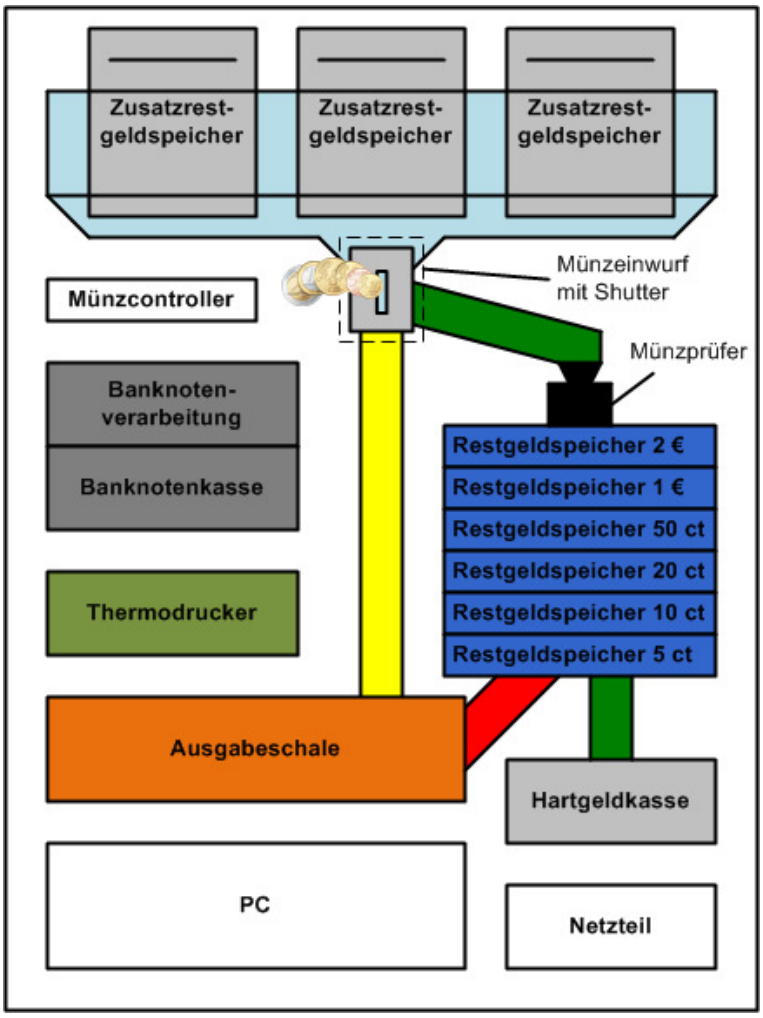

Bild 1.2: Schematische Darstellung des Automaten *almex.compact.* Angelehnt an [3]

Um ein Ticket an einem Automaten zu lösen, muss der Fahrgast zunächst im Verkaufsmenü das entsprechende Ticket auswählen. Der Automat wechselt dann aus dem Ruhezustand in den Verkaufsvorgang. Der Betrag für das Ticket kann u. a. durch das Einwerfen von Münzen bezahlt werden. Beim Wechsel in den Verkaufsvorgang öffnet sich der Münzeinwurf bzw. die Shutterklappe, welche im Bild 1.2 neben den Münzen abgebildet ist. Wird ein Objekt eingeworfen, wird zunächst mittels eines induktiven Sensors am Shutter geprüft, ob es sich beim eingeworfenen Objekt um einen metallischen Gegenstand handelt. Wenn dies der Fall ist, schließt die Shutterklappe und die eingeworfene Münze wird über

den Münzlauf (*grüner* Kanal) zum Münzprüfer weitergeleitet. Handelt es sich um einen nicht-metallischen Gegenstand, wird nach Ablauf einer kurzen Zeit der eingeworfene Gegenstand über den *gelb* eingezeichneten Kanal in die Ausgabeschale zurückgegeben. Der Münzprüfer prüft nun, ob es sich bei der eingeworfenen Münze um eine gültige oder ungültige Münze handelt. Sollte die Münze nicht erkannt werden (falsche oder gefälschte Münze), so wird diese über den *roten* Kanal an die Ausgabeschale zurückgegeben. Wird die Münze als gültig erkannt, so wird sie in dem dazugehörigen Restgeldspeicher (RGS) / Wechselgeldmagazin zwischengespeichert.
Sollte der Fahrpreis noch nicht erreicht sein, öffnet sich die Shutterklappe wieder und der beschriebene Vorgang wird wiederholt bis der Fahrpreis erreicht wurde. Das ausgewählte Ticket wird nach vollständiger Zahlung mittels Thermodrucker ausgedruckt. Zahlt der Fahrgast mehr als den fälligen Betrag, wird das Wechselgeld entweder aus den RGS oder aus den Zusatzrestgeldspeichern (ZRS), im Folgenden Hopper genannt, ausgegeben. Primär wird das Wechselgeld aus den RGS ausgegeben. Sollten diese nicht die benötigten Münzen ausgeben können, wird der noch fehlende Betrag aus den Hoppern ausgegeben. Kommt es zu einem Überlauf bzw. ist eine RGS gefüllt, so werden die zwischengespeicherten bzw. übergelaufenen Münzen über den *grünen* Kanal in die Hartgeldkasse entleert. Sollen Münzen wieder an den Fahrgast zurückgegeben werden, z. B. beim Abbruch des Verkaufsvorgangs, so werden die eingeworfenen Münzen wieder ausgegeben.

Ein wesentliches Element, welches nicht in der schematischen Darstellung zu sehen ist, ist das Display-Modul. Das Display-Modul dient zur Bedienung des Automaten und besteht u. a. aus einem 15“-XGA-LCD-Farbgrafikdisplay zur Anzeige des Verkaufsmenüs und einem 8-wire resistiven Touchsystem[2] zur Benutzereingabe [3].

In der nachfolgenden Tabelle 1.1 wird die Funktion der einzelnen Komponenten aus der schematischen Darstellung des Automaten präziser beschrieben.

[2] Bei einem resistiven Touchscreen wird der Ort des Berührungspunkts auf dem Touchscreen anhand der Widerstandsänderung zweier leitender Folienschichten ermittelt [16].

Tabelle 1.1: Beschreibung der Komponenten des Automaten *almex.compact*

Komponente	Beschreibung
Münzeinwurf mit Shutter	Der Shutter dient zur geregelten Annahme von Münzen, welche über einen Münzlauf zum Münzprüfer weitergeleitet werden. Der Shutter sorgt dafür, dass keine Fremdkörper in den Automaten gelangen. Fremdkörper werden noch vor dem Münzlauf abgewiesen.
Münzprüfer	Der elektronische Münzprüfer entscheidet aufgrund mehrerer Messparameter, ob es sich bei der eingeworfenen Münze um eine gültige oder ungültige Münze handelt. Mittels einer internen Weiche wird die Münze entweder in den Annahme- oder in den Abweiskanal geleitet. Zusätzlich wird die Information der Münzwertigkeit an den Münzcontroller weitergeleitet. Die Kommunikation zwischen Münzcontroller und Münzprüfer erfolgt über ein serielles Bussystem mit dem *ccTalk*-Protokoll.
Restgeld-speicher	Ein Restgeldspeicher (RGS) ist für die Aufnahme bzw. Zwischenspeicherung der eingeworfenen Münzen und für die Ausgabe von Wechselgeld zuständig. In einem Automaten befinden sich 6 RGS, in welchen die Münzsorten 5, 10, 20, 50 Cent, 1 und 2 Euro aufgenommen werden können. Die Münzsorten 1 und 2 Cent werden vom Automaten nicht angenommen. Jeder der 6 RGS nimmt also nur eine bestimmte Münzsorte auf und speichert sie zwischen. Die Steuerung z. B. für die Zuordnung einer eingeworfenen Münze zu dem jeweiligen RGS erfolgt über den Münzcontroller. Ein RGS kann maximal 50 Münzen aufnehmen. Kommt es zu einem Überlauf bzw. ist der RGS gefüllt, so werden die zwischengespeicherten bzw. übergelaufenen Münzen nach dem FIFO-Prinzip (First In First Out) in die Hartgeldkasse entleert. Sollen Münzen wieder an den Fahrgast zurückgegeben werden, z. B. beim Abbruch des Verkaufsvorgangs, so werden die eingeworfenen Münzen nach dem LIFO-Prinzip (Last In First Out) wieder ausgegeben [3]. Auf diese Weise wird auch vermieden, dass andere als die eingeworfenen Münzen beim Abbruchvorgang ausgegeben werden. Damit alle eingeworfenen Münzen wieder ausgegeben werden können, ist lediglich der Einwurf einer bestimmten Anzahl einer Münzsorte erlaubt. Technisch wäre es zwar möglich, 50 Münzen

	einer Münzsorte einzuwerfen, die höchstmögliche Anzahl wurde jedoch vom Automatenhersteller auf 30 begrenzt.
Hartgeldkasse (Endkasse)	In der Hartgeldkasse kommen die aus den RGS übergelaufenen Münzen an.
Zusatzrest-geldspeicher (Hopper)	Der Zusatzrestgeldspeicher (ZRS), im Folgenden Hopper genannt, dient ausschließlich zur Wechselgeldausgabe. In einem Automaten befinden sich 3 Hopper. Ein Hopper wird gefüllt eingesetzt und kommt erst zum Einsatz, wenn die benötigten Münzen nicht aus den RGS ausgegeben werden können. Ein Hopper besitzt eine bestimmte Anzahl an Münzen von jeder Münzsorte. Zzt. befinden sich in den Automaten nur Hopper mit den Münzsorten 5, 10 Cent und 2 Euro.
Münzcontroller	Der Münzcontroller steuert die gesamte Bargeldverarbeitung. Einige Aufgaben sind u. a.: Anhand der vom Münzprüfer erkannten Münzwertigkeit wird der dazugehörige RGS angesteuert. Durch die Datenerfassung der Füllstände der Baugruppen RGS und ZRS wird die Wechselgeld-ausgabe gesteuert.
Banknoten-verarbeitung	Die Banknotenverarbeitung (BNV) dient zum Transport und zur Erkennung der eingeführten Banknote. Eine BNV kann zzt. nur die Banknoten 5, 10, 20 und 50 Euro aufnehmen. Wenn die Banknote durch das Erkennungsmodul korrekt erkannt wird, wird diese nach Abschluss des Verkaufsvorgangs der Banknotenkasse hinzugefügt. Sollte die Banknote nicht erkannt werden, wird diese wieder an den Fahrgast ausgegeben. Da die BNV wegen ihrer kompakten Bauweise keine Zwischenkasse besitzt, kann nur eine Banknote pro Verkaufsvorgang eingeführt werden.
Banknotenkasse	Die Banknotenkasse (BNK) dient zur Speicherung der Banknoten, welche von der BNV hinzugefügt werden.
PC	Auf dem PC läuft die Automatensoftware bzw. die Verkaufs-applikation. Des Weiteren sind der PC und der Münzcontroller über das ARCNET[3] verbunden, damit u. a. zwischen den beiden der Datenaustausch der Füllstände der jeweiligen Baugruppen statt-finden kann.

[3] ARCNET steht für ***A**ttached **R**esources **C**omputer **Net**work* und ist eine Vernetzungstechnologie für lokale Netzwerke.

Bei den in der Tabelle beschriebenen Komponenten muss ergänzend erwähnt werden, dass sowohl RGS als auch ZRS vom Automatenhersteller eine ID-Nr. zugewiesen bekommen haben, damit der Automat erkennt, welcher RGS bzw. ZRS für eine Münzsorte zuständig ist. Die Relevanz der ID-Nr. wird im folgenden Abschnitt zur Aufgabenstellung erläutert.

1.3 Aufgabenstellung

Der Automatenhersteller Höft & Wessel AG hat den RGS und den ZRS eine ID-Nr. zugewiesen. Anhand dieser ID-Nr. bzw. anhand eines Teils der ID-Nr. kann der Münzcontroller des Automaten u. a. die Münzwertigkeit der jeweiligen Geldspeicher erkennen.

Verändert sich der Fahrkartenpreis muss evtl. das Verhalten der Wechselgeldausgabe angepasst werden. Da nur eine begrenzte Anzahl an ZRS im Automaten eingesetzt werden können (3 Stück wie oben beschrieben), muss das Unternehmen in der Lage sein, die vorhandenen ZRS flexibel zu nutzen, also z. B. einen 10 Cent ZRS durch einen 20 Cent ZRS zu ersetzen, wenn es durch eine Preisänderung notwendig wird. Hieran wird die Notwendigkeit deutlich, die ID-Nrn. der Geldspeicher verändern zu können.

Die im Lager befindlichen ZRS *Serial Compact Hopper MK2 (SCH2)*, welche als Ersatzteile vorhanden sind, haben vom Automatenhersteller keine ID-Nr. zugewiesen bekommen. Leider ist auch kein Gerät vorhanden, welches die ID-Nr. einspeichert. Im Falle eines Austauschs bzw. Einsatz eines Hoppers im Automaten würde dieser nicht erkannt werden.

Ziel der Abschlussarbeit ist es, eine Anwendung zu programmieren, welche die ID-Nr. der *SCH2* verändern kann. Zusätzlich sollen Möglichkeiten zur Instandsetzung der *SCH2* untersucht und diese in die Anwendung integriert werden. Vorgesehen ist ein Testprogramm, welches den Hopper ansteuert und bei Fehlverhalten eine entsprechende Fehlermeldung ausgibt.

2 Zusatzrestgeldspeicher *Serial Compact Hopper MK2*

In diesem Kapitel wird zunächst auf die allgemeine Beschreibung des Zusatzrestgeldspeichers *Serial Compact Hopper MK2 (SCH2)*, nachfolgend Hopper genannt, eingegangen. Des Weiteren folgt die funktionale Beschreibung, in der die einzelnen Bestandteile des Hoppers und die mechanische Funktionsweise zur Münzausgabe beschrieben werden. Im letzten Abschnitt sind die technischen Daten des Hoppers zu finden.

2.1 Allgemeines

Der Hopper, welcher im Bild 2.1 dargestellt wird, ist ein Auszahlungsgerät und hat im Automaten *almex.compact* die Funktion zur Wechselgeldausgabe. Der Hersteller des Hoppers ist die Fa. Money Controls™, welche sich seit 2010 der Gruppe Crane Payment Solutions angeschlossen hat [1]. Aus der Bezeichnung des Hoppers lässt sich schließen, dass dieser die seriell anzusteuernde Version der Produktreihe *Compact Hopper* ist [14]. Die serielle Schnittstelle verwendet das *ccTalk* (**c**oin **c**ontrols **Talk**) Protokoll, welches im Kapitel 3 beschrieben wird.

Bild 2.1: Zusatzrestgeldspeicher *Serial Compact Hopper MK2*

2.2 Funktionale Beschreibung

In diesem Abschnitt werden die Hauptbestandteile des Hoppers beschrieben.

Der Hopper besteht im Wesentlichen aus:

- Dem Gehäuse, welches gleichzeitig auch der Münzbehälter ist.
- Der Drehscheibe, mit der die Münzen zum Ausgang transportiert werden können.
- Der Lichtschranke, die sich direkt am Ausgang befindet, um eine ausgegebene Münze zu erfassen.
- Einem Sensor zur Erfassung des Füllstands des Hoppers.
- Der Steuerelektronik, mit der der Hopper angesteuert wird. Die Steuerelektronik führt die an der seriellen Schnittstelle des Hoppers empfangenen *ccTalk*-Befehle aus.

Im Folgenden wird die Funktion der Drehscheibe und des Sensors genauer beschrieben.

2.2.1 Drehscheibe

Die Funktion der Drehscheibe ist es, die im Hopper befindlichen Münzen zum Ausgang zu transportieren.
Es gibt zwei Drehscheibenarten, welche in Abhängigkeit der Münzgröße (Durchmesser und Dicke der Münze) eingesetzt werden. Für die größeren Münzen 50 Cent, 1 und 2 Euro wird eine blaue Scheibe (s. Bild 2.2) und für die kleineren Münzen 5, 10 und 20 Cent wird eine grüne Scheibe eingesetzt. Die blaue Scheibe besitzt vier und die grüne drei Löcher mit einem jeweiligen Durchmesser, der etwas mehr als der Münzgröße entspricht, für die der Hopper vorgesehen wurde.

2.2.2 Sensor zur Erfassung des Füllstands des Hoppers

Der Sensor besteht aus zwei rundförmigen metallischen Platten, welche getrennt voneinander auf derselben Höhe innerhalb des Münzbehälters angebracht sind. Ist der Hopper oberhalb der zwei metallischen Platten mit Münzen gefüllt, entsteht wegen der Leitfähigkeit der Münzen eine elektrische Verbindung. Dieser Zustand wird von der Steuerelektronik des Hoppers erkannt und als Information, dass der Hopper noch fast voll ist, in seinem Speicher hinterlegt. Befindet sich der Füllstand unterhalb der zwei Platten, ist die elektrische Verbindung unterbrochen und der Hopper hinterlegt im Speicher die Information, dass sein Füllstand fast leer ist. Der Hopper fragt den Status des Füllstands alle 2 Sekunden ab.

2.3 Mechanische Funktionsweise der Münzausgabe

Zunächst wird die Annahme getroffen, dass der Hopper einsatzbereit ist. Einsatzbereit ist der Hopper, wenn er gefüllt ist und ordnungsgemäß in Betrieb genommen wurde. Sobald der Hopper den Befehl empfängt, dass eine bestimmte Münzanzahl ausgegeben werden soll, rotiert die *blaue* Drehscheibe (s. Bild 2.2), damit die im Behältnis eingelagerten Münzen ausgegeben werden können. Im Bild 2.2 wurde zur besseren Darstellung der Münzbehälter entfernt.

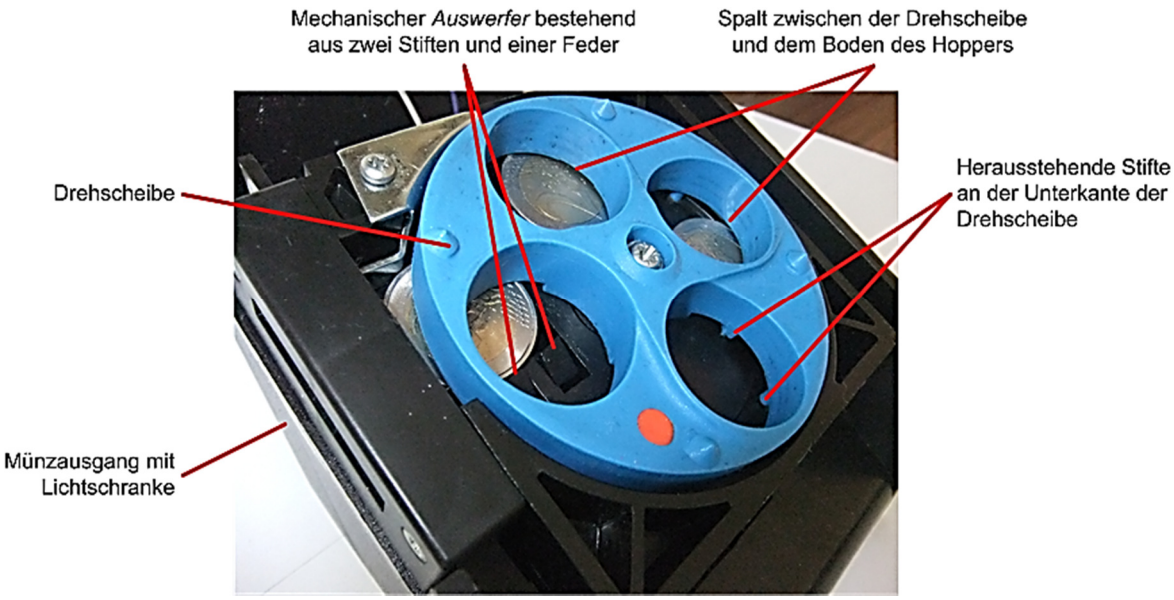

Bild 2.2: Beschreibung der Münzausgabe des *Serial Compact Hoppers MK2*

Unterhalb der blauen Drehscheibe, nachfolgend Scheibe genannt, bzw. zwischen der Unterkante der Scheibe und dem Boden des Behältnisses, befindet sich ein kleiner Spalt von ca. 3 mm. Die Spalthöhe ist so gewählt worden, dass zwei Münzen nicht übereinander, d. h. gleichzeitig, ausgegeben werden können. Bei der Rotation der Scheibe können die Münzen somit durch die Löcher der Scheibe zum Boden des Behältnisses gelangen. Unterhalb der Scheibe befinden sich Stifte, die dafür sorgen, dass die in die Löcher hineingefallenen Münzen bei der Rotation zum Ausgang bewegt werden. Diese Stifte haben zusätzlich eine Trennfunktion mit der verhindert werden kann, dass eine Münze zum benachbarten Loch gelangt.

Eine Münzausgabe kann nur stattfinden, wenn die Scheibe im Linkslauf bzw. gegen den Uhrzeigersinn gedreht wird. Vor dem Münzausgang befindet sich ein mechanischer *Auswerfer*. Der *Auswerfer* besteht aus zwei Stiften, welche aus dem Boden des Behältnisses leicht hervorstehen und einer Schraubenzugfeder, nachfolgend Feder genannt, an der die Stifte befestigt sind. Die Feder ist im Bild 2.2 nicht abgebildet, da sie

sich unterhalb des Bodens befindet. Die Scheibe drückt zunächst die Münze gegen die zwei Stifte. Da die zwei Stifte lose bzw. nur durch die Feder am Gehäuse des Hoppers befestigt sind, geben sie nach bzw. die Feder wird gedehnt. Spätestens beim Erreichen der Endposition der zwei Stifte wird die Münze mittels der ausgedehnten Feder aus dem Hopper ausgeworfen. Auf dem Bild 2.2 ist der Zustand zu sehen, in welchem die zwei Stifte ihre Endposition fast erreicht haben. Bei der nächsten Drehbewegung der Scheibe würde die zwischen dem Ausgang und den Stiften eingeklemmte Münze ausgeworfen werden.
Der Hopper besitzt zusätzlich eine Funktion zur Auflösung eines Münzstaus. Erkennt der Hopper einen Münzstau, löst er diesen durch kurzeitige Richtungsänderung, d. h. durch Drehung der Scheibe im Rechtslauf.

2.4 Technische Daten

Die serielle Ausführung des Hoppers zeichnet sich dadurch aus, dass sowohl mehrere Hopper als auch andere Geräte, welche mittels des *ccTalk*-Protokolls seriell angesteuert werden, an einem Bus angeschlossen werden können. Andere Geräte könnten bspw. Münzprüfer und Banknotenverarbeitung sein.

Der Hopper kann in zwei Modi betrieben werden. Zum einen im *single-mode*, in dem zwei Münzen pro Sekunde ausgegeben werden können und zum anderen im *multi-mode*, in dem acht bis zehn Münzen pro Sekunde ausgegeben werden können. Die Konfiguration des Modus für den der Hopper betrieben werden soll wird mittels eines *ccTalk*-Befehls ermöglicht. Ein auf Werkseinstellungen eingestellter Hopper ist zunächst im *multi-mode* konfiguriert.

Der Hopper kann so konfiguriert werden, dass Münzen mit einem Durchmesser von 16,25 bis 31 mm und mit einer Dicke von 1,25 bis 3,2 mm ausgegeben werden können. Diese Angaben erfüllen die Maße der Euromünzen[4]. Bei der Konfiguration handelt es sich nur um den Tausch der Drehscheibe, welche mit einer Schraube am Motor des Hoppers befestigt ist (s. Bild 2.2).

[4] Angaben der Europäischen Zentralbank (EZB): 1 Cent (16,25 mm, 1,67 mm); 2 Euro (25,75 mm, 2,20 mm); (Durchmesser, Dicke)

3 Serielles Kommunikationsprotokoll *ccTalk*

3.1 Einleitung

Das serielle Kommunikationsprotokoll *ccTalk* (**c**oin **c**ontrols **Talk**) wurde im Jahr 1996 von der Fa. Money Controls™, ehemals Coin Controls, eingeführt. Es wird heute weitgehend als industrieller Standard akzeptiert. *ccTalk* wurde entwickelt, um die Zusammenschaltung von mehreren Bargeldgeräten zu ermöglichen. Diese Zusammenschaltung wird anhand eines Bussystems, welches nur die drei Leitungen Versorgungsspannung, Datenleitung und Masse benötigt, ermöglicht [10].

Im Jahr 2010 wurde dem Protokoll noch nachträglich die Data Encryption Standard (DES) Verschlüsselung hinzugefügt, um Befehle wie z. B. die Münzausgabe des Hoppers, so zu sichern, dass diese noch robuster gegen Systemangriffe werden [1].

Der minimale Systemaufbau besteht mindestens aus einem Hostcontroller z. B. µC oder PC und einem Peripheriegerät z. B. Hopper. Der Aufbau des Protokolls ist an den RS-232-Standard angelehnt. Dabei ist zu beachten, dass *ccTalk* andere elektrische Eigenschaften aufweist. Es ist bidirektional und verwendet nur eine Datenleitung.

3.2 Aufbau des Datenpakets

Eine Kommunikation bzw. der Datenaustausch zwischen zwei Geräten hat zur Folge, dass immer zwei Nachrichten bzw. Datenpakete gesendet werden. Das erste Datenpaket wird immer vom Master zum Slave bzw. vom Hostcontroller/Rechner zum Hopper gesendet. Das zweite Datenpaket ist dann die Antwort des Hoppers, welches zum Hostcontroller gesendet wird [10].

Ein Datenpaket besteht aus der Zieladresse, der Anzahl der gesendeten Datenbytes, der Quelladresse, dem Header, evtl. aus Datenbytes und der Prüfsumme (s. Tabelle 3.1). Die Erläuterung der einzelnen Bestandteile eines Datenpakets erfolgt in Tabelle 3.3. Wenn keine Datenbytes gesendet wurden, steht im Feld Anzahl Datenbytes der Wert 0 (s. Tabelle 3.2). Die Tabelle 3.2 zeigt den minimalen Aufbau eines Datenpakets.

Tabelle 3.1: Darstellung eines Datenpakets mit Datenbytes

Ziel-adresse	Anzahl Datenbytes	Quell-adresse	Header	Datenbyte 1	…	Datenbyte N	Prüf-summe

Tabelle 3.2: Darstellung des minimalen Datenpakets

Ziel-adresse	0	Quell-adresse	Header	Prüf-summe

Die einzelnen Felder eines Datenpakets können nur die Werte 0 bis 255 annehmen. Wegen des Adressfeldes von einem Byte können theoretisch bis zu 256 Geräte (Adresse 0 bis 255) angeschlossen werden. Die Adresse 0 ist als Broadcastadresse und die Adresse 1 ist als Standardadresse für den Hostcontroller reserviert. Es bleiben also noch 254 Adressen übrig, welche für die Slave Geräte wie z. B. Hopper verwendet werden können.

Tabelle 3.3: Beschreibung der einzelnen Bestandteile eines Datenpakets

Bestandteil	Beschreibung
Zieladresse	Die Zieladresse beinhaltet entweder die Adresse des Hostcontrollers (Rechners) oder der Baugruppe (Hopper). Wird ein Befehl vom Rechner aus gesendet, steht in diesem Feld die Adresse des Hoppers z. B. der Wert 3. Bei der Antwort des Hoppers steht in diesem Feld die Adresse des Rechners, der standardmäßig den Wert 1 hat.
Anzahl der Datenbytes	In diesem Feld wird die Anzahl der zu übertragenen Datenbytes eingetragen. Bei einem einfachen Befehl wie z. B. *Simple Poll* müssen keine Datenbytes übertragen werden und deshalb wird in diesem Feld der Wert 0 eingetragen. Es existiert aber eine Reihe an Befehlen, die Datenbytes senden müssen. In diesem Fall steht hier die Anzahl der zu sendenden bzw. zu empfangenen Datenbytes.
Quelladresse	In diesem Feld steht die eigene Adresse. Beim Hopper handelt es sich um den Wert 3 und beim Rechner um den Wert 1.
Header	Dient zur Erkennung eines gesendeten Befehls oder der empfangenen Antwort. Wenn der Rechner den Befehl *Simple Poll* senden möchte, steht hier bspw. der Wert 254. Anhand dieser Zahl kann der Hopper erkennen, dass der Rechner diesen Befehl gesendet hat. Bei der Antwort des Hoppers steht für die Quittung des empfangenen Befehls der Wert 0.
Datenbytes	Zwischen dem Header und der Prüfsumme werden die zu sendenden Datenbytes, beginnend mit dem niedrigsten Byte, eingefügt.

Prüfsumme	Die Prüfsumme dient zur Erkennung von Übertragungsfehlern. Die Berechnung der Prüfsumme findet beim Sender und die Prüfung zur Erkennung eines Übertragungsfehlers findet beim Empfänger statt. Beim Sender werden alle zu sendenden Bytes, beginnend mit der Zieladresse bis hin zum letzten zu sendenden Datenbyte, aufaddiert. Ein entstehender Übertrag bei der Byte-Addition wird dabei nicht berücksichtigt bzw. verworfen. Die Prüfsumme ergibt sich aus dem Zweierkomplement[5] der aufaddierten Summe. Der Empfänger addiert alle empfangenen Bytes analog des Verfahrens beim Sender zzgl. der Prüfsumme. Die Summe muss den Wert 0 ergeben. Andernfalls liegt ein Übertragungsfehler vor. Die Berechnungsvorschrift für die Prüfsumme und die Prüfung auf Übertragungsfehler wird im nachfolgenden Beispiel beschreiben. Die Prüfsumme kann alle einfachen Bitfehler erkennen und fast alle doppelten Fehler. Die Wahrscheinlichkeit, dass ein doppelter Bitfehler nicht erkannt wird, liegt unterhalb von 1/n, mit n = Anzahl der zu übertragenen Bits.

Beispiel: Der Hostcontroller sendet den Befehl *Simple Poll* zum Hopper mit der Adresse 3. Der Hopper empfängt diesen Befehl und prüft zunächst, ob das Datenpaket korrekt bzw. ohne Übertragungsfehler angekommen ist. Anschließend sendet er die dazugehörige Antwort.

Der Hostcontroller sendet das folgende Datenpaket:

Tabelle 3.4: Hostcontroller sendet den Befehl *Simple Poll*

Ziel-adresse	Anzahl Datenbytes	Quell-adresse	Header	Prüf-summe
3	0	1	254	254

[5] Beim Zweierkomplement werden die Bits invertiert und anschließend wird ein Bit hinzuaddiert.

Der Hostcontroller hat die Prüfsumme für das zu sendende Datenpaket folgendermaßen berechnet:

Tabelle 3.5: Berechnung der Prüfsumme zum Befehl *Simple Poll*

Operation	Beschreibung
3 + 0 + 1 + 254 = 258	Alle zu übertragenen Bytes des Datenpakets werden, beginnend mit der Zieladresse bis hin zum letzten zu sendenden Datenbyte, aufaddiert. Da in diesem Beispiel keine Datenbytes gesendet werden müssen, wird einschließlich des Headers aufaddiert.
258 Modulo 256 = 2	Mit Hilfe der Modulo Division wird der Rest der soeben aufaddierten Summe 258 zum Wert 256 ermittelt.
256 - 2 = 254	Die Prüfsumme ist die Differenz des berechneten Rests (2) zum Wert 256. Die Prüfsumme hat somit den Wert 254.

Die Modulo Division und die danach berechnete Differenz ist eine softwareseitige Nachbildung des Zweierkomplements.

Nun empfängt der Hopper den soeben gesendeten Befehl des Hostcontrollers und prüft zunächst, ob dieser korrekt empfangen wurde.

Tabelle 3.6: Prüfung des empfangenen Befehls *Simple Poll* auf Übertragungsfehler

Operation	Beschreibung
(3 + 0 + 1 + 254 + 254) Modulo 256 = 0	Um einen Übertragungsfehler zu erkennen, werden zunächst alle Bytes des Datenpakets zzgl. der Prüfsumme aufaddiert. Das Ergebnis der Restdivision (Modulo) zwischen der aufaddierten Summe und dem Wert 256 muss den Wert 0 ergeben. Andernfalls liegt ein Übertragungsfehler vor.

Die Antwort bzw. Quittung des Hoppers auf den zuvor gesendeten Befehl bzw. das empfangene Datenpaket am Hostcontroller sieht folgendermaßen aus:

Tabelle 3.7: Antwort des Hoppers zum Befehl *Simple Poll*

Ziel-adresse	Anzahl Datenbytes	Quell-adresse	Header	Prüf-summe
1	0	3	0	252

3.3 Auflistung der wichtigsten Befehle

Tabelle 3.8: Auflistung der wichtigsten Befehle [11] [14]

Header	Name	Beschreibung
254	Simple Poll	Wird verwendet, um festzustellen, ob das Gerät (Hopper) am Bus angeschlossen ist und ordnungsgemäß arbeitet. Das Gerät antwortet mit einer ACK Meldung (Header = 0).
247	Request Variable Set	Abruf der eingestellten Parameterwerte sowie der gemessenen Strom- und Spannungswerte des Hoppers. Die Parameterwerte für den Hopper sind u. a. der Strombegrenzungswert für den im Hopper eingebauten Motor. Gemessene Strom- und Spannungswerte sind u. a. die Stromaufnahme des Motors und die Spannungsversorgung des Hoppers. Eine genauere Beschreibung befindet sich im Anhang (s. Bedienungsanleitung).
244	Request Product Code	Mit diesem Befehl kann festgestellt werden, ob es sich um eine verschlüsselte oder unverschlüsselte Version des Hoppers handelt. Bei der unverschlüsselten Version lautet die Antwort des Hoppers *SCH2-NOENCRYPT* und bei der verschlüsselten *SCH2*. Die Texte stehen in den Feldern der Datenbytes und werden in Form von ASCII Zeichen gesendet.
217	Request Payout High /Low Status	Gibt Auskunft über die vorhandenen Sensoren und somit auch über den Füllstand des Hoppers.
215	Read Data Block	Mit diesem Befehl können die im Non Volatile (NV)-Speicher bzw. die im EEPROM gespeicherten Werte des Hoppers gelesen werden. Der EEPROM des Hoppers besteht aus vier acht Byte langen Registern, welche als Blöcke bezeichnet werden. Diese entsprechen den Block-Nrn. 0 bis 3. Um aus dem entsprechenden Register auszulesen, wird der Wert der jeweiligen Block-Nr. als Datenbyte mitgesendet. Im Register mit der Block-Nr. 0 ist die ID-Nr. des Hoppers vom Automaten-hersteller gespeichert.
214	Write Data Block	Mit diesem Befehl können die Registerinhalte des NV-Speichers des Hoppers geändert werden. Mit dem Befehl *Write Data Block* und der Angabe der Block-Nr. 0 ist es möglich, die vom Automatenhersteller vergebene ID-Nr. zu ändern.

172	Emergency Stop	Dient zur sofortigen Abschaltung des Motors bzw. der Münzausgabe. Dieser Befehl hat denselben Effekt wie ein Software-Reset, d. h. der Hopper muss mit dem Befehl *Enable Hopper* wieder aktiviert werden, bevor eine erneute Münzausgabe erfolgen kann.
167	Dispense Hopper Coins	Dient zur Aktivierung der Münzausgabe. Zuvor müssen mittels des Befehls *Request Cipher Key* die verschlüsselten Bytes vom Hopper abgefragt werden. In Abhängigkeit der Version des Hoppers (verschlüsselt oder unverschlüsselt), welche mit dem Befehl *Request Product Code* abgefragt werden kann, wird entschieden, ob eine Verschlüsselung erfolgen muss oder nicht. Der Befehl benötigt insgesamt neun Datenbytes. Acht Datenbytes für die verschlüsselten Bytes und ein Datenbyte für die Münzanzahl, die ausgegeben werden soll. Bei der verschlüsselten Version müssen die acht Datenbytes zuvor noch berechnet werden (s. Abschnitt 3.4). Bei der unverschlüsselten Version können die acht Datenbytes einen beliebigen Wert erhalten.
166	Request Hopper Status	Wird verwendet, um den Status der Münzausgabe abzufragen. Die Antwort des Hoppers besteht u. a. aus der Anzahl der ausgegebenen und noch auszugebenen Münzen.
164	Enable Hopper	Bevor eine Münzausgabe erfolgen kann, muss der Hopper zunächst aktiviert werden. Als Aktivierungscode dient das Datenbyte mit dem Wert 165. Jeder andere Wert deaktiviert den Hopper.
163	Test Hopper	Wird verwendet, um die Status-/Fehlermeldungen des Hoppers anzuzeigen. Der Hopper sendet zwei Datenbytes, in dem der Zustand jedes Bits Auskunft über eine entsprechende Meldung gibt. Es gibt insgesamt 16 verschiedene Meldungen, welche in der Bedienungsanleitung beschrieben sind (s. Anhang).
160	Request Cipher Key	Mit diesem Befehl werden die verschlüsselten Bytes (*Cipher Key*) vom Hopper abgefragt. Der Befehl muss unabhängig von der Version des Hoppers (verschlüsselt oder unverschlüsselt) vor dem Befehl *Dispense Hopper Coins* gesendet werden.

001	Reset Device	Mit diesem Befehl wird ein Software-Reset ausgeführt. Der Hopper antwortet nach dem Erhalt des Befehls mit einer ACK-Meldung (Header = 0). Nach einem Reset werden alle zuvor gesetzten Status-/Fehlermeldungen des Hoppers, sofern diese nicht noch vorhanden sind, gelöscht.

3.4 Funktionsweise der Münzausgabe

Die Befehlskette bzw. die Befehlsausführung zur Münzausgabe erfolgt unabhängig von der Version (verschlüsselt oder unverschlüsselt) des Hoppers. Der einzige Unterschied besteht darin, dass bei der verschlüsselten Version der Hostcontroller eine DES-Entschlüsselung und -Verschlüsselung durchführen muss. Die Befehlskette zur Münzausgabe ist in der nachfolgenden Tabelle 3.9 dargestellt.

Tabelle 3.9: Befehlsausführung zur Münzausgabe

Nr.	Header		Befehlsbezeichnung	Kurzbeschreibung
	DEC	HEX		
1	001	01	Reset Device	Status-/Fehlermeldungen löschen.
2	164	A4	Enable Hopper	Münzausgabe aktivieren.
3	244	F4	Request Product Code	Version des Hoppers abfragen.
4	111	6F	Request Encryption Support	Ggf. DES-Schlüssel abfragen. Nur bei der verschlüsselten Version.
5	160	A0	Request Cipher Key	Verschlüsselte Bytes abfragen.
6	-	-	-	Ggf. DES Ent- und Verschlüsselung im Hostcontroller durchführen.
7	167	A7	Dispense Hopper Coins	Befehl zur Münzausgabe senden.
8	166	A6	Request Hopper Status	Ggf. Hopper Status abfragen.
9	163	A3	Test Hopper	Ggf. auf Fehler prüfen.

Jeder Hopper besitzt einen einmaligen DES-Schlüssel. Hierdurch wird die Möglichkeit, den Schlüssel des Hoppers durch einen Brute-Force-Angriff[6] zu knacken auf nur einen Hopper begrenzt, da der ermittelte Schlüssel nur für einen Hopper gültig ist. Da heutzutage ein DES-Schlüssel in weniger als einem Tag geknackt werden kann, wurde ein Befehl entwickelt, mit dem der Schlüssel beliebig gewechselt werden kann. Die Empfehlung des Herstellers lautet, den Schlüssel, aufgrund der Lebensdauer des internen Speichers,

[6] Brute-Force ist ein Angriff, bei dem alle möglichen Schlüssel systematisch ausprobiert werden [2].

frühestens alle 8 Stunden zu wechseln [13]. Sollte der Schlüssel geknackt werden, ist dieser nach dem Schlüsselwechsel ungültig. Im Folgenden wird die Funktionsweise der DES-Verschlüsselung im Hostcontroller und im Hopper erläutert.

3.5 Funktionsweise der DES-Verschlüsselung

3.5.1 DES-Verschlüsselung im Hostcontroller

Die Funktionsweise der DES-Verschlüsselung im Hostcontroller wird im Bild 3.1 dargestellt.

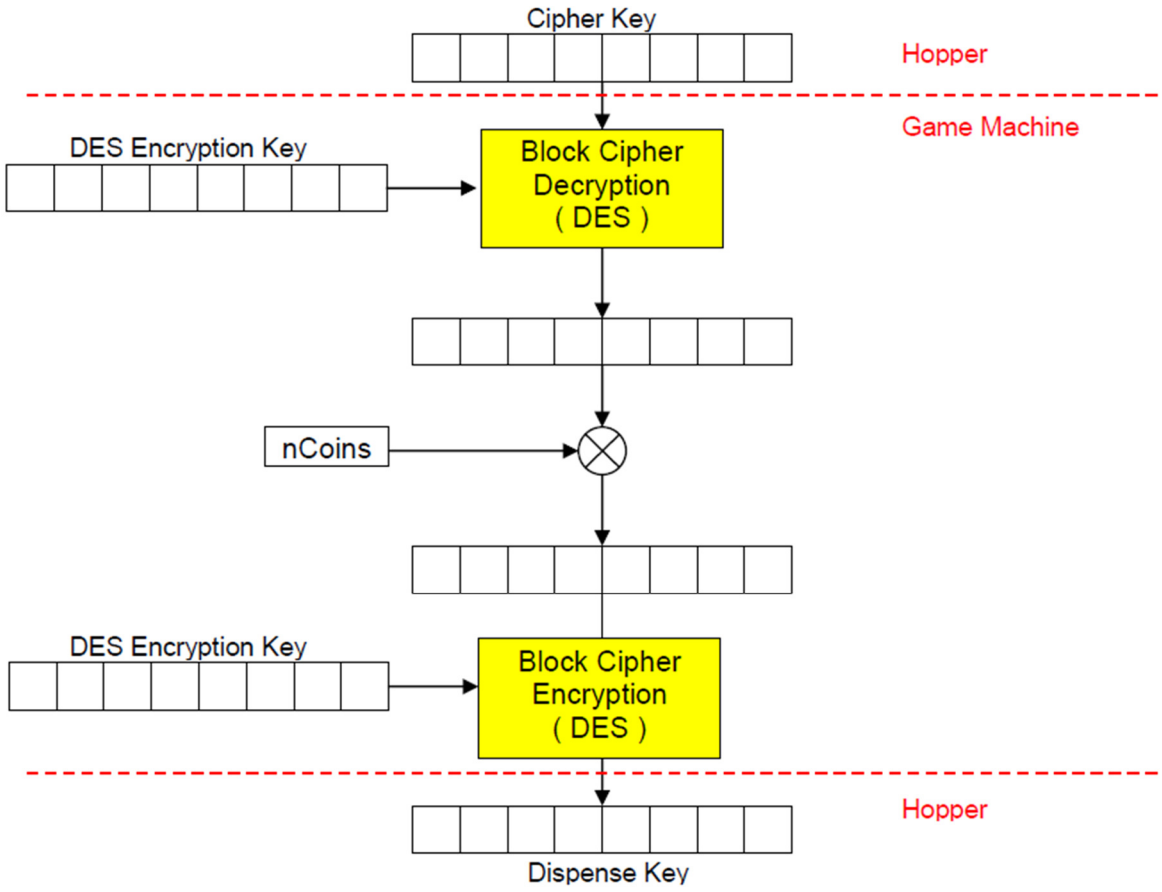

Bild 3.1: Beschreibung der DES-Verschlüsselung im Hostcontroller. Entnommen aus [13]

Der Befehl zur Münzausgabe (*Dispense Hopper Coins*) benötigt acht Datenbytes (*Dispense Key*), welche den verschlüsselten Bytes des Hostcontrollers (*Game Machine*) entsprechen. Damit die Verschlüsselung der Bytes erfolgen kann, ist es zunächst erforderlich, den Befehl *Request Encryption Support* an den Hopper zu senden, um den DES-Schlüssel (*DES Encryption Key*) des Hoppers zu erhalten.

Nun können mittels des Befehls *Request Cipher Key* die verschlüsselten Bytes (*Cipher Key*) des Hoppers abgefragt werden. Diese werden mittels des DES-Schlüssels im Hostcontroller entschlüsselt. Anschließend findet eine byteweise XOR-Verknüpfung zwischen den entschlüsselten Bytes und der Münzanzahl (*nCoins*), die ausgegeben werden soll, statt. Als letztes werden die Bytes aus der XOR-Verknüpfung mit dem DES-Schlüssel verschlüsselt und mittels des Befehls *Dispense Hopper Coins* an den Hopper gesendet. Der Hopper prüft den empfangenen Befehl und beginnt mit der Münzausgabe, sofern die Datenbytes korrekt sind. Die Überprüfung des Hoppers wird nachfolgend beschrieben.

3.5.2 DES-Verschlüsselung im Hopper

Die Funktionsweise der DES-Verschlüsselung im Hopper wird im Bild 3.2 dargestellt.

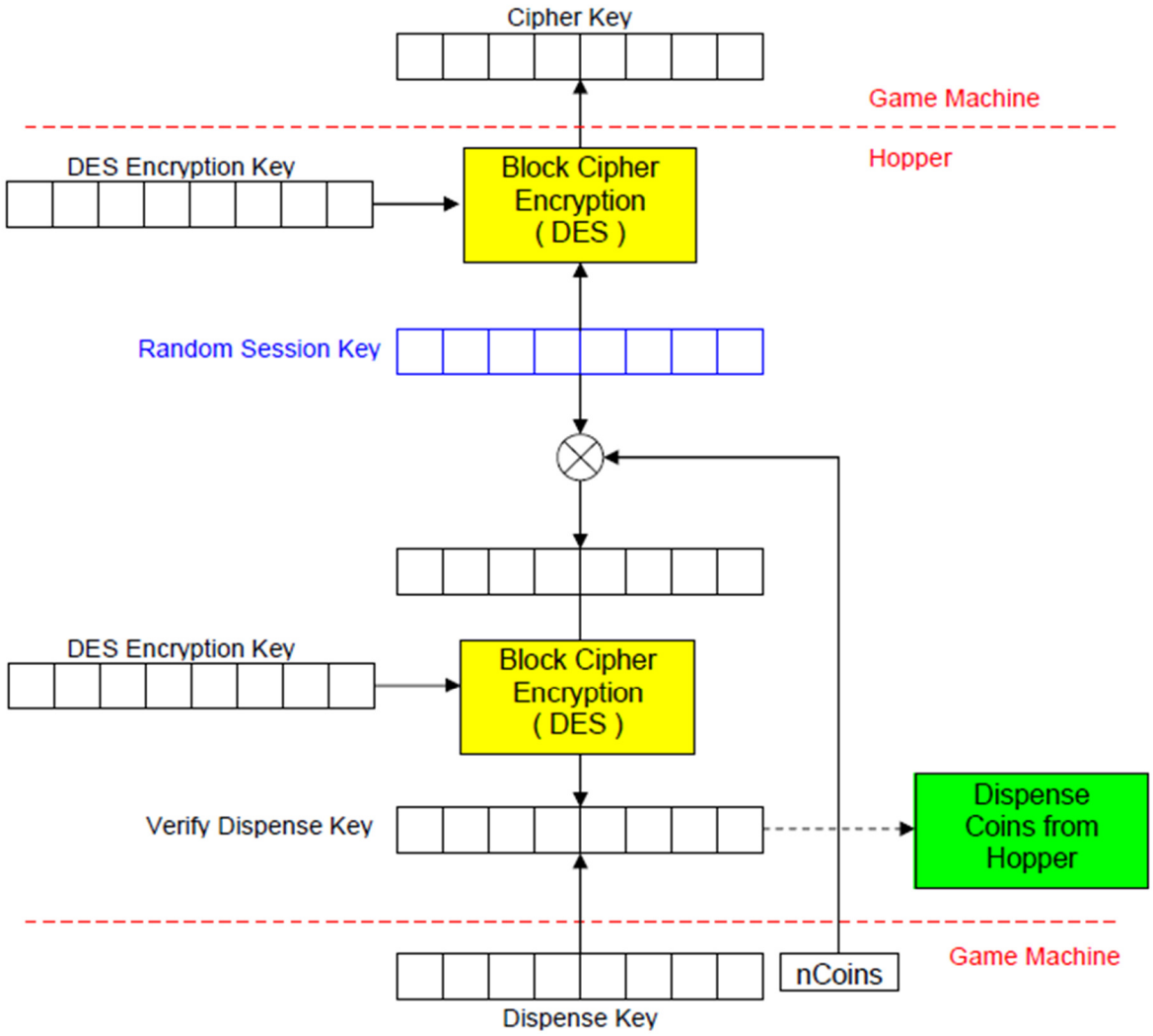

Bild 3.2: Beschreibung der DES-Verschlüsselung im Hopper. Entnommen aus [13]

Die verschlüsselten Bytes (*Cipher Key*) des Hoppers werden berechnet, in dem der Hopper mittels eines Zufallszahlengenerators acht Bytes (*Random Session Key*) generiert und diese mit dem DES-Schlüssel (*DES Encryption Key*) verschlüsselt.
Wenn der Hostcontroller (*Game Machine*) den Befehl *Dispense Hopper Coins* sendet, prüft der Hopper die empfangenen Datenbytes (*Dispense Key*). Anstatt die verschlüsselten Datenbytes zu entschlüsseln wird wie folgt vorgegangen. Der Hopper führt eine Verschlüsselung seiner eigenen generierten Bytes (*Random Session Key*) durch. Dabei findet zunächst, wie auch im Hostcontroller, eine byteweise XOR-Verknüpfung zwischen den generierten Bytes und der Münzanzahl (*nCoins*), die ausgegeben werden soll, statt. Die Münzanzahl ist das neunte Datenbyte des empfangenen Datenpakets.
Anschließend werden die Bytes aus der XOR-Verknüpfung mit dem DES-Schlüssel verschlüsselt. Danach werden die beiden verschlüsselten Byteblöcke (*Verify Dispense Key* und *Dispense Key*) miteinander verglichen. Stimmen diese überein, wird die Münzausgabe gestartet und der Hostcontroller erhält die Antwort des Hoppers, dass dieser den Befehl erhalten hat. Sollten die Byteblöcke nicht übereinstimmen, erhält der Hostcontroller die Antwort, dass die Münzausgabe nicht gestartet werden konnte.

Es ist zu erkennen, dass der Hopper keine Entschlüsselung der verschlüsselten Datenbytes des empfangenen Datenpakets, sondern eine Verschlüsselung der eigenen generierten Bytes und einen einfachen Vergleich der verschlüsselten Byteblöcke durchführt. Der Grund für diese Realisierung ist, dass auf diese Weise eine Steigerung der Performance des Hoppers erfolgt. Eine Entschlüsselung der Daten würde länger dauern und einen wesentlichen höheren Hardware- sowie Softwareaufwand bedeuten.

Der Hopper generiert aus Sicherheitsgründen nach jedem *Dispense Hopper Coins* Befehl neue Zufallszahlen (*Random Session Key*) und somit auch neue verschlüsselte Bytes (*Cipher Key*). Einfache Wiederholungsangriffe werden auf diese Weise verhindert [13].

3.5.3 Beispiel zur Funktionsweise der Verschlüsselung

Das Beispiel zur Funktionsweise der Verschlüsselung ist angelehnt an [13].

Ausgangssituation: Der DES-Schlüssel des Hoppers lautet: $1032547698BADCFE_{16}$ und der Hopper soll fünf Münzen ausgeben.

Mit den nachfolgenden Bezeichnungen **TxD** und **RxD** sind die jeweils gesendeten Datenpakete vom Hostcontroller zum Hopper (*TxD*) und vom Hopper zum Hostcontroller (*RxD*) gemeint. Die gesendeten Bytes sind wegen der übersichtlicheren und kompakteren Darstellung in Hexadezimal angegeben.

1. Status-/Fehlermeldungen löschen bzw. Software-Reset durchführen (*Reset Device*).

TxD:	03	00	01	01	FB
RxD:	01	00	03	00	FC

2. Münzausgabe aktivieren (*Enable Hopper*).

TxD:	03	01	01	A4	A5	B2
RxD:	01	00	03	00	FC	

3. Version des Hoppers abfragen (*Request Product Code*).

TxD:	03	00	01	F4	08				
RxD:	01	04	03	00	53	43	48	32	48

4. DES-Schlüssel abfragen (*Request Encryption Support*).

TxD:	03	06	01	6F	AA	55	00	00	55	AA	89	
RxD:	01	17	03	00	01	65	18	40	40	FF	21	43	65
	10	32	54	76	98	BA	DC	FE	ED				

Der DES-Schlüssel des Hoppers lautet also: $1032547698BADCFE_{16}$

5. Die verschlüsselten Bytes abfragen (*Request Cipher Key*).

TxD:	03	00	01	A0	5C							
RxD:	01	08	03	00	B6	CE	58	C5	0B	77	CD	64	A0

6. DES-Entschlüsselung und -Verschlüsselung im Hostcontroller durchführen:
   1. Entschlüsselung der empfangenen Datenbytes (verschlüsselte Bytes) des Befehls *Request Cipher Key.*

Cipher Key:	B6	CE	58	C5	0B	77	CD	64
DES Key:	10	32	54	76	98	BA	DC	FE
Decrypted:	01	02	03	04	05	06	07	08

2. Durchführung einer byteweisen XOR-Verknüpfung zwischen den entschlüsselten Bytes (Decrypted) und der angegebenen Münzanzahl (Number of Coins), die ausgegeben werden soll.

Decrypted:	01	02	03	04	05	06	07	08
Number of Coins:	05	05	05	05	05	05	05	05
XOR Result:	04	07	06	01	00	03	02	0D

3. Die Bytes aus der XOR-Verknüpfung mit dem DES-Schlüssel verschlüsseln.

XOR Result:	04	07	06	01	00	03	02	0D
DES Key:	10	32	54	76	98	BA	DC	FE
Encrypted:	04	DD	E7	A2	97	90	31	7E

7. Befehl zur Münzausgabe senden (*Dispense Hopper Coins*).

TxD:	03	09	01	A7	04	DD	E7	A2	97	90	31	7E	05	07
RxD:	01	01	03	00	01	FA								

Der Hopper prüft den empfangenen Befehl *Dispense Hopper Coins*, sendet an den Hostcontroller die Bestätigung zum Erhalt des Befehls und gibt die fünf Münzen aus.

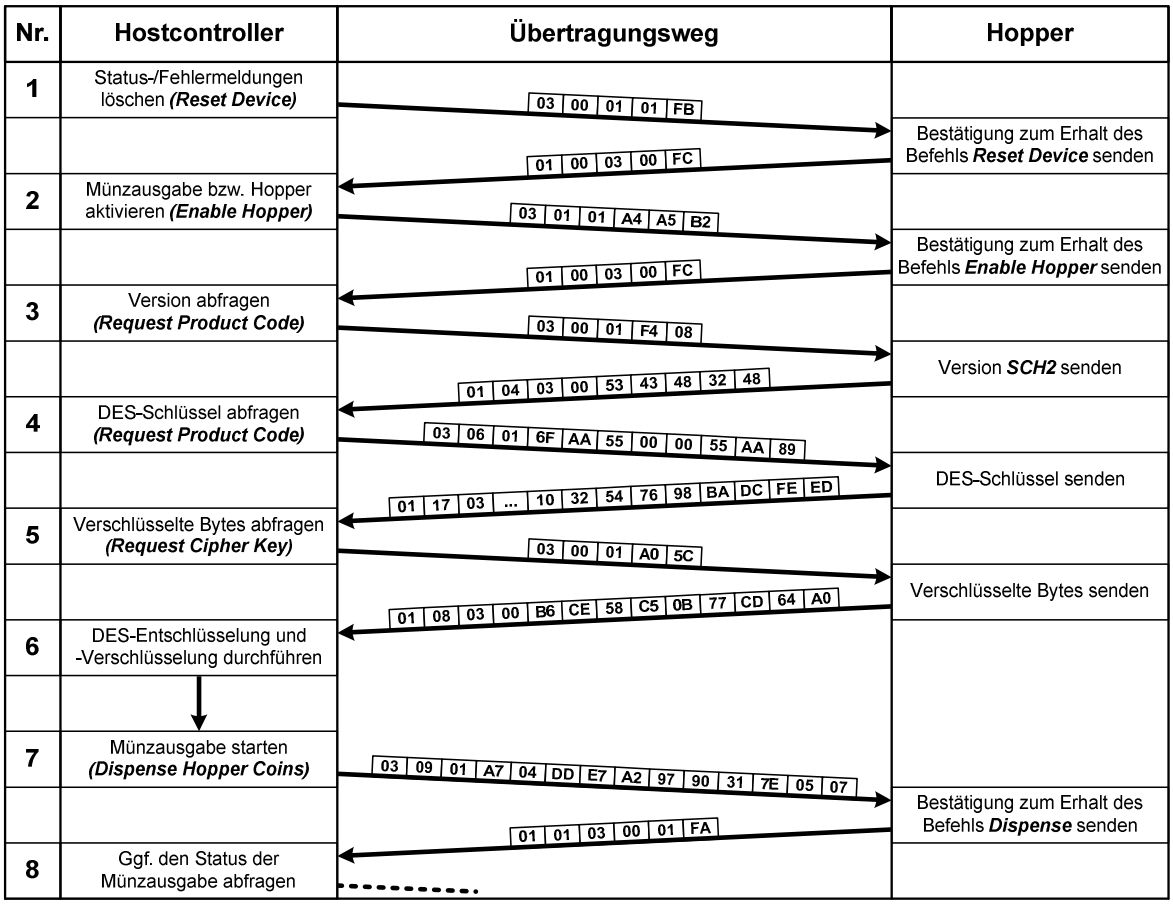

Bild 3.3: Grafische Darstellung der Funktionsweise der Verschlüsselung

Die im Bild 3.3 dargestellten Datenpakete entsprechen deren Aufbau und nicht der Reihenfolge, in der sie gesendet werden. Dies betrifft hier nur die gesendeten Datenpakete vom Hostcontroller zum Hopper. Z. B. beim ersten Datenpaket ist das erste gesendete Byte [03] und nicht [FB]. Bei den Antworten des Hoppers hingegen entspricht die Darstellung auch der Reihenfolge, in der sie gesendet werden. Die Datenpakete werden also von links nach rechts gelesen.

4 Hardware

Die Kommunikation zwischen der Testsoftware und dem Hopper verläuft mittels des seriellen Protokolls *ccTalk*, welches im vorherigen Kapitel vorgestellt wurde. Es wurde festgestellt, dass das *ccTalk*-Protokoll eine modifizierte Form der seriellen RS-232-Schnittstelle ist, welche andere elektrische Eigenschaften besitzt. Um eine Verbindung zwischen der Testsoftware und dem Hopper herzustellen, war es erforderlich einen RS232-zu-*ccTalk*-Adapter herzustellen, welcher u. a. die elektrischen Eigenschaften des *ccTalk*-Protokolls berücksichtigt.

4.1 RS-232-Schnittstelle

Die RS-232-Schnittstelle ist eine serielle Spannungsschnittstelle, welche zur asynchronen Datenübertragung, kurz UART (Universal Asynchronous Receiver/Transmitter) genannt, verwendet wird [17]. Die RS-232-Schnittstelle wurde von der EIA (Electronic Industries Association) standardisiert. Die Datenübertragung erfolgt bitseriell mit je einer Datenleitung für beide Übertragungsrichtungen. Da es sich um eine asynchrone Übertragung handelt, wird die Information, dass Daten gesendet werden sollen, mittels eines Startbits gekennzeichnet. Die logischen bzw. binären Zustände der Daten werden anhand von bipolaren Spannungspegeln festgestellt. Die Pegeldefinition ist in Tabelle 4.1 dargestellt.

Tabelle 4.1: Pegeldefinition der RS-232-Schnittstelle nach [17]

Logischer Wert	Min. Pegelwert	Max. Pegelwert	Typischer Wert
Low „0“	+3 V	+15 V	+12 V
High „1“	-3 V	- 15 V	-12 V

Aus der Tabelle 4.1 kann entnommen werden, dass für die logische „0“ ein minimaler Pegel von +3 V und ein maximaler Pegel von +15 V erlaubt sind. Für die logische „1“ hingegen ist ein minimaler Pegel von -3 V und ein maximaler Pegel von -15 V erlaubt. Die minimalen Pegelwerte sind diejenigen, die beim Empfänger mindestens anliegen müssen, damit dieser den logischen Wert sicher erkennen kann. Der Spannungsbereich zwischen ±3 V gilt als undefiniert. Die typischen Pegelwerte ±12 V sind auf die verfügbare Spannungsversorgung (Netzteil) eines Rechners zurückzuführen.

Die RS-232-Schnittstelle eines Rechners ist in Form eines Steckers vorzufinden. Es folgt zunächst die Anschlussbelegung des 9poligen RS-232-Steckers.

Tabelle 4.2: Anschlussbelegung des 9poligen RS-232-Steckers nach [4]

Pin	Ein-/Ausgang	Bezeichnung	Funktion
-	-	Shield	Schirmleitung
1	Ein	DCD (Data Carrier Detect)	Empfangssignalpegel
2	Ein	RxD (Receive Data)	Empfangsdaten
3	Aus	TxD (Transmit Data)	Sendedaten
4	Aus	DTR (Data Terminal Ready)	Endgerät bereit
5	-	GND (Ground)	Betriebserde
6	Ein	DSR (Data Set Ready)	Betriebsbereitschaft
7	Aus	RTS (Request To Send)	Sendeteil einschalten
8	Ein	CTS (Clear To Send)	Sendebereitschaft
9	Ein	RI (Ring Indicator)	Ankommender Ruf

Für einen bidirektionalen Datenaustausch werden mindestens drei Leitungen benötigt. Die Sendeleitung (TxD, *Pin 3*), die Empfangsleitung (RxD, *Pin 2*) und die gemeinsame Bezugsleitung (GND, *Pin 5*). Die restlichen Leitungen dienen zum Aufbau und zur Steuerung der Datenübertragung. Alle Ein- und Ausgänge sind kurzschlussfest und die Ausgänge können einen Strom bis zu ca. 20 mA liefern [4].

Heute ist die RS-232 weitgehend durch USB (Universal Serial Bus) verdrängt worden. Beim USB lässt sich anhand der Bezeichnung feststellen, dass es sich um einen Bus handelt. Alle am Bus angeschlossenen Teilnehmer erhalten die gleichen Daten und können anhand der Adressierung erkennen, ob diese für sie bestimmt sind. Bei der RS-232-Schnittstelle hingegen handelt es sich nur um eine direkte Verbindung zwischen zwei Geräten. Es existieren noch viele Anwendungsgebiete, in der die RS-232 verwendet wird. Vor allem im Bereich der Mess-, Steuer- und Regelungstechnik wird die RS-232 für kleinere Anwendungen, wegen ihrer Einfachheit, gegenüber der USB-Schnittstelle bevorzugt [15]. Sollte an einem Rechner keine serielle RS-232-Schnittstelle vorhanden sein, können USB-zu-Seriell-Adapter verwendet werden. Dem Rechner wird hierdurch ein virtueller COM-Port hinzugefügt, welcher wie eine gewöhnliche RS-232-Schnittstelle angewendet werden kann.

4.2 Grundschaltung des RS232-zu-*ccTalk*-Adapters

Die Grundidee der erstellten Schaltung wurde aus der *ccTalk*-Spezifikation entnommen. Im nachfolgenden Bild 4.1 wird die Grundschaltung dargestellt, welche als PC-Schnittstelle bezeichnet wird.

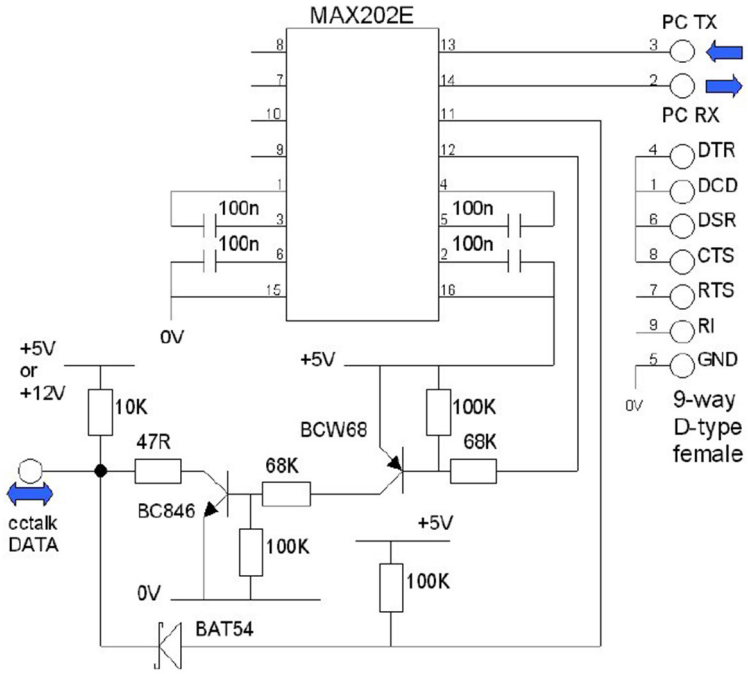

Bild 4.1: Grundschaltung des RS-232-zu-*ccTalk*-Adapters. Entnommen aus [12]

Die Schaltung in Bild 4.1 zeigt, wie die 9polige serielle RS-232-Schnittstelle am *ccTalk*-Bus anzuschließen ist. Der Treiberbaustein *MAX202E* des Herstellers Maxim hat die Aufgabe, die Spannungspegel des Rechners von +12 V und -12 V in 0 V und +5 V und die am *ccTalk*-Bus anliegenden Spannungen von 0 V und +5 V in +12 V und -12 V umzusetzen. Dadurch, dass der Treiber selbst nur eine Spannungsversorgung von +5 V hat, werden für die Spannungserzeugung der ±12 V Ladungspumpen in Form von Kondensatoren benötigt. Diese sind in Bild 4.1 links und rechts neben dem Treiber eingezeichnet. Die Beschaltung unterhalb des Treibers ist für die Aufbereitung der Daten auf die bidirektionale Datenleitung des *ccTalk*-Busses zuständig.

Die Sendedaten des Rechners (TxD, *Pin 3*) werden dem Eingang (*Pin 13*) des Treibers zugeführt. Der Treiber führt die Pegelumsetzung durch und gibt den Pegel am *Pin 12* aus. Wenn keine Daten gesendet werden, liegt am *Pin 3* des COM-Ports der Ruhepegel von -12 V an. Somit liegen am *Pin 12* +5 V an. *Pin 12* ist an die Basis des PNP-Transistors *BCW68* angeschlossen und somit ist dieser im Ruhestand gesperrt. Dadurch, dass der PNP-Transistor gesperrt ist, ist der NPN-Transistor *BC846* ebenfalls gesperrt, da an dessen Basis das Massepotential (0 V) anliegt. Ist der NPN-Transistor gesperrt, liegen auf dem

ccTalk-Bus +5 V an. Somit wurde festgestellt, dass am *ccTalk*-Bus ein Ruhepegel von +5 V anliegt.
Wenn der Rechner Daten senden möchte, werden diese mit dem Startbit initiiert. Das Startbit ist dadurch gekennzeichnet, dass es einen Low-Pegel hat. Am *Pin 3* des COM-Ports wechselt somit der Pegel von -12 V auf +12 V und am *Pin 12* liegen 0 V an. Dadurch wird der PNP-Transistor durchgeschaltet und an der Basis des NPN-Transistors liegen 5 V. Der NPN-Transistor schaltet somit ebenfalls durch und der *ccTalk*-Bus wird auf Masse gezogen. Es liegen somit 0 V am *ccTalk*-Bus an.
Über den *Pin 11* des Treibers werden die auf dem *ccTalk*-Bus anliegenden Daten dem Treiber zugeführt. Der Treiber setzt den entsprechenden Pegel um und gibt diesen am *Pin 14* aus. Der *Pin 14* des Treibers wird mit dem *Pin 2* des Rechners angeschlossen, damit der Rechner die Empfangsdaten erhält. Die Diode *BAT54* vor dem Eingang des Treibers wird als Verpolungsschutz verwendet, um den Eingang zu schützen.
Der NPN-Transistor *BC846* und der PNP-Transistor *BCW68* werden als elektronische Schalter verwendet und können durch beliebige andere Transistoren mit den gleichen Eigenschaften ersetzt werden.
Aus der Schaltung lässt sich zusätzlich erkennen, dass der *ccTalk*-Bus auch mit +12 V betrieben werden kann. Der Hopper unterstützt nur die Variante mit +5 V.

4.3 Erweiterung der Grundschaltung

Die im vorherigen Abschnitt beschriebene Grundschaltung für die PC-Schnittstelle wurde um einige Funktionen erweitert. Der Schaltplan, die Stückliste und das Leiterplattenlayout befinden sich im Anhang.

Damit keine externe Spannungsversorgung am RS232-zu-*ccTalk*-Adapter angeschlossen werden muss, mit der der Treiber versorgt wird, wurde die nachfolgend vorgestellte Lösung erarbeitet.
Die Lösung besteht darin, den Treiber vom Rechner aus zu versorgen. Die RS-232-Schnittstelle verfügt über 9 Pinanschlüsse (s. Tabelle 4.2). TxD (*Pin 3*) und RxD (*Pin 2*) sowie die Bezugsmasse (*Pin 5*) werden bereits für die Kommunikation verwendet. Es stehen somit noch zwei Ausgangspins zur Verfügung (*Pin 4* und *Pin 7*), welche als Spannungsquellen für den Treiber verwendet werden können, sofern diese den benötigten Strom liefern können. Als Bezugsmasse dient ebenfalls *Pin 5*. Aus dem Datenblatt bzw. den elektrischen Eigenschaften des *MAX202E* wurde entnommen, dass der Treiber eine typische Stromversorgung von 8 mA benötigt. Der maximale Wert liegt bei 15 mA [6].

Da die Ausgänge der RS-232 einen Strom bis zu 20 mA liefern können, ist diese Bedingung erfüllt. Für die Spannungsversorgung des Treibers *MAX202E* wurde der Ausgang *Pin 4* der RS-232-Schnittstelle verwendet.

Der Ausgang *Pin 4* ist zunächst inaktiv und hat somit den typischen Pegelwert von -12 V. Für den Treiberbaustein wird jedoch eine Versorgungsspannung von +5 V ±10 % benötigt [6]. In der Testsoftware wird beim Öffnen einer seriellen Schnittstelle (COM-Port) der Ausgang zunächst in den aktiven Zustand versetzt, damit am *Pin 4* der Pegel +12 V anliegt. Diese Spannung wird auf den Eingang eines Spannungsreglers gegeben. Am Ausgang des Spannungsreglers stehen dann +5 V zur Verfügung, welche dem Treiber zugeführt werden. Weil der Adapter auch an einem geschlossenen COM-Port angeschlossen sein könnte, wurde eine Diode eingebaut, welche als Verpolungsschutz dient. Liegt eine negative Spannung am *Pin 4* an, sperrt die Diode und der Verbraucher (Spannungsregler) erhält somit keinen Strom.
Eine alternative Lösungsvariante wäre es, eine Spannungsbegrenzung/-stabilisierung mittels einer Z-Diode zu verwenden. Dabei würde eine 5,1 V Z-Diode in Sperrrichtung mit einem Vorwiderstand anstelle des Spannungsreglers eingesetzt werden.

Sollte die Stromversorgung nicht ausreichend sein, gibt es folgende zwei Lösungsvarianten. Zum einen könnte ein weiterer Ausgang z. B. (*Pin 7*) als Spannungsquelle parallel dazu angeschlossen werden. Zum anderen könnte der Treiberbaustein *MAX202E* durch den *MAX232E* ersetzt werden. Beide Bausteine besitzen eine identische Anschlussbelegung. Der *MAX232E* benötigt eine typische Stromversorgung von 5 mA und der maximale Wert liegt bei 10 mA. Jedoch benötigt der *MAX232* Kondensatoren mit einer Kapazität von 1 µF [7]. Im Gegensatz zum *MAX202E*, welcher Kondensatoren mit einer Kapazität von 0,1 µF benötigt [6]. Um einen platzsparenden Schaltungsentwurf zu ermöglichen, wurde der *MAX202E* wegen der kleineren Bauform der Kondensatoren bevorzugt.

5 Marktanalyse bestehender Softwarelösungen

Vor der Entscheidung eine eigene Software zu implementieren, wurde eine Marktanalyse durchgeführt. Es wurde festgestellt, dass eine Testsoftware der Fa. Money Controls™ existiert [9], welche in diesem Kapitel vorgestellt wird.

5.1 Software *Money Controls Serial Hopper Test Program*

Die vollständige Bezeichnung der Software lautet: *Money Controls Serial Hopper Test Program - UnEncrypted Version*. Aus der Bezeichnung der Software lässt sich erkennen, dass diese nur die unverschlüsselte Version des Hoppers unterstützt. Die Startfläche der Software wird im Bild 5.1 angezeigt.

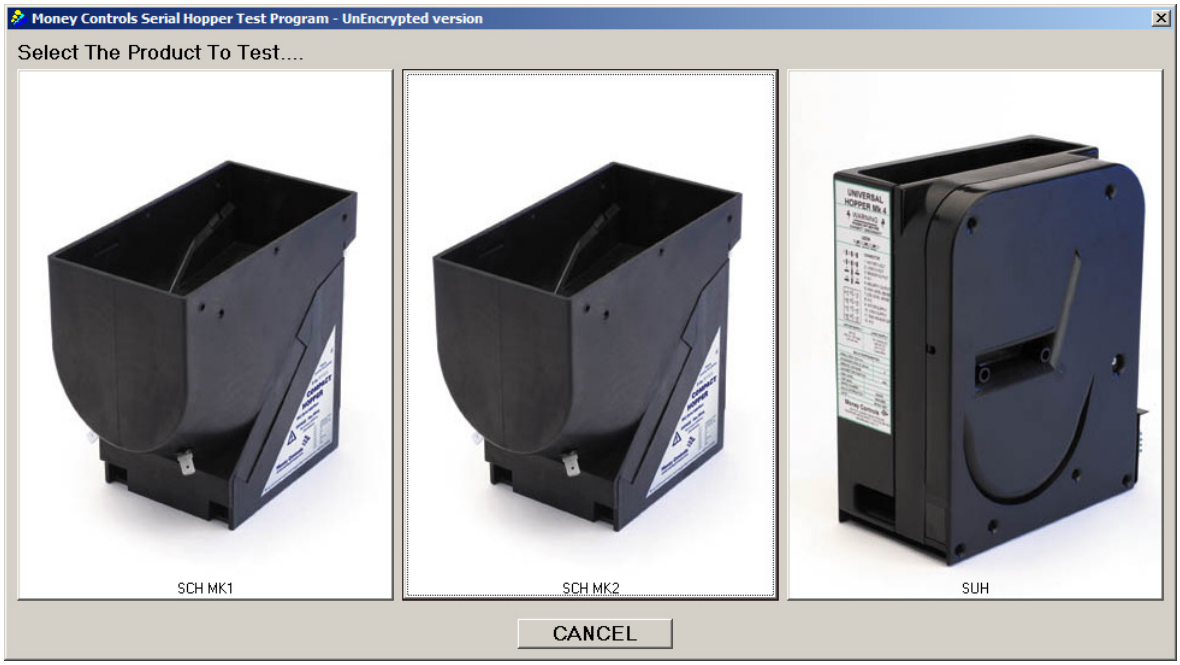

Bild 5.1: Startfläche *Money Controls Serial Hopper Test Program - UnEncrypted Version*

Auf der Startfläche der Software ist es zunächst möglich zwischen den drei Produkten: SCH MK1, SCH MK2 und Serial Universal Hopper (SUH) auszuwählen. Der Hopper MK1 ist das Vorgängermodell vom MK2. Der SUH hat eine größere Bauform und kann z. B. in den größeren Automatentypen wie *almex.station* verwendet werden. Der SUH kann bis zu 1500 Münzen aufnehmen. Nach Auswahl des Produkts SCH MK2 erscheint die im Bild 5.2 dargestellte grafische Bedienoberfläche.

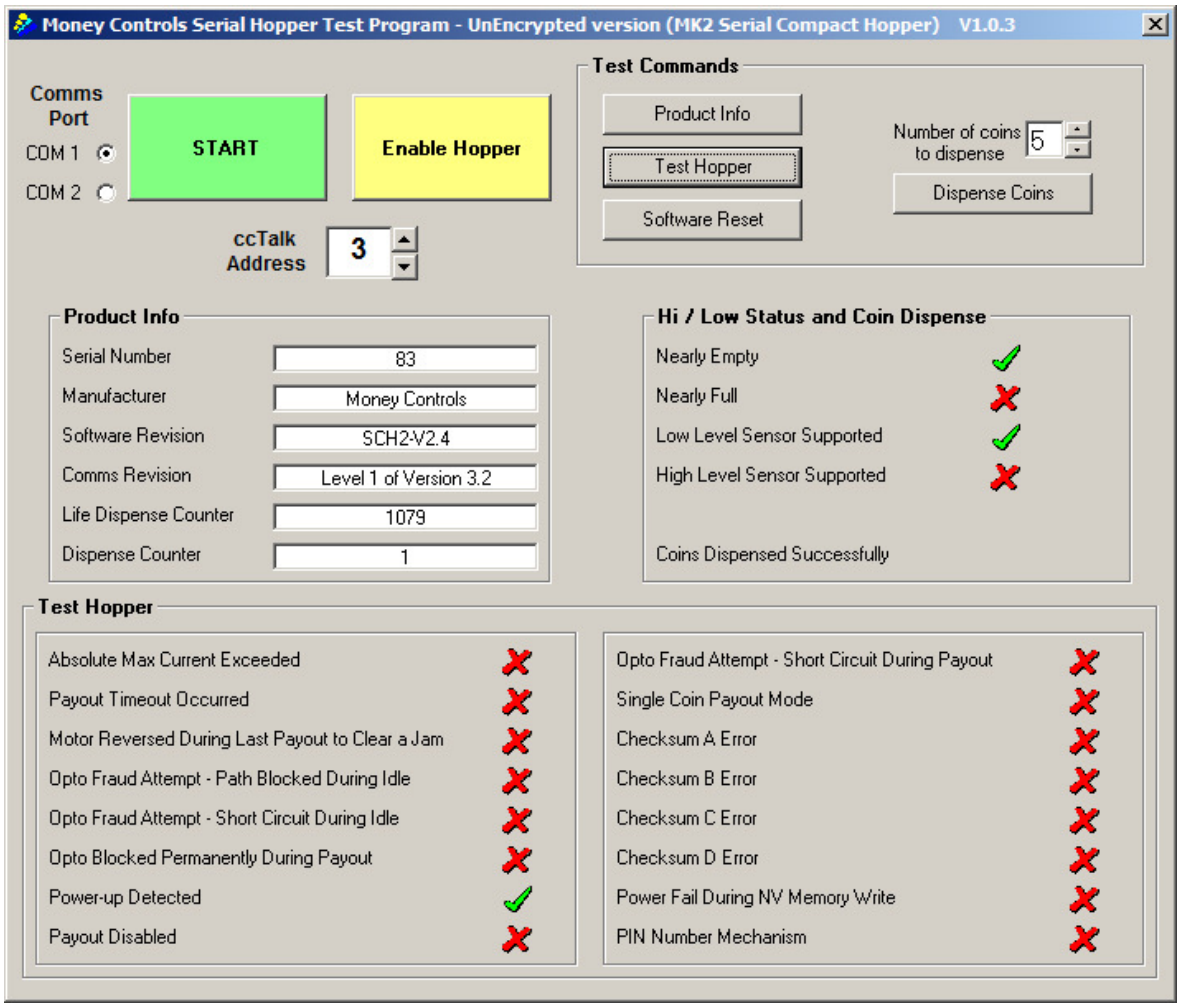

Bild 5.2: Bedienoberfläche *Test Program - UnEncrypted Version* für SCH MK2.

Es fällt zunächst auf, dass die Bedienoberfläche strukturiert aufgebaut ist. Zusammengehörige Elemente werden in Gruppen dargestellt.

Die Bedienoberfläche besteht aus:

- Den zwei Optionsfeldern zur Auswahl des COM Ports (im Bild 5.2 oben links). Die Auswahl kann zwischen den Ports eins und zwei getroffen werden.
- Der Taste **Start** (*grüne* Taste), um den ausgewählten COM Port zu öffnen.
- Dem numerischen Feld **ccTalk Address** zur Eingabe/Auswahl der Hopper-Adresse.
- Der Taste **Enable Hopper** (*gelbe* Taste), um den Hopper zu aktivieren.
- Den Gruppen **Test Commands**, **Product Info**, **Hi / Low Status and Coin Dispense** und **Test Hopper**.

Die Gruppe **Test Commands** besteht aus den vier Tasten *Product Info*, *Test Hopper*, *Software Reset*, *Dispense Coins* und dem numerischen Feld *Number of coins to dispense*. Die Bezeichnungen der Taster entsprechen den Bezeichnungen der Gruppen, in denen die jeweiligen Daten aktualisiert werden. Die Funktion der Steuerelemente (Taster und numerisches Feld) werden in der Tabelle 5.1 beschrieben.

Tabelle 5.1: Funktionsbeschreibung der Steuerelemente der Gruppe **Test Commands**

Steuerelement	Kurzbeschreibung
Product Info	Mit der Taste *Product Info* werden die allgemeinen Informationen des Hoppers in der Gruppe **Product Info** angezeigt.
Test Hopper	Mit der Taste *Test Hopper* werden die Status-/Fehlermeldungen des Hoppers in der Gruppe **Test Hopper** angezeigt. Rechts neben jeder Meldung erscheint entweder ein *rotes* Kreuz oder ein *grünes* Häkchen. Das *rote* Kreuz weist nicht auf einen Fehler hin, sondern dient dazu, dem Anwender mitzuteilen, dass die Meldung nicht zutrifft. Das *grüne* Häkchen hingegen soll mitteilen, dass die Meldung zutrifft. Z. B. im Bild 5.2 hat der Hopper ein Power-up erkannt.
Software Reset	Die Taste *Software Reset* führt den Befehl *Reset Device* aus. Nach dem Reset werden alle zuvor gesetzten Status-/Fehlermeldungen des Hoppers, sofern diese nicht noch vorhanden sind, gelöscht.
Dispense Coins	Mit der Taste *Dispense Coins* kann die Münzausgabe des Hoppers geprüft werden. Es wird die im numerischen Feld angegebene Münzanzahl ausgegeben.
Number of coins to dispense	Das numerische Feld dient zur Eingabe der auszugebenden Münzanzahl. Das Feld besitzt eine Auf- und eine Abschaltfläche mit der der Wert erhöht bzw. verringert werden kann.

In der Gruppe **Hi / Low Status and Coin Dispense** werden zum einen die im Hopper verfügbaren Sensoren und der Füllstand des Hoppers ermittelt und angezeigt. Zum anderen wird in dieser Gruppe, nachdem eine Münzausgabe gestartet wurde, angezeigt, ob die Münzausgabe erfolgt ist oder nicht. Die Anzeige erfolgt ebenfalls anhand der in der Tabelle 5.1 beschriebenen Symbole (*rotes* Kreuz oder *grünes* Häkchen).

5.2 Bewertung der Software

Die Software ist zunächst übersichtlich aufgebaut. Zusammengehörige Elemente wurden in Gruppen sortiert. Des Weiteren ist die Software selbsterklärend. Für einen einfachen Test des Hoppers wäre die Software ausreichend.

Die Software hat jedoch die folgenden Nachteile.
Es kann nur zwischen den COM Ports 1 und 2 ausgewählt werden. Das Betriebssystem Windows weist i. d. R. einem am Rechner angeschlossen USB-zu-Seriell-Adapter zunächst die Anschlussnummer 3 zu. Somit muss der Anwender dem Adapter über den Geräte-Manager zuvor eine der Anschlussnummern 1 oder 2 zuweisen.
Die wesentlichen Elemente, die diese Software nicht erfüllt, sind:

- Das Auslesen und das Schreiben des EEPROMs bzw. der Block-Nr. 0 des Hoppers, in der die ID-Nr. vom Automatenhersteller gespeichert ist.
- Des Weiteren ist es nicht möglich die Münzausgabe einer verschlüsselten Version des Hoppers zu überprüfen.
- Bei Wartungsarbeiten am Hopper (Austausch von Verschleißteilen) werden keine Meldungen über die bisher ausgegebene Gesamtmünzanzahl angezeigt.

Weiterhin fehlt eine Hilfefunktion zur Bedienung der Software.
Grundsätzlich ist es jedem Automatenhersteller selbst überlassen, wie er den Hopper bzw. die Münzwertigkeit seiner Hopper identifiziert. Aus den vorgenannten Nachteilen der bestehenden Softwarelösung war es erforderlich, eine eigene Software zu implementieren.

6 Software

Das Kapitel Software gibt zunächst einen Überblick über die Anwendung *Testsoftware for Serial Compact Hopper MK2*. Die ausführliche Beschreibung zur Bedienung der Anwendung (Bedienungsanleitung) befindet sich im Anhang. Dieses Kapitel befasst sich mit dem Design der Software. Das Design bzw. die grafische Modellierung wurde mittels Klassen- und Aktivitätsdiagrammen[7] der Unified Modeling Language (UML)[8] in der zzt. aktuellen Version 2.4.1 umgesetzt.

6.1 Überblick

Bevor die Erklärung des Designs der Software erfolgt, ist es erforderlich, auf die grafische Bedienoberfläche der Anwendung einzugehen. Die Oberfläche beinhaltet Steuerelemente wie z. B. Tasten, mit denen der Anwender bei Betätigung Ereignisse auslöst.

Die Anwendung *Testsoftware for Serial Compact Hopper MK2* besteht aus den zwei Fenstern:

1. Hauptfenster (Main Window)
2. Informationsfenster (Info Window)

Das Haupt- sowie das Informationsfenster sind auf den nachfolgenden Seiten abgebildet.

[7] Ein Klassendiagramm modelliert die statischen und zeitunabhängigen Elemente eines Systems. Ein Aktivitätsdiagramm modelliert die dynamischen Aspekte bzw. das Verhalten eines Systems und dessen Komponenten [5].
[8] UML ist eine weit verbreitete Sprache für die grafische Modellierung von Softwaresystemen. Für die Spezifikation ist das Standardisierungsgremium OMG (Object Management Group) zuständig [5].

6.1.1 Hauptfenster (Main Window)

Das Hauptfenster beinhaltet alle benötigten Steuerelemente zur Ansteuerung des Hoppers.

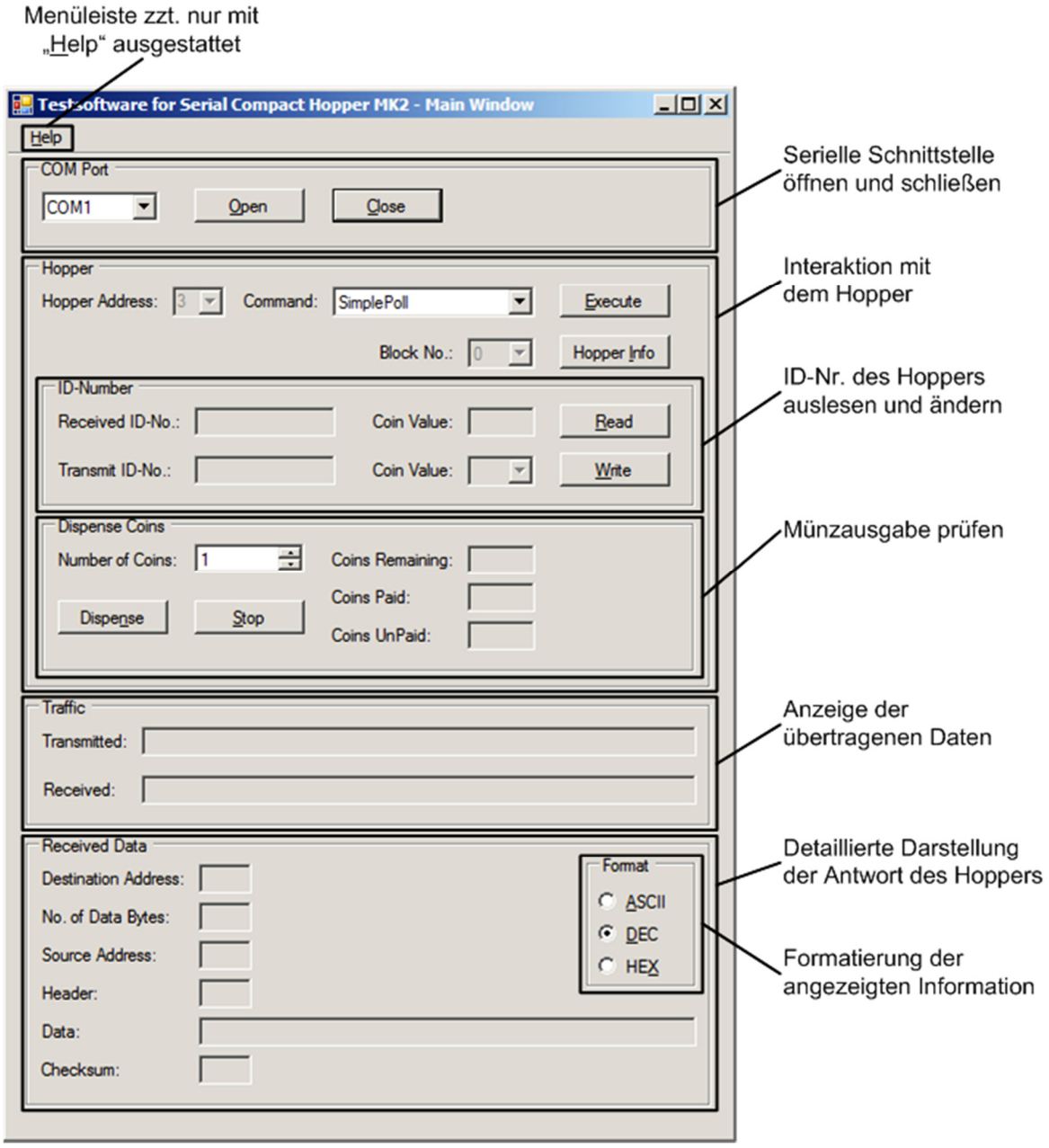

Bild 6.1: Hauptfenster der Testsoftware mit allgemeiner Funktionsbeschreibung

Das Hauptfenster besteht im Wesentlichen aus den vier Gruppen:

1. COM Port
2. Hopper
    a. ID-Number
    b. Dispense Coins
3. Traffic
4. Received Data
    a. Format

Die nachfolgende Tabelle beinhaltet eine Kurzbeschreibung der aufgelisteten Gruppen. Eine ausführlichere Beschreibung befindet sich im Anhang (s. Bedienungsanleitung).

Tabelle 6.1: Kurzbeschreibung der vier Gruppen des Hauptfensters

Nr.	Gruppe	Kurzbeschreibung
1	COM Port	Die Gruppe *COM Port* dient zur Interaktion mit einer oder mehreren am Rechner vorhandenen seriellen Schnittstellen. Die vorhandenen Schnittstellen werden im Kombinationsfeld der Gruppe aufgelistet. Mit der Taste **Open** wird die im Kombinationsfeld ausgewählte serielle Schnittstelle geöffnet und mit der Taste **Close** wird eine geöffnete serielle Schnittstelle geschlossen.
2	Hopper	Die Gruppe *Hopper* dient zur Interaktion mit dem *Serial Compact Hopper MK2* und besteht aus den Kombinationsfeldern **Hopper Address**, **Command**, **Block No.**, den Tasten **Execute** und **Hopper Info** sowie aus den zwei Gruppen *ID-Number* und *Dispense Coins*. Das Kombinationsfeld **Hopper Address** ist zzt. deaktiviert. Im Kombinationsfeld **Command** werden alle zweckmäßigen Befehle für das manuelle Testen des Hoppers aufgelistet. Durch die Auswahl eines Befehls und der Betätigung der Taste **Execute** wird der ausgewählte Befehl ausgeführt. Das Kombinationsfeld **Block No.** ist zunächst deaktiviert und wird nur aktiviert, wenn der Befehl *Read Data Block* im Kombinationsfeld **Command** ausgewählt wird. Die Taste **Hopper Info** führt eine Reihe an Befehlen aus, um die Informationen sowie die Status-/Fehlermeldungen des Hoppers anzuzeigen. Hiermit wird das Informationsfenster (Info Window) angezeigt.
2a	ID-Number	Mit der Gruppe *ID-Number* kann die vom Automatenhersteller eingestellte ID-Nr. des Hoppers ausgelesen und ggf. geändert werden. Mittels der Taste **Read** kann die ID-Nr. ausgelesen werden. Die ID-Nr. wird im Textfeld **Received ID-No.** angezeigt und im Textfeld **Coin Value** wird die dazugehörige Münzwertigkeit angezeigt. Aus der Liste des Kombinationsfelds **Coin Value** kann die neue Münzsorte ausgewählt werden und mittels der Taste **Write** wird die neue ID-Nr. im Hopper eingespeichert.

2b	Dispense Coins	Mit der Gruppe *Dispense Coins* kann die Münzausgabefunktion des Hoppers geprüft werden. In das numerische Feld **Number of Coins** kann die Anzahl der Münzen eingegeben werden, die ausgegeben werden sollen. Die Taste **Dispense** dient zur Aktivierung der Münzausgabe und die Taste **Stop** zur sofortigen Abschaltung der Münzausgabe. In den Textfeldern **Coins Remaining**, **Coins Paid** und **Coins UnPaid** wird der aktuelle Status der Münzausgabe angezeigt.
3	Traffic	In der Gruppe *Traffic* werden die gesendeten sowie die empfangenen Datenpakete angezeigt.
4	Received Data	In der Gruppe *Received Data* wird das empfangene Datenpaket bzw. die Antwort des Hoppers in seine Bestandteile zerlegt.
4a	Format	Die Gruppe *Format* dient zur Umschaltung der Darstellungsart der angezeigten Daten. Es ist möglich zwischen den Darstellungsarten **ASCII**, Dezimal (**DEC**) und Hexadezimal (**HEX**) umzuschalten.

6.1.2 Informationsfenster (Info Window)

Das Informationsfenster dient zur Anzeige von Informationen sowie von Status-/Fehlermeldungen des Hoppers.

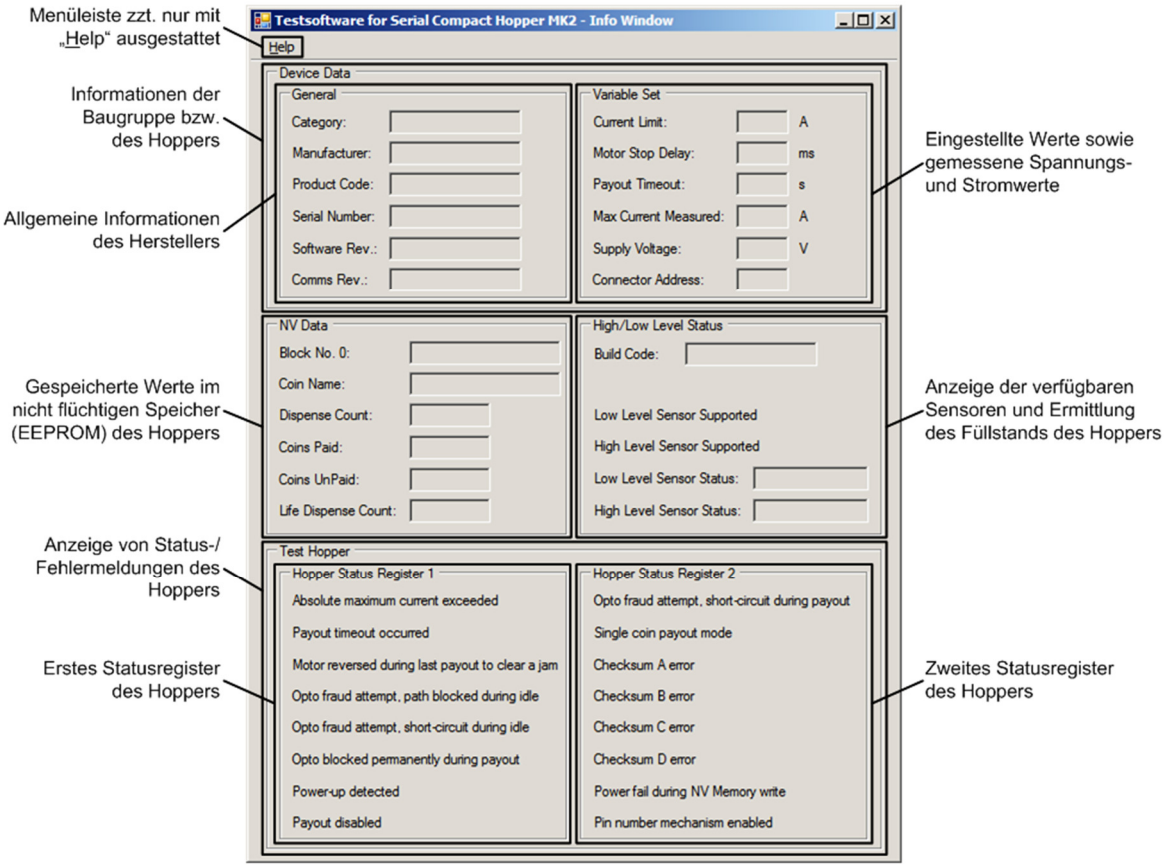

Bild 6.2: Informationsfenster der Testsoftware mit allgemeiner Funktionsbeschreibung

Das Informationsfenster besteht aus den vier Gruppen mit ihren jeweiligen Untergruppen:

1. Device Data
    a. General
    b. Variable Set
2. NV (Non Volatile) Data
3. High/Low Level Status
4. Test Hopper
    a. Hopper Status Register 1
    b. Hopper Status Register 2

Die nachfolgende Tabelle beinhaltet eine Kurzbeschreibung der aufgelisteten Gruppen. Eine ausführliche Beschreibung befindet sich im Anhang (s. Bedienungsanleitung).

Tabelle 6.2: Kurzbeschreibung der vier Gruppen des Informationsfensters

Nr.	Gruppe	Kurzbeschreibung
1	Device Data	Die Gruppe *Device Data* beinhaltet die Gruppen *General* und *Variable Set*. In der Gruppe *General* werden die allgemeinen Informationen des Hoppers angezeigt. In der Gruppe *Variable Set* werden die eingestellten Parameterwerte sowie die vom Hopper gemessenen Spannungs- und Stromwerte angezeigt.
2	NV Data	In der Gruppe *NV Data* werden die im EEPROM des Hoppers gespeicherten Werte angezeigt.
3	High/Low Level Status	In der Gruppe *High/Low Level Status* werden die im Hopper verfügbaren Sensoren und der Füllstand des Hoppers ermittelt und angezeigt.
4	Test Hopper	Die Gruppe *Test Hopper* beinhaltet die Gruppen *Hopper Status Register 1* und *Hopper Status Register 2*, in denen diverse Status-/Fehlermeldungen des Hoppers angezeigt werden.

In diesem Abschnitt wurde die grafische Bedienoberfläche dargestellt und somit ein grober Einblick in die Funktionalität der Anwendung geschaffen. Im folgenden Abschnitt wird auf das Design der Software eingegangen.

6.2 Verwendete Klassen

Die Anwendung beinhaltet u. a. die drei Klassen:

- Command
- Transmit
- Received

Die Klasse *Command* dient als Vorlage für den Aufbau eines zu sendenden Befehls. Die Klasse *Transmit* ist für den Aufbau eines zu sendenden Datenpakets zuständig. Die Klasse *Received* wiederum dient als Vorlage für den Aufbau eines zu empfangenen Datenpakets. Das empfangene Datenpaket enthält das Echo des gesendeten Datenpakets und die Antwort des Hoppers. Deshalb wird die Klasse *Received* von der Klasse *Transmit* geerbt.

6.2.1 Klasse *Command*

Die Klasse *Command* dient dazu, die Informationen, welche mit einem Befehl eng verknüpft sind, in einem Objekt zusammenzufassen. Die Informationen bzw. Eigenschaften der Klasse sind im Bild 6.3 zu sehen und werden in der Tabelle 6.3 beschrieben.

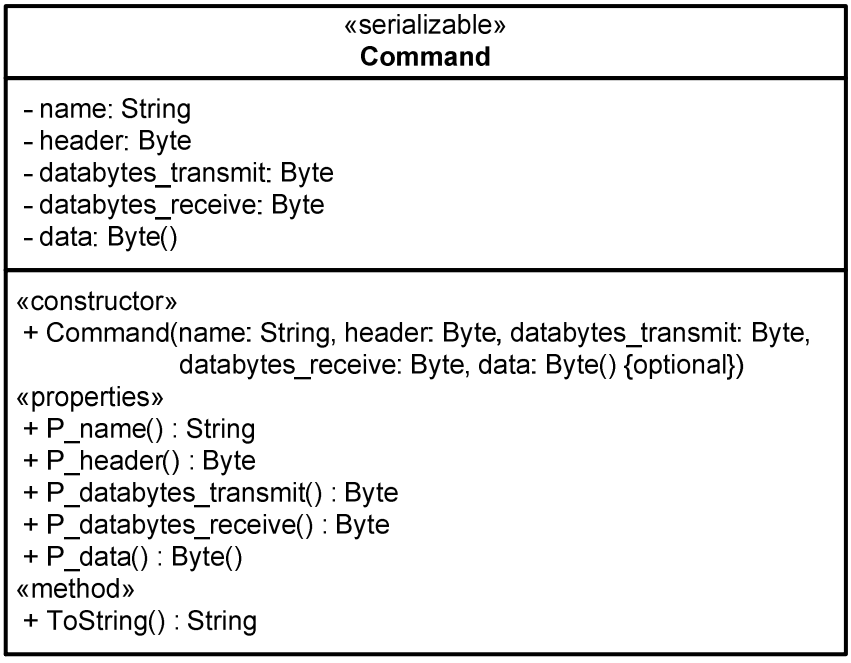

Bild 6.3: Klasse *Command* mit Eigenschaften und Operationen

Tabelle 6.3: Beschreibung der Eigenschaften der Klasse *Command*

Eigenschaft	Beschreibung
name	Die Befehlsbezeichnung dient zur Zuordnung. Soll ein Befehl ausgeführt werden, kann z. B. die Bezeichnung aus dem Kombinationsfeld **Command** entnommen werden.
header	Anhand des Headers kann der Hopper einen Befehl erkennen.
databytes_transmit	Mit der Anzahl der zu sendenden Datenbytes kann die Länge eines zu sendenden Befehls festgelegt werden.
databytes_receive	Anhand der Anzahl der zu sendenden sowie der zu empfangenen Datenbytes kann die Länge eines zu empfangenen Datenpakets festgelegt werden.
data	Einige Befehle enthalten spezielle Datenbytes, die mitgesendet werden müssen.

Die Eigenschaften der Klasse *Command* sind *private* und deshalb nur innerhalb der Klasse sichtbar. Der Zugriff auf die Eigenschaften außerhalb der Klasse erfolgt nur über die Eigenschaftsmethoden, welche als *properties* gekennzeichnet sind. Dies ermöglicht einen kontrollierten Zugriff auf die Eigenschaften. Die Methode *ToString* wird überschrieben und gibt die Befehlsbezeichnung als Zeichenkette zurück.

Der Konstruktor beinhaltet alle Attribute der Klasse. Wobei zu beachten ist, dass das Attribut *data* im Konstruktor die Eigenschaft *optional* besitzt, da nicht alle Befehle Datenbytes beinhalten und somit dieser Wert nicht immer mit übergeben werden muss. In diesem Fall bekommt das Attribut den Wert *Nothing* zugewiesen.

In Bild 6.3 ist über dem Klassennamen *Command* die Bezeichnung *serializable* zu sehen. Diese wird in UML als Stereotyp bezeichnet und gibt die Auskunft, dass die Klasse serialisierbar ist. Dies wird benötigt, damit die Objekte der Klasse als solches in einer Datei gespeichert werden können.

6.2.2 Klasse *Transmit*

Die Klasse *Transmit* dient dazu, alle Informationen in einem Objekt zusammenzufassen, welche in Form eines Datenpakets an den Hopper gesendet werden sollen.
Im nachfolgenden Bild 6.4 sind die Eigenschaften und Operationen der Klassen *Transmit* und *Received* dargestellt. Die Klasse *Received* wird im Kapitel 6.2.3 erläutert.

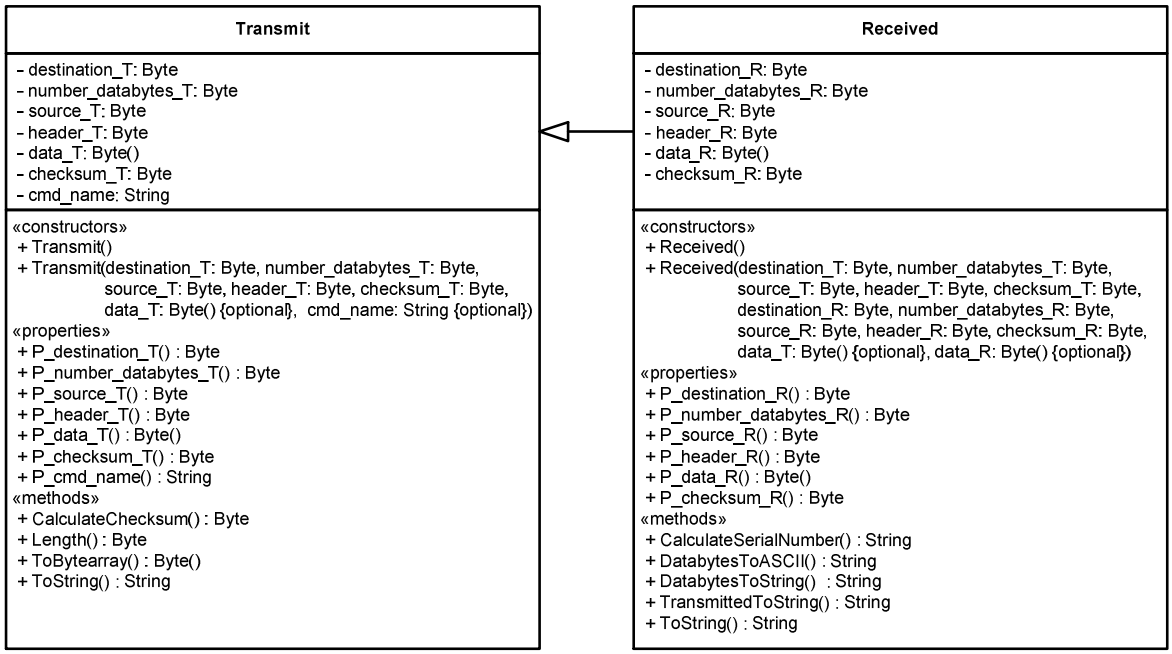

Bild 6.4: Klassen *Transmit* und *Received* mit Eigenschaften und Operationen

Die Eigenschaften der Klasse sind in der nachfolgenden Tabelle aufgelistet.

Tabelle 6.4: Kurzbeschreibung der Eigenschaften der Klasse *Transmit*

Bezeichnung	Kurzbeschreibung
destination_T	Zieladresse
number_databytes_T	Anzahl der Datenbytes
source_T	Quelladresse
header_T	Header
data_T	Datenbytes
checksum_T	Prüfsumme
cmd_name	Befehlsbezeichnung

Die Bezeichnungen *destination_T* bis *checksum_T* sind die einzelnen Elemente eines zu sendenden Datenpakets. Das Datenpaket wurde in Kapitel 3.2 detailliert beschrieben.

Die Befehlsbezeichnung bzw. die Variable *cmd_name* wird benötigt, um bei der Überprüfung der empfangenen Daten, diese dem jeweils gesendeten Befehl zuzuordnen. Sicherlich könnte dies auch mittels des Headers realisiert werden. Die Entscheidung für die Realisierung über die Befehlsbezeichnung dient allerdings zur besseren Lesbarkeit des Programms. Die Überprüfung eines empfangenen Datenpakets geschieht nicht direkt nach einem gesendeten Datenpaket, sondern in einem separaten Thread.

Die Eigenschaften der Klasse *Transmit* sind *private* und deshalb nur innerhalb der Klasse sichtbar. Der Zugriff auf die Eigenschaften außerhalb der Klasse erfolgt nur über die Eigenschaftsmethoden, welche als *properties* gekennzeichnet sind. Die Methoden der Klasse *Transmit* werden im Anhang dargestellt.

Die Klasse *Transmit* besitzt zwei Konstruktoren. Zum einen den parameterlosen Standardkonstruktor und zum anderen einen Konstruktor, der alle Eigenschaften der Klasse beinhaltet. Es ist zu beachten, dass die Attribute *data_T* und *cmd_name* im zweiten Konstruktor die Eigenschaft *optional* besitzen. Falls diese Eigenschaften nicht angegeben werden, bekommen sie den Wert *Nothing* zugewiesen.

6.2.3 Klasse *Received*

Die Klasse *Received* dient dazu, die empfangenen Daten in einem Objekt zusammenzufassen. Die empfangenen Daten bestehen aus dem Echo des gesendeten Datenpakets und der Antwort des Hoppers. Da die gesendeten Daten als Echo empfangen werden, wird die Klasse *Received* von der Klasse *Transmit* geerbt. Die Eigenschaften und Operationen der Klasse *Received* sind im Bild 6.4 dargestellt. Die Eigenschaften der Klasse sind in der nachfolgenden Tabelle aufgelistet.

Tabelle 6.5: Kurzbeschreibung der Eigenschaften der Klasse *Received*

Bezeichnung	Kurzbeschreibung
destination_R	Zieladresse
number_databytes_R	Anzahl der Datenbytes
source_R	Quelladresse
header_R	Header
data_R	Datenbytes
checksum_R	Prüfsumme

Die Eigenschaften sind die einzelnen Elemente eines zu empfangenen Datenpakets. Das zu empfangene sowie gesendete Datenpaket haben denselben Rahmenaufbau, welcher in Kapitel 3.2 detailliert beschrieben wurde.

Die Eigenschaften der Klasse *Received* sind *private* und deshalb nur innerhalb der Klasse sichtbar. Der Zugriff auf die Eigenschaften außerhalb der Klasse erfolgt nur über die Eigenschaftsmethoden, welche als *properties* gekennzeichnet sind. Zusätzlich kann die Klasse *Received* auch auf die Eigenschaftsmethoden der Klasse *Transmit* zugreifen.

Die Methoden der Klasse *Received* werden im Anhang dargestellt. Zusätzlich kann ein Objekt der Klasse *Received* auch auf die Methoden der Klasse *Transmit* zugreifen.

Die Klasse *Received* besitzt zwei Konstruktoren. Zum einen den parameterlosen Standardkonstruktor und zum anderen einen Konstruktor, der alle Eigenschaften der Klasse beinhaltet. Wobei zu beachten ist, dass die Eigenschaften *data_T* und *data_R* im zweiten Konstruktor die Eigenschaft *optional* besitzen, da nicht alle gesendeten oder empfangenen Datenpakete Datenbytes beinhalten. Falls diese Eigenschaften nicht angegeben werden, bekommen sie den Wert *Nothing* zugewiesen.

Die Angabe der zwei Konstruktoren bei den Klassen *Transmit* und *Received* hat den Hintergrund, dass dem Programmierer zwei Möglichkeiten zur Verfügung gestellt werden.
Die erste Möglichkeit besteht darin, mit dem Standardkonstruktor ein globales Objekt der Klasse zu erstellen, welches den Sichtbarkeitsmodifikator *public* vorangeschrieben wird. Durch dieses Vorgehen ist das soeben erstellte Objekt im gesamten Projekt sichtbar. Zu einem bestimmten Zeitpunkt können dann in einem geeigneten Modul die zu sendenden oder empfangenen Daten mittels der Eigenschaftsmethoden *(properties)* dem globalen Objekt zugewiesen werden.
Die zweite Möglichkeit sieht vor, mit dem zweiten Konstruktor für jedes zu sendende oder empfangene Datenpaket ein eigenes Objekt zu erstellen, indem die Eigenschaften direkt übergeben werden.
Beide Möglichkeiten haben Vor- und Nachteile. Deshalb wurden beide implementiert. Das Programm wurde nach der ersten Möglichkeit realisiert. Da die Befehle sequentiell übertragen werden ist nur jeweils ein globales Objekt ausreichend. Soll ein weiterer Befehl gesendet werden, braucht kein weiteres Objekt erzeugt werden, sondern das bereits Erstellte kann weiterverwendet werden.

6.3 Aktivitätsdiagramme

Die vorliegenden Aktivitätsdiagramme dienten als Grundlage bzw. Arbeitsvorlage für die Implementierung der Testsoftware und beschreiben die Arbeitsweise der Software. Die Software ist modular aufgebaut. Die Aufteilung der Software in verschiedene Module hat mehrere Vorteile. Beispielhaft sollen die folgenden beiden Vorteile erwähnt werden. Zum einen dient die Aufteilung zur Übersichtlichkeit des Projekts. Zum anderen können die verschiedenen Module auch in zukünftigen Projekten eingesetzt werden.

Zur angenehmeren Lesbarkeit der Aktivitätsdiagramme sind alle Bezeichnungen in deutscher Sprache verfasst worden. Um die Anbindung zum daraus umgesetzten Quellcode, welcher ausschließlich aus englischen Bezeichnungen besteht, nicht zu verlieren, sind die englischen Bezeichnungen, dort wo sie benötigt werden, neben die deutsche Bezeichnung in Klammern gesetzt worden.

6.3.1 Ereignis *Execute betätigt*

Der Anwender betätigt die Taste **Execute**, nachdem er einen Befehl aus dem Kombinationsfeld **Command** ausgewählt hat. Durch die Betätigung der Taste **Execute** mit der Maus oder der Taste **E** wird das Ereignis *Execute betätigt* ausgelöst.

Das Diagramm in Bild 6.5 besteht aus den vier Schnittstellen Anwender, Programm, Serielle Schnittstelle des Rechners und Hopper. Beim Ereignis *Execute betätigt* sind die folgenden drei Modulen des Programms involviert:

- Klasse *MainWindow*
- Modul *Transceive*
- Modul *MySerialPort*

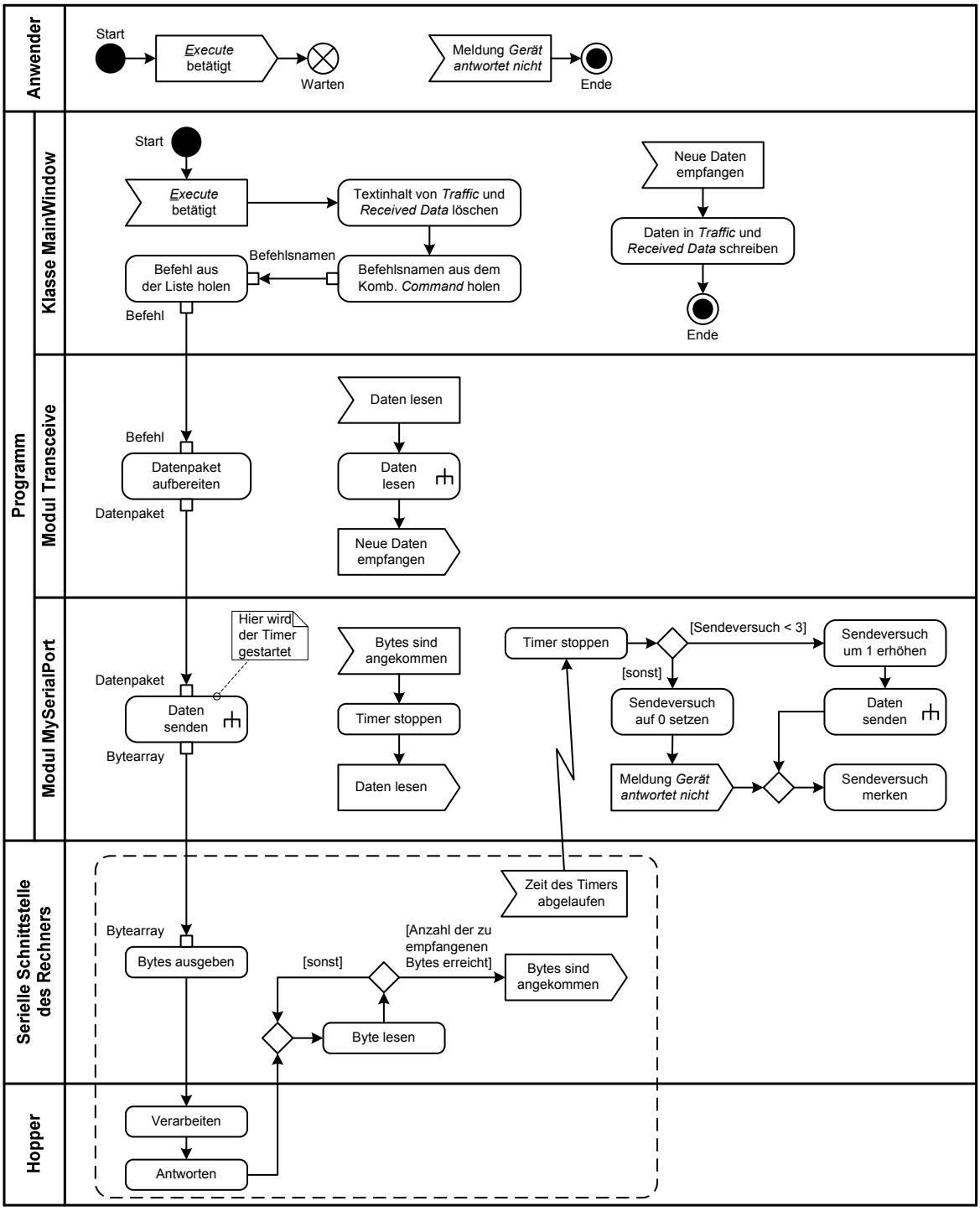

Bild 6.5: Aktivitätsdiagramm zum Ereignis *Execute betätigt*

Die Klasse *MainWindow* empfängt das Signal, welches durch die Betätigung der Taste gesendet wurde. Zunächst werden die vorherigen Textinhalte der Textfelder der Gruppen *Traffic* und *Received Data* gelöscht, damit auch im Fehlerfall keine alten Daten angezeigt bleiben. Im nächsten Schritt wird der Befehlsname aus dem Kombinationsfeld **Command** entnommen, um im nachfolgenden Schritt den Befehl aus der Befehlsliste, welche sich im Speicher befindet, zu holen.

Ein Befehl ist ein Objekt der Klasse *Command*, welche im Kapitel 6.2.1 bereits beschrieben wurde. Die Befehlsliste enthält alle wichtigen Informationen, um einen Befehl an den Hopper zu senden. Die Klasse *MainWindow* ruft die Methode *Datenpaket aufbereiten* des Moduls *Transceive* auf und übergibt ihr den *Befehl*. Die Funktion der Methode *Datenpaket aufbereiten* besteht darin, dass zu sendende Datenpaket zusammenzustellen und dieses an die Aktivität *Daten senden* des Moduls *MySerialPort* zu übergeben. Die Aktivität *Daten senden* sendet dann die Daten in Form eines Bytearrays über die serielle Schnittstelle des Rechners an den Hopper. Die Aktivität *Daten senden* wird im Kapitel 6.3.2 dargestellt und ausführlicher beschrieben. Der Hopper verarbeitet die auf dem *ccTalk*-Bus angekommenen Bytes und antwortet entsprechend.

Die Programmabschnitte, welche für die Kommunikation des Hoppers zuständig sind, sind ereignis-/zeitgesteuert implementiert worden. Der Ablauf ist so konzipiert, dass nach einem gesendeten Befehl nicht auf die Antwort des Hoppers gewartet wird, sondern weitere Programmaufgaben erledigt werden können. Z. B. den soeben gesendeten Befehl in einer Logdatei zu speichern.

Nach dem Senden eines Befehls gibt es zwei Möglichkeiten:

1. Der Hopper antwortet. Nachdem die zu erwartende Anzahl an Bytes angekommen ist, wird das Ereignis *Daten sind angekommen* auslöst.
2. Der Hopper antwortet nicht. Nach dem Senden des Befehls wird ein Zeitgeber gestartet. Ist dieser abgelaufen, wird das Ereignis *Zeit des Timers abgelaufen* ausgelöst.

1. Ereignis *Daten sind angekommen*

Wenn dieses Ereignis eintritt, ist die Antwort des Hoppers angekommen. Die Antwort bzw. die Bytes liegen im Empfangspuffer der seriellen Schnittstelle zur Abholung bereit. Zunächst wird der Zeitgeber, welcher für die Sendewiederholung zuständig ist, gestoppt. Für die Abholung der Daten ist die Aktivität *Daten lesen* des Moduls *Transceive* zuständig. Das Modul *MySerialPort* sendet das Signal an die Aktivität *Daten lesen*, welche für das Auslesen der Daten aus dem Empfangspuffer und für die Zuweisung der Daten an dem Objekt *Empfangenes Datenpaket* zuständig ist. Das Modul *Transceive* sendet nach Ablauf der Aktivität *Daten lesen* seinerseits das Signal *Neue Daten empfangen* an die Klasse *MainWindow*, welches die empfangenen Daten in der Textfeldern der Gruppen *Traffic* und *Received Data* einfügt. Der Anwender bekommt die Daten somit angezeigt.

2. Ereignis *Zeit des Timers abgelaufen*

Dieses Ereignis tritt ein, wenn die Anzahl der zu erwartenden Bytes nicht am Empfangspuffer der seriellen Schnittstelle angekommen sind. Möglicherweise antwortet der Hopper nicht. Dies kann verschiedene Ursachen haben wie z. B. Störungen auf dem Bus. Möglich ist auch, dass der Hopper gerade mit einer anderen Aufgabe beschäftigt ist und nicht auf den gesendeten Befehl reagiert. Aus diesem Grund wird der Befehl erneut gesendet. Die Sendewiederholung ist zzt. auf drei Wiederholungen eingestellt. Dieser Wert wurde in der Testphase ermittelt und bestimmt.

In [14] ist sogar eine Sendewiederholung mit bis zu zehn Wiederholungen angegeben. Es handelt sich hierbei aber nur um einen empfohlenen Wert für den Befehl *ReqHopper DispenseCount*. Dieser wird zur Laufzeit des Motors in kurzen Abständen (z. B. alle 200 ms) gesendet, um den Status der Münzausgabe zu aktualisieren. Während der Motorlaufzeit können Störungen auf den Leitungen entstehen, die einen Bitfehler verursachen. In der Folge kann der Hopper den Befehl nicht mehr als solchen erkennen.

6.3.2 Aktivität *Daten senden*

Die Aktivität *Daten senden* befindet sich im Modul *MySerialPort* und dient zum Senden von Datenpaketen über die serielle Schnittstelle. Zunächst wird der Inhalt des Empfangs- und Ausgangspuffers gelöscht und die zu erwartende Byteanzahl am Empfangspuffer festgelegt. Beim Aufruf der Aktivität wird das zu sendende Datenpaket als Parameter übergeben und in Form eines Bytearrays an den TxD-Ausgang der seriellen Schnittstelle weitergeleitet. Nachdem das Datenpaket im Puffer geschrieben wurde, wird der Timer gestartet.

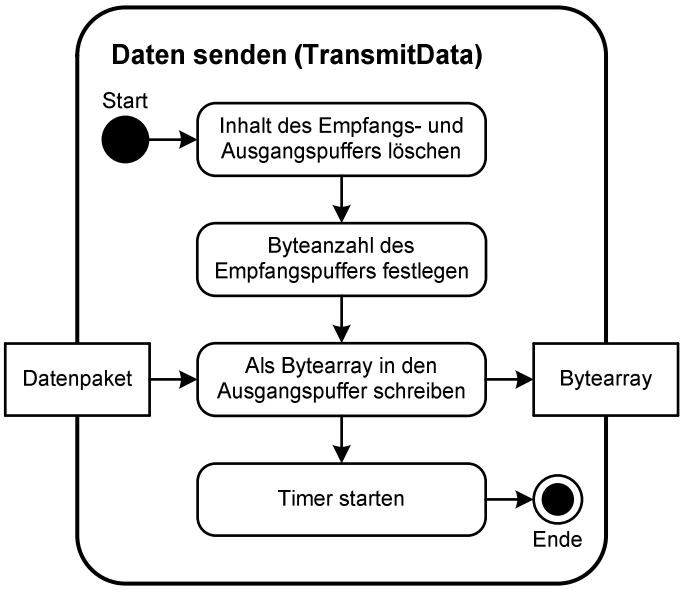

Bild 6.6: Aktivitätsdiagramm der Methode *Daten senden* des Moduls *MySerialPort*

6.3.3 Aktivität *Daten lesen*

Die Aktivität *Daten lesen* befindet sich im Modul *Transceive* und dient zur Abholung der am Empfangspuffer der seriellen Schnittstelle anliegenden Bytes und deren Zuweisung am Objekt *Empfangenes Datenpaket*.

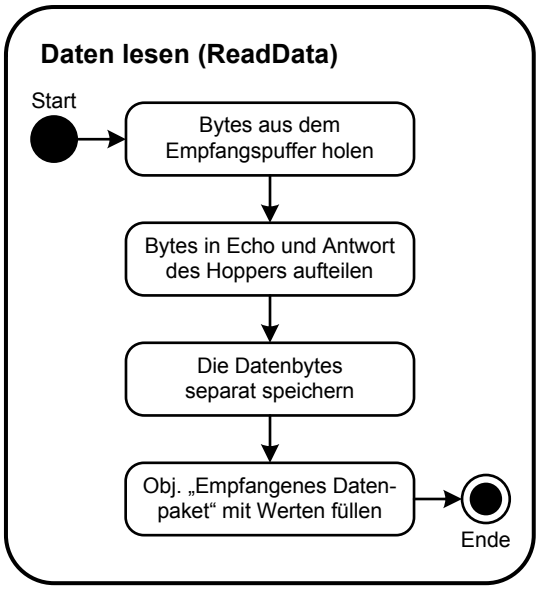

Bild 6.7: Aktivitätsdiagramm der Methode *Daten lesen* des Moduls *Transceive*

6.3.4 Ereignis *Read betätigt*

Das folgende Diagramm zeigt den Programmablauf nach der Betätigung der Taste **Read** an. Durch die Betätigung der Taste wird das Signal *Read betätigt* gesendet, welches in der Klasse *MainWindow* empfangen wird. In der Klasse *MainWindow* wird zunächst dem Befehl *ReadDataBlock* das Datenbyte 0 zugewiesen. Anschließend wird der Befehl aus der Befehlsliste geholt und der Methode *Datenpaket aufbereiten* des Moduls *Transceive* übergeben, welches für die Weiterleitung des Befehls an die serielle Schnittstelle zuständig ist. Dem Befehl *ReadDataBlock* wird das Datenbyte 0 zugewiesen, um den Speicherinhalt bzw. die Bytes der Block-Nr. 0 des Hoppers zu erhalten, aus denen sich dann die ID-Nr. berechnen lässt.

Zur übersichtlichen Darstellung wurde das Diagramm nur bis einschließlich des Moduls *Transceive* gezeichnet. Der weitere Verlauf ab und innerhalb des Moduls *Transceive* wurde bereits in Bild 6.5 dargestellt. Das Modul *Transceive* wird außerdem angezeigt, um auszudrücken, aus welchem Modul das empfangene Signal *Neue Daten empfangen* in der Klasse *MainWindow* ausgelöst wird.

Wenn der Befehl erfolgreich ausgeführt wird und die Antwort des Hoppers angekommen ist, wird das Modul *Transceive* das Signal *Neue Daten empfangen* senden, welches in der Klasse *MainWindow* empfangen wird. Nun werden die in *MainWindow* empfangenen Daten zunächst in den Textfeldern der Gruppen *Traffic* und *Received Data* eingefügt, damit der Anwender die *„rohen“* Daten angezeigt bekommt. Da es sich bei dem gesendeten Befehl um den *ReadDataBlock* handelt, wird die Methode *ID-Nr. aus den Datenbytes berechnen* aufgerufen und die daraus berechnete ID-Nr. wird im Textfeld *Received ID-No.* der Gruppe *ID-Number* angezeigt.

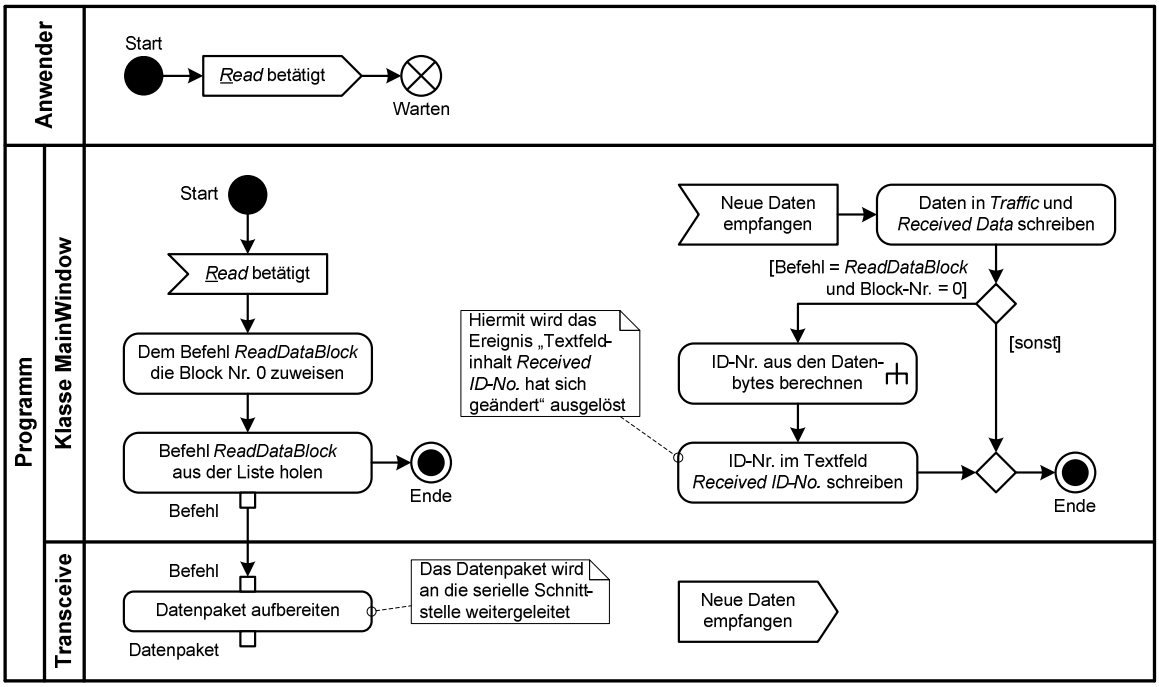

Bild 6.8: Aktivitätsdiagramm zum Ereignis *Read betätigt*

6.3.5 Aktivität *ID-Nr. aus den Datenbytes berechnen*

Die Aktivität *ID-Nr. aus den Datenbytes berechnen* befindet sich im Modul *Calculate* und dient zur Berechnung der ID-Nr. aus den empfangenen Datenbytes.

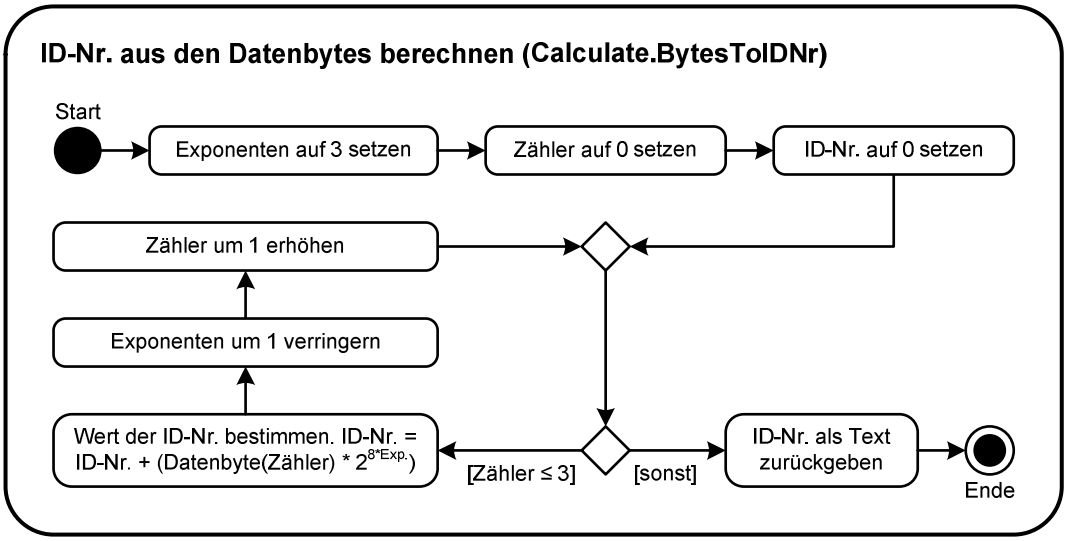

Bild 6.9: Aktivitätsdiagramm *ID-Nr. aus Datenbytes berechnen* des Moduls *Calculate*

Für die Berechnung der ID-Nr. werden immer nur die ersten vier der acht empfangenen Datenbytes benötigt. Die Berechnungsvorschrift lautet:

$$\text{ID Nr.} = [1tes\ Byte] \times 2^{24} + [2tes\ Byte] \times 2^{16} + [3tes\ Byte] \times 2^{8} + [4tes\ Byte] \times 2^{0}$$

Beispiel: Die empfangenen Datenbytes lauten: 2, 252, 210 und 192.
Laut der Berechnungsvorschrift ergibt sich somit die eingestellte ID-Nr. des Hoppers aus
$2 \times 2^{24} + 252 \times 2^{16} + 210 \times 2^{8} + 192 \times 2^{0} = 50123456$
und lautet 50123456.

6.3.6 Ereignis *Textfeldinhalt Received ID-No. hat sich geändert*

Das Ereignis *Textfeldinhalt Received ID-No. hat sich geändert* wird ausgelöst, sobald eine ID-Nr. empfangen wurde.
Nach dem Auslösen dieses Ereignisses, werden die folgenden drei Aufgaben bearbeitet:

- Den Textinhalt des Textfelds **Coin Value** setzen.
- Die Empfangene ID-Nr. auf Richtigkeit überprüfen.
- Die Elemente des Kombinationsfelds **Coin Value** verwalten.

Primär wird dieses Ereignis dafür benötigt, um aus der empfangenen ID-Nr., die entsprechende Münzwertigkeit im Textfeld **Coin Value** anzuzeigen. Zusätzlich wird dieses Ereignis dafür verwendet, um die noch übrigen Münzsorten, aus denen der Anwender die neue Münzsorte wählen kann, für die der Hopper zuständig sein soll, in das Kombinationsfeld **Coin Value** zu setzen.
Zunächst werden, nachdem das Ereignis ausgelöst wurde, die vorhandenen Münzsorten aus dem Kombinationsfeld **Coin Value** entfernt und alle möglichen Münzsorten hinzugefügt. Dies kann vor der Überprüfung der ID-Nr. stattfinden, da, wenn dem Hopper keine ID-Nr. zugewiesen wurde, alle Münzwertigkeiten zur Verfügung gestellt werden können. Sollte der Hopper bereits eine ID-Nr. haben, muss nur noch die aktuell eingestellte Münzsorte aus dem Kombinationsfeld entfernt werden.
Aus der ersten Ziffer der ID-Nr. kann festgestellt werden, ob es sich um eine gültige ID-Nr. handelt. Wenn die Ziffer den Wert 5 hat, handelt es sich um eine gültige ID-Nr. und es wird zunächst die Hintergrundfarbe des Textfelds *Received ID-No.* entfernt. Dies ist erforderlich, da es vorkommen kann, dass vorher versucht wurde, die ID-Nr. eines Hoppers auszulesen, dem noch keine ID-Nr. zugewiesen wurde. In diesem Fall wäre die Hintergrundfarbe des Textfelds *Received ID-No.* rot gewesen. Aus der zweiten Ziffer der ID-Nr. kann die Münzwertigkeit des Hoppers festgestellt werden. In Abhängigkeit der zweiten Ziffer wird das Textfeld **Coin Value** gesetzt und die Münzsorte aus dem Kombinationsfeld **Coin Value** entfernt. Sollte es sich bei der ersten Ziffer der ID-Nr. nicht um den Wert 5 handeln, handelt es sich um eine ungültige ID-Nr. bzw. dem Hopper wurde zuvor noch keine ID-Nr. zugewiesen. In diesem Fall wird zunächst die Hintergrundfarbe des Textfelds *Received ID-No.* rot eingefärbt. Es folgt die Ermittlung der zuletzt eingetragenen ID-Nr. aus der Textdatei *„ID-Number_list.txt“*. Sollte die Datei nicht gefunden werden, wurde noch keine ID-Nr. vergeben und es kann die ID-Nr. *50999999* verwendet werden. Im Textfeld *Transmit ID-No.* wird dann entweder der Wert der zuletzt vergebenen ID-Nr. um eins verringert oder die ID-Nr. *50999999* geschrieben. Anschließend wird das Kombinationsfeld **CoinValue** aktiviert und das erste Element der Liste ausgewählt.

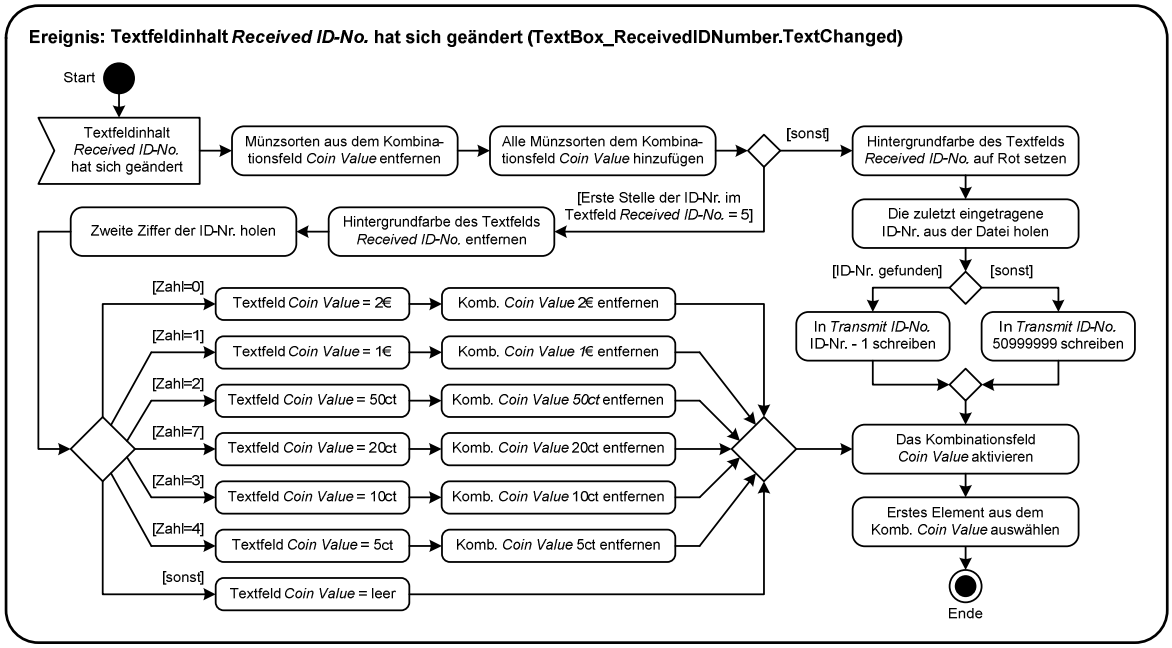

Bild 6.10: Aktivitätsdiagramm zum Ereignis *Textfeldinhalt Received ID-No. hat sich geändert*

6.3.7 Ereignis *Write betätigt*

Das folgende Diagramm zeigt den Programmablauf nach der Betätigung der Taste **Write** an.

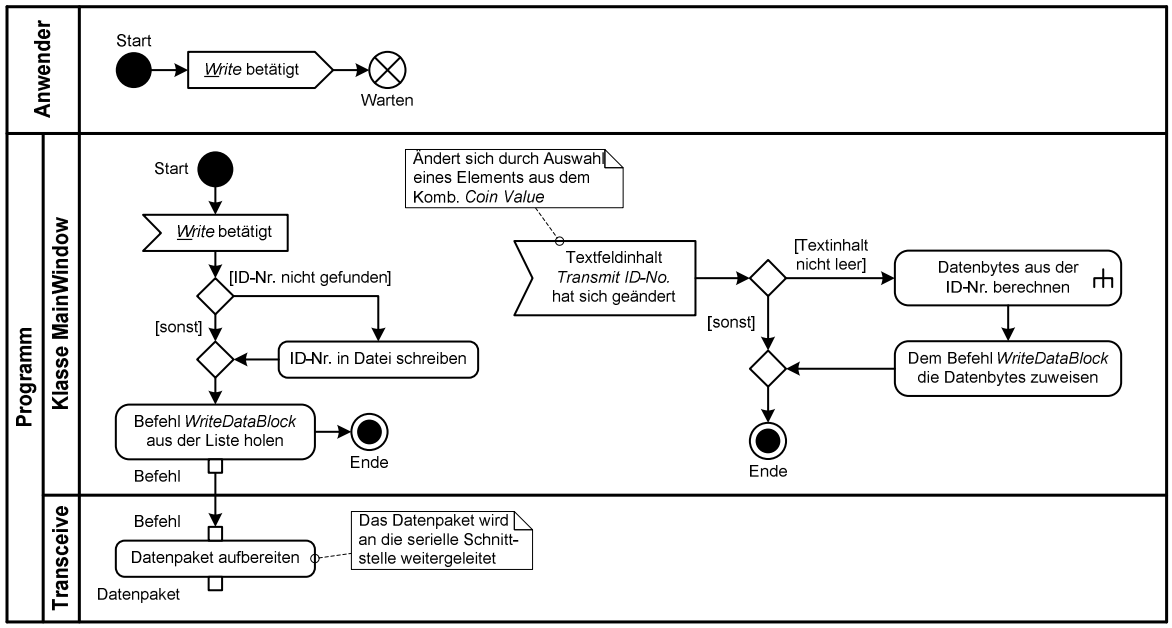

Bild 6.11: Aktivitätsdiagramm zum Ereignis *Write betätigt*

Die ID-Nr. eines Hoppers kann mittels der Taste **Write** nur geändert werden, wenn die im Hopper eingespeicherte ID-Nr. vorher mittels der Taste **Read** eingelesen wurde.
Der Inhalt des Textfelds *Transmit ID-No.* wird entweder durch die Taste **Read** oder durch die Auswahl eines Elements aus dem Kombinationsfeld **Coin Value** geändert. Somit wird das Ereignis *Inhalt des Textfelds Transmit ID-No. hat sich geändert* ausgelöst. Hier wird zunächst geprüft, ob der Inhalt des Textfelds nicht leer ist. Trifft dies zu, können aus der ID-Nr. die Datenbytes berechnet werden und dem Befehl *WriteDataBlock* zugewiesen werden.

Durch die Betätigung der Taste **Write** wird das Signal *Write betätigt* gesendet, welches in der Klasse *MainWindow* empfangen wird. In der Klasse *MainWindow* wird zunächst geprüft, ob im Textfeld *Received ID-No.* die Meldung erscheint, dass eine ID-Nr. nicht gefunden wurde. In diesem Fall wird die zu übertragene ID-Nr. in der Textdatei *„ID-Number_list.txt“* gespeichert. Anschließend wird der Befehl *WriteDataBlock* aus der Befehlsliste geholt und der Methode *Datenpaket aufbereiten* des Moduls *Transceive* übergeben, welches für die Weiterleitung des Befehls an die serielle Schnittstelle zuständig ist.

Zur übersichtlichen Darstellung wurde das Diagramm, wie schon beim Ereignis *Read betätigt,* nur bis einschließlich des Moduls *Transceive* gezeichnet. Der weitere Verlauf ab und innerhalb des Moduls *Transceive* wurde bereits in Bild 6.5 dargestellt.

6.3.8 Aktivität *Datenbytes aus der ID-Nr. berechnen*

Die Aktivität *Datenbytes aus der ID-Nr. berechnen* befindet sich im Modul *Calculate* und dient zur Berechnung der zu sendenden Datenbytes aus der ID-Nr.

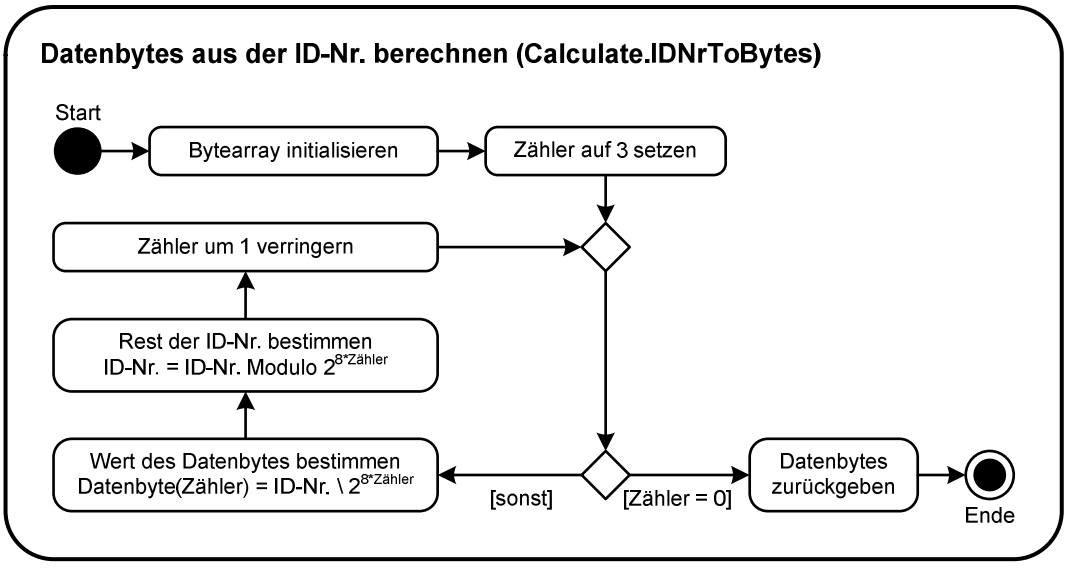

Bild 6.12: Aktivitätsdiagramm *Datenbytes aus der ID-Nr. berechnen* des Moduls *Calculate*

Die Berechnungsvorschrift, um aus der ID-Nr. die vier zu sendenden Datenbytes zu bestimmen lautet:

$[1tes\ Byte] = IDNr.\ \backslash\ 2^{24}$

$[2tes\ Byte] = (IDNr.\ mod\ 2^{24}) \backslash\ 2^{16}$

$[3tes\ Byte] = (IDNr.\ mod\ 2^{16}) \backslash\ 2^{8}$

$[4tes\ Byte] = (IDNr.\ mod\ 2^{8}) \backslash\ 2^{0}$

Beispiel: Die ID-Nr. lautet: 50123456.

Laut der Berechnungsvorschrift ergeben sich somit die an den Hopper zu sendenden Datenbytes aus

$[1tes\ Byte] = 50123456 \backslash\ 2^{24} = 2$

$[2tes\ Byte] = (50123456\ mod\ 2^{24}) \backslash\ 2^{16} = 252$

$[3tes\ Byte] = (IDNr.\ mod\ 2^{16}) \backslash\ 2^{8} = 210$

$[4tes\ Byte] = (IDNr.\ mod\ 2^{8}) \backslash\ 2^{0} = 192$

und lauten 2, 252, 210 und 192.

6.3.9 Ereignis *Index des Komb. Coin Value hat sich geändert*

Wird das Ereignis *Index des Kombinationsfelds Coin Value hat sich geändert* ausgelöst, wird die angezeigte ID-Nr. des Textfelds *Transmit ID-No.* entsprechend der ausgewählten Münzsorte geändert.

Zunächst wird entschieden, aus welchem Textfeld der unveränderliche Teil der ID-Nr. entnommen werden soll. Der unveränderliche Teil der ID-Nr. beginnt ab der dritten Ziffer und dient als laufende Serien-Nr. des Hoppers. Dieser Teil kann entweder aus dem Textfeld *Received ID-No.* oder *Transmit ID-No.* entnommen werden. Aus welchem Textfeld der Teil entnommen wird, ist davon abhängig, ob in *Received ID-No.* eine gültige ID-Nr. empfangen wurde. Sollte dies nicht der Fall sein, steht dort der Text *ID-No. not found!* Dies bedeutet zugleich, dass eine ID-Nr. bereits im Textfeld *Transmit ID-No.* stehen wird. In beiden Fällen wird die jeweils eingetragene ID-Nr. ab der dritten Stelle entnommen und zunächst in einer Variable zwischengespeichert. Aus der ausgewählten Münzsorte des Kombinationsfelds kann dann die zweite Ziffer der ID-Nr. entsprechend angepasst werden und der Inhalt des Textfelds *Transmit ID-No.* bekommt die neu zu übertragene ID-Nr. zugewiesen.

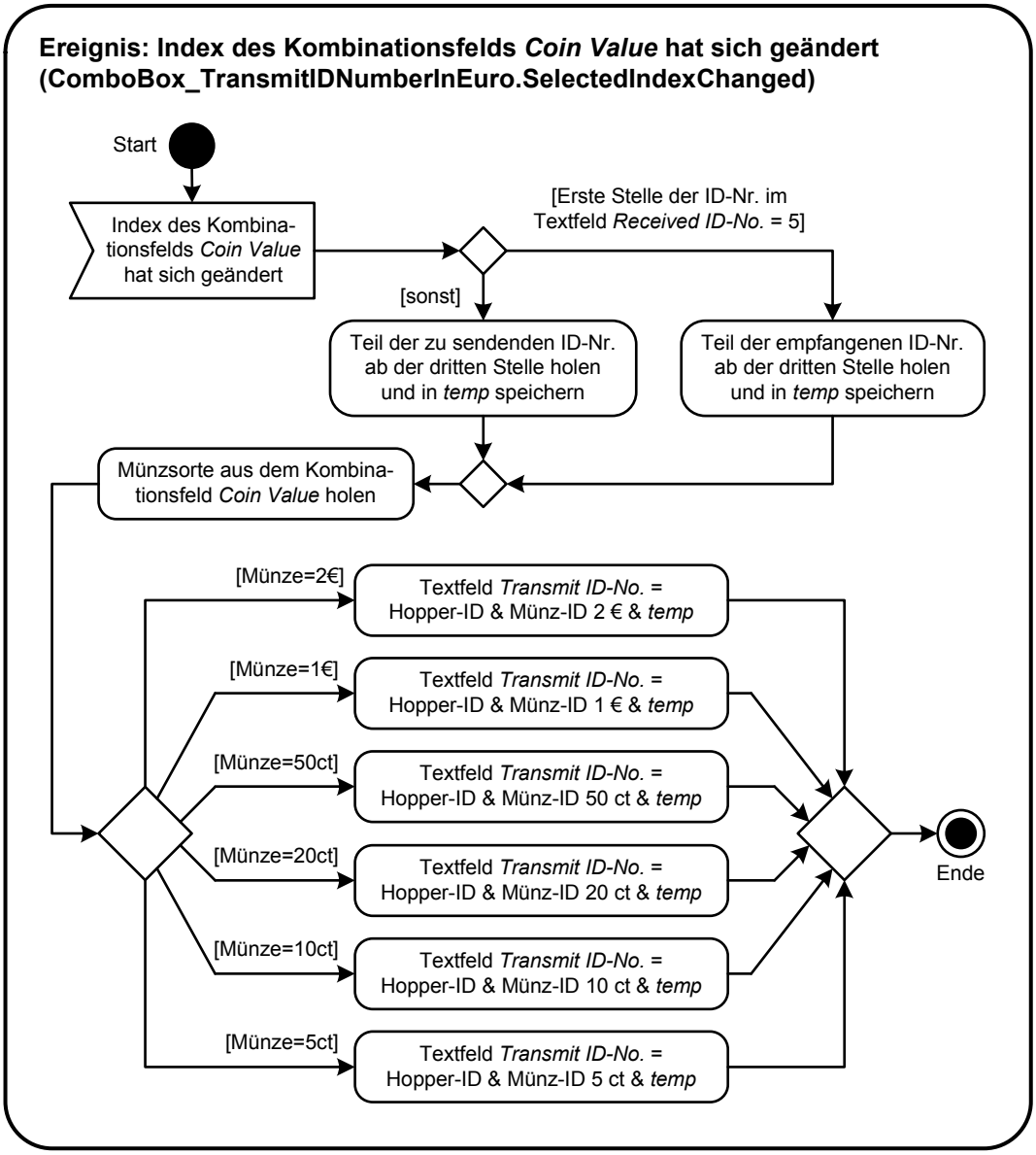

Bild 6.13: Aktivitätsdiagramm *Index des Kombinationsfelds Coin Value hat sich geändert*

6.3.10 Ereignis *Hopper Info betätigt*

Das folgende Diagramm zeigt den Programmablauf nach der Betätigung der Taste **Hopper Info** an. Durch die Betätigung der Taste wird das Signal *Hopper Info betätigt* gesendet, welches in der Klasse *MainWindow* empfangen wird. Die Klasse *MainWindow* sendet seinerseits das Signal *Information des Hoppers holen*, welches von der Klasse *InfoWindow* empfangen wird. Die Klasse *InfoWindow* löscht zunächst alle Inhalte der Textfelder des Fensters und sendet anschließend den ersten Befehl der Befehlskette. Die Befehlskette beinhaltet alle notwendigen Befehle, welche für die Darstellung aller Informationen sowie Status-/ und Fehlermeldungen des Hoppers zuständig sind. Der jeweilige Befehl wird dann aus der Befehlsliste geholt und der Methode *Datenpaket aufbereiten* des Moduls

Transceive übergeben, welches für die Weiterleitung des Befehls an die serielle Schnittstelle zuständig ist.

Wenn der Befehl erfolgreich ausgeführt wird und die Antwort des Hoppers angekommen ist, wird das Modul *Transceive* das Signal *Neue Daten empfangen* senden, welches in der Klasse *InfoWindow* empfangen wird. Nun werden die empfangenen Daten mittels der Aktivität *Daten verarbeiten* weiterverarbeitet. Dies bedeutet, dass die Daten den Textfeldern der jeweiligen Gruppen zugeordnet werden. Nach welchem Schema die Befehle der Befehlskette abgearbeitet werden, kann aus dem nachfolgenden Aktivitätsdiagramm entnommen werden.

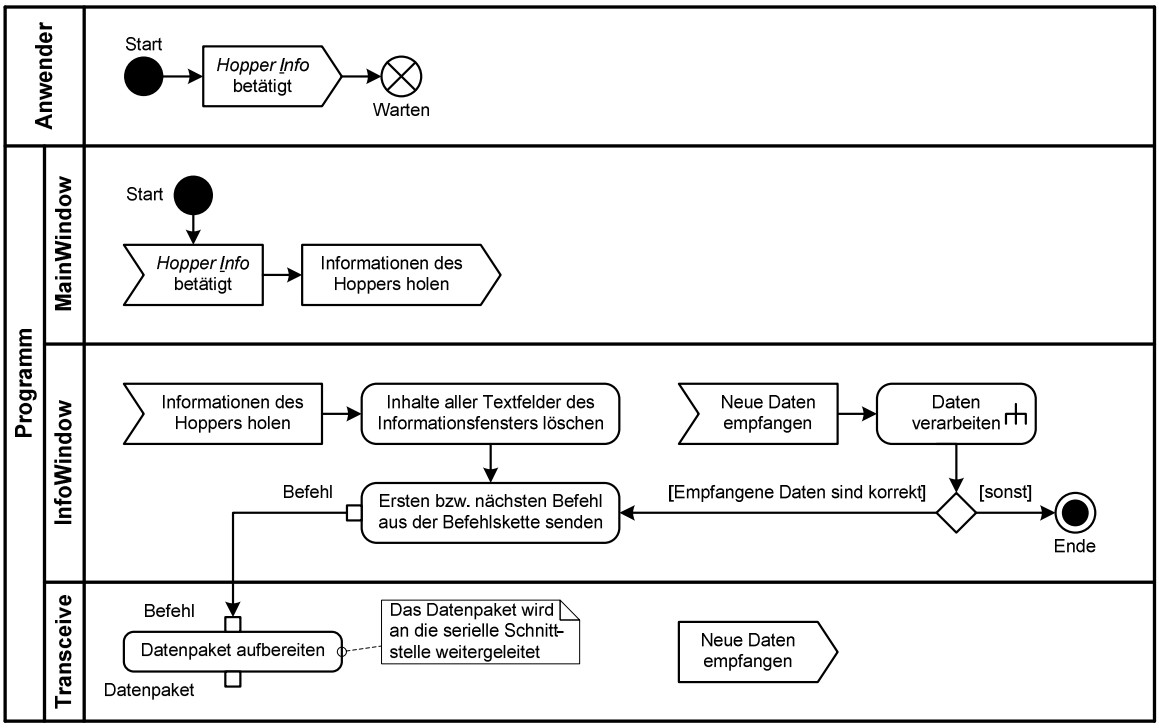

Bild 6.14: Aktivitätsdiagramm zum Ereignis *Hopper Info betätigt*

6.3.11 Aktivität *Daten verarbeiten* der Klasse *InfoWindow*

Die Aktivität *Daten verarbeiten* der Klasse *InfoWindow* besitzt zwei Funktionen. Einerseits werden die empfangenen Daten anhand der Abfrage des zuletzt gesendeten Befehlsnamen in die jeweiligen Textfelder gesetzt. Andererseits gesendet sie den nächsten auszuführenden Befehl.

Der erste Befehl, der aus der Befehlskette gesendet wird, ist der Befehl *SimplePoll*. Er ist erforderlich, um zu prüfen, ob ein Gerät angeschlossen ist. Ist das Gerät angeschlossen, wird dieses mit *ACK* antworten und es kann der nächste Befehl gesendet werden. Ist das Gerät nicht angeschlossen, wird eine Meldung ausgegeben und der Anwender hat

entweder die Möglichkeit einen erneuten Sendeversuch durchzuführen oder den Ablauf zu beenden. Der nächste Befehl lautet *ReqEquipmentCategoryID* um festzustellen, ob es sich beim angeschlossenen Gerät um ein Auszahlungsgerät (Hopper) handelt. Wenn ja, wird der nachfolgende Befehl gesendet. Andernfalls wird ebenfalls eine Meldung ausgegeben und der Anwender hat entweder die Möglichkeit einen erneuten Sendeversuch durchzuführen oder den Ablauf zu beenden. Der weitere Programmablauf bzw. die Reihenfolge, in der die gesendeten Befehle abgearbeitet werden, kann dem Diagramm entnommen werden.

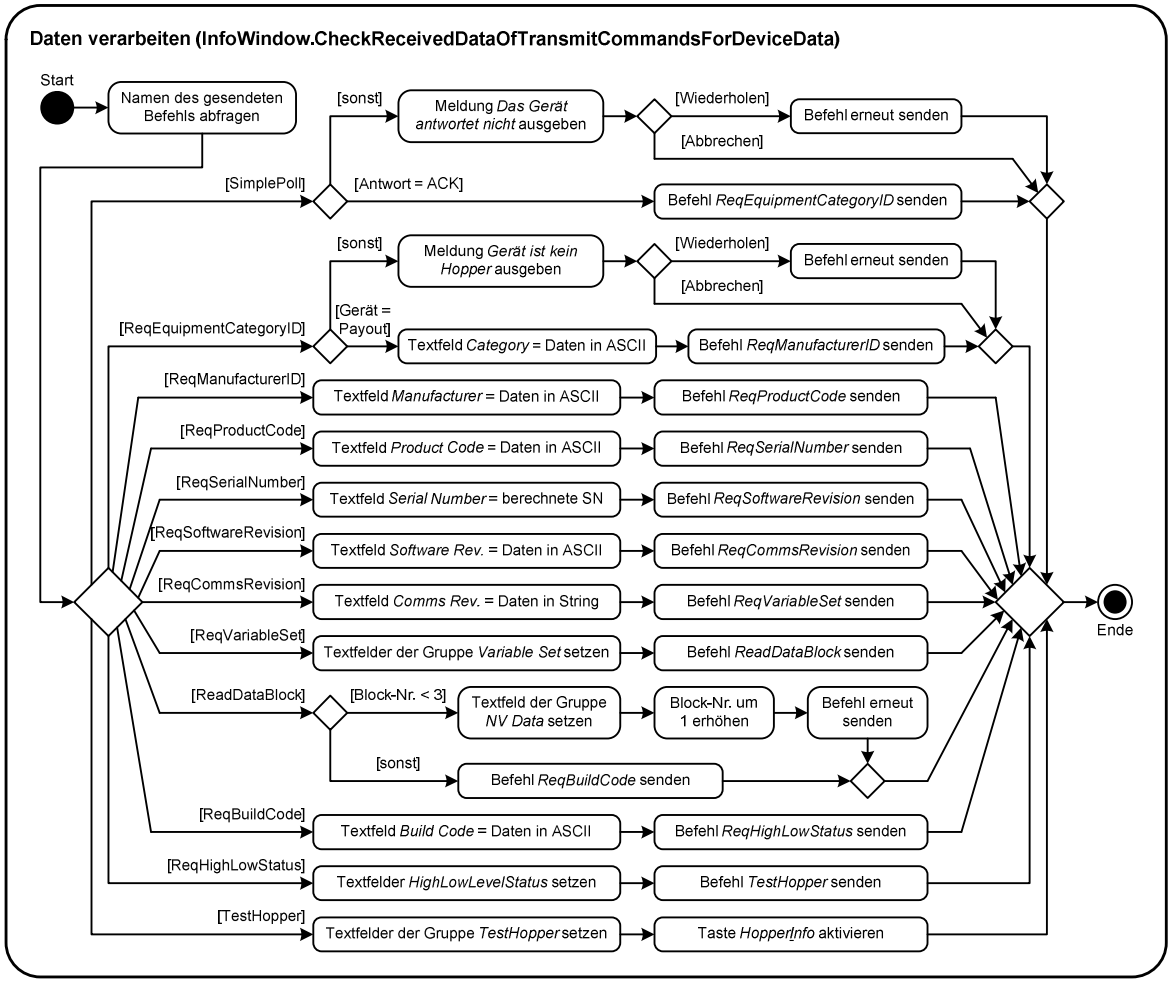

Bild 6.15: Aktivitätsdiagramm der Methode *Daten verarbeiten* der Klasse *InfoWindow*

6.3.12 Ereignis *Taste Dispense betätigt*

Das folgende Diagramm zeigt den Programmablauf nach der Betätigung der Taste **Dispense** an. Der Ablauf ist vergleichbar mit dem der Taste **Hopper Info**.

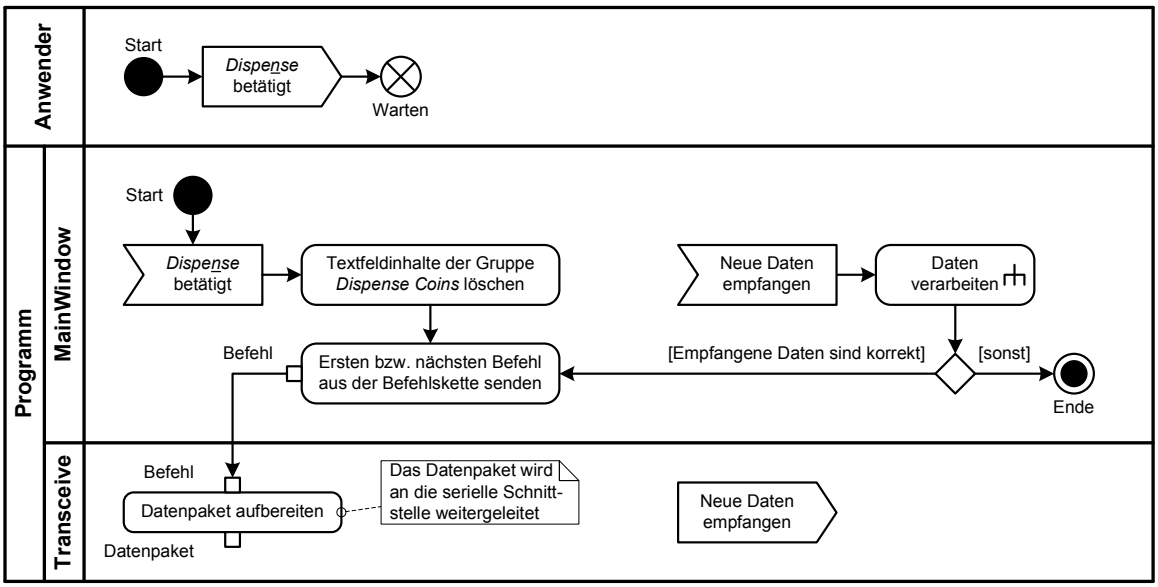

Bild 6.16: Aktivitätsdiagramm zum Ereignis *Dispense betätigt*

Durch die Betätigung der Taste wird das Signal *Dispense betätigt* gesendet, welches in der Klasse *MainWindow* empfangen wird. Die Klasse *MainWindow* löscht zunächst alle Inhalte der Textfelder der Gruppe *Dispense Coins* und sendet anschließend den ersten Befehl der Befehlskette. Der Befehl wird dann aus der Befehlsliste geholt und der Methode *Datenpaket aufbereiten* des Moduls *Transceive* übergeben, welches für die Weiterleitung des Befehls an die serielle Schnittstelle zuständig ist.

Wenn der Befehl erfolgreich ausgeführt wird und die Antwort des Hoppers angekommen ist, wird das Modul *Transceive* das Signal *Neue Daten empfangen* senden, welches in der Klasse *MainWindow* empfangen wird. Nun werden die empfangenen Daten mittels der Aktivität *Daten verarbeiten* weiterverarbeitet. Nach welchem Schema die Befehle der Befehlskette abgearbeitet werden, kann aus dem Aktivitätsdiagramm in Bild 6.17 entnommen werden.

6.3.13 Aktivität *Daten verarbeiten* der Klasse *MainWindow*

Die Aktivität *Daten verarbeiten* der Klasse *MainWindow* ist zuständig für die Verarbeitung der empfangenen Datenbytes eines zuvor gesendeten Befehls. Anhand dessen kann der nächste zu sendende Befehl bestimmt werden. Die Zuordnung der empfangenen Daten zu einem zuvor gesendeten Befehl geschieht durch die Abfrage des Befehlsnamens, der zuletzt gesendet wurde.

Die ersten zwei Befehle sind *SimplePoll* und *ReqEquipmentCategoryID,* welche in der Aktivität *Daten verarbeiten* der Klasse *InfoWindow* beschrieben wurden.

Nachdem nun ermittelt wurde, dass das Gerät angeschlossen wurde und es sich um ein Auszahlungsgerät (Hopper) handelt, wird der Befehl *ReqProductCode* gesendet. Mit diesem Befehl kann festgestellt werden, ob es sich um eine verschlüsselte oder unverschlüsselte Version des Hoppers handelt. Wenn es sich um eine verschlüsselte Version handelt, muss zunächst ein Merker gesetzt werden. Anschließend wird der Befehl *ReqEncryptionCode* gesendet, um den Schlüssel vom Hopper zu erhalten.

Um evtl. vorherige Status-/Fehlermeldungen des Hoppers, welche eine Auszahlung verhindern könnten, zu quittieren, wird der Befehl *ResetDevice* gesendet. Da das Gerät nun mittels des Befehls *ResetDevice* zurückgesetzt wurde, muss der Hopper zunächst mittels des Befehls *EnableHopper* aktiviert werden, um eine Auszahlung zu ermöglichen. Nachdem der Hopper aktiviert wurde, werden mittels des Befehls *ReqCipherKey* die verschlüsselten Bytes des Hoppers abgefragt. Dieser Befehl muss unabhängig von der Version des Hoppers (verschlüsselt oder unverschlüsselt) gesendet werden. Dies wurde vom Hopper-Hersteller beabsichtigt, damit der Programmablauf zur Auszahlung unabhängig von der Version des Hoppers gleich bleibt. Nun wird mittels des Merkers der Verschlüsselung entschieden, ob eine Verschlüsselung stattfinden soll oder nicht. Liegt eine unverschlüsselte Version des Hoppers vor, wird direkt nach Ankunft der empfangenen Daten des Befehls *ReqCipherKey* der Befehl *DispenseHopperCoins* gesendet. Liegt eine verschlüsselte Version des Hoppers vor, werden die empfangenen Daten (verschlüsselte Bytes) des Befehls *ReqCipherKey* zunächst entschlüsselt. Anschließend findet eine byteweise XOR-Verknüpfung zwischen den entschlüsselten Daten und der angegebenen Münzanzahl, die ausgegeben werden soll, statt. Die sich aus der XOR-Verknüpfung ergebenen Bytes werden verschlüsselt und mittels des Befehls *DispenseHopperCoins* an den Hopper gesendet. Der Hopper beginnt unmittelbar nach dem Empfang der Daten mit der Auszahlung der Münzen.

Nun wird der Hopper alle 200 ms mittels des Befehls *ReqHopperStatus* nach dem Status der Münzausgabe abgefragt. Dies bedeutet, dass die Anzahl der ausgegebenen und noch auszugebenen Münzen in den jeweiligen Textfeldern angezeigt werden.

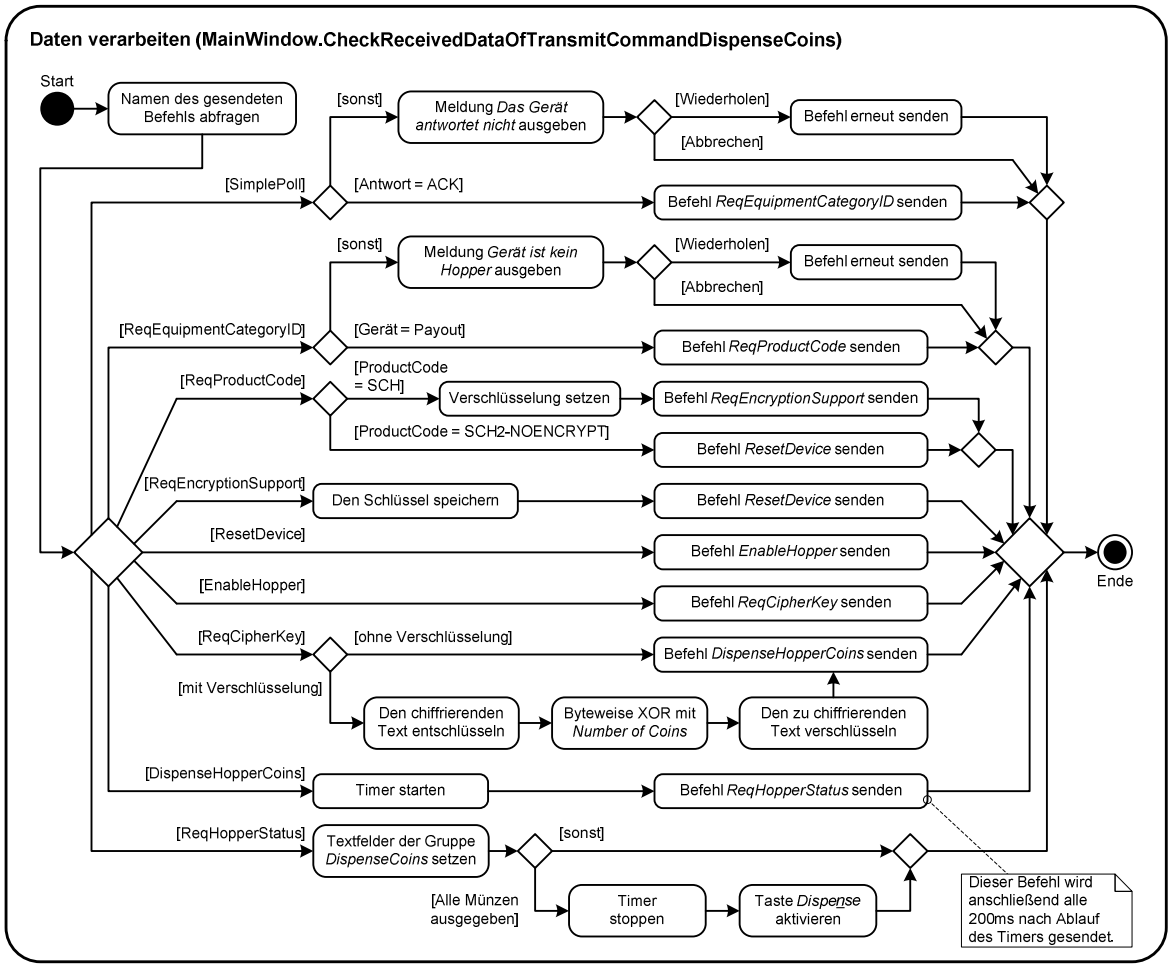

Bild 6.17: Aktivitätsdiagramm der Methode *Daten verarbeiten* der Klasse *MainWindow*

6.3.14 Ereignis *Startup* der Klasse *MyApplication*

Das Ereignis *Startup* wird ausgelöst während die Anwendung startet und zwar noch bevor das Hauptfenster der Anwendung angezeigt wird. Beim Start wird geprüft, ob serielle Schnittstellen am Rechner vorhanden sind. Wird keine serielle Schnittstelle gefunden, erscheint ein Hinweis und der Anwender hat zwei Möglichkeiten. Entweder kann eine serielle Schnittstelle angeschlossen und anschließend die Taste *Wiederholen* betätigt werden, damit die Anwendung gestartet wird. Der Anwender kann auch die Taste *Abbrechen* betätigen, um die Anwendung zu beenden. Wenn serielle Schnittstellen an Rechner vorhanden sind, werden diese im Kombinationsfeld der Gruppe *COM Port* aufgelistet.

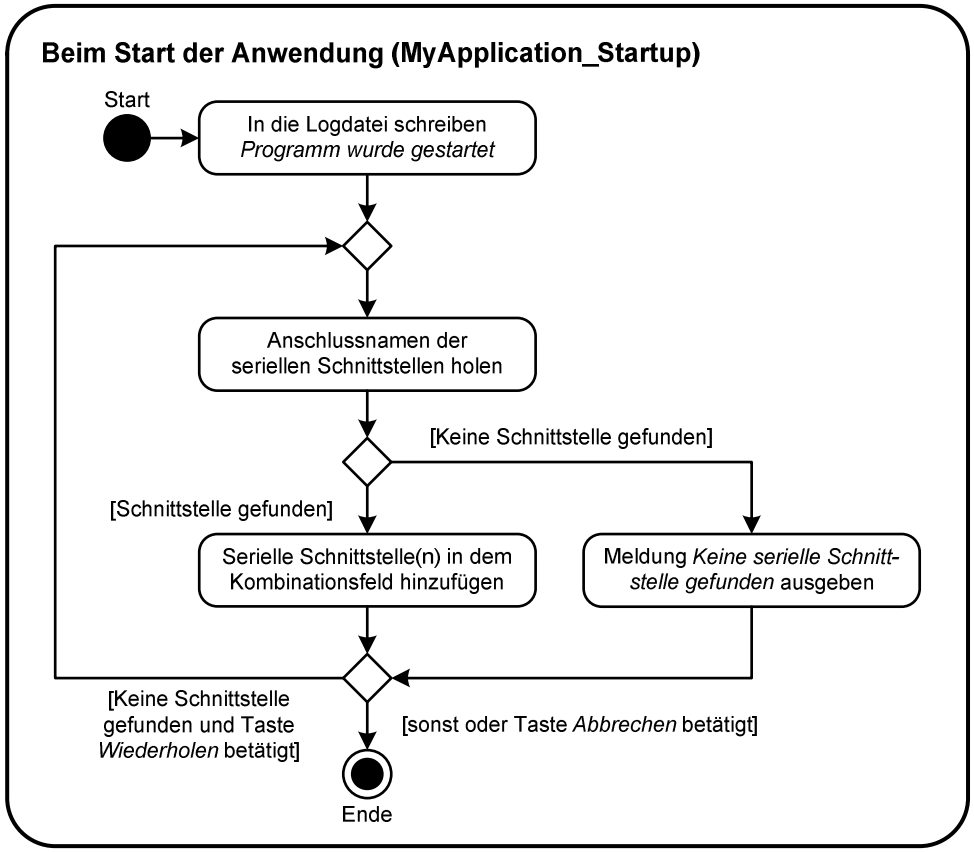

Bild 6.18: Aktivitätsdiagramm zum Ereignis *Startup* der Klasse *MyApplication*

6.3.15 Betriebssystemmeldungen abfangen

In der heutigen Zeit besitzen die Rechner i. d. R. keine RS-232 serielle Schnittstelle mehr. Dafür gibt es die sogenannten USB-zu-Seriell-Adapter, mit denen dem Rechner ein virtueller COM-Port zur Verfügung gestellt werden kann. Dies hat zur Folge, dass weitere Fehlerquellen hinzukommen, welche vom Programm abgefangen werden müssen. Im Gegensatz zu einer am Rechner vorhandenen seriellen Schnittstelle bspw. kann der USB-Adapter physikalisch entfernt oder sogar wieder hinzugefügt werden. Das Windows Betriebssystem erkennt diese Aktionen und leitet diese in Form von Meldungen an alle zzt. laufenden Anwendungen weiter.

Das Programm empfängt die Meldungen des Betriebssystems und prüft, ob es sich dabei um die Meldung, dass Änderungen an der Hardwarekonfiguration des Rechners aufgetreten sind, handelt. In diesem Fall wird die Identifikationsstruktur des veränderten Geräts geholt, aus der dann abgefragt wird, ob es sich um den Gerätetyp COM Port handelt. Wenn dies der Fall ist, wird zunächst der Anschlussname des COM Ports geholt, da auch mehrere serielle Schnittstellen am Rechner vorhanden sein können. In Abhängigkeit des ausgelösten Ereignisses wird entweder der COM Port anhand des Anschlussnamens aus

dem Kombinationsfeld der Gruppe *COM Port* entfernt oder hinzugefügt. Sollte es sich um einen aus der Anwendung heraus geöffneten COM Port handeln, der entfernt wurde, wird dieser geschlossen.

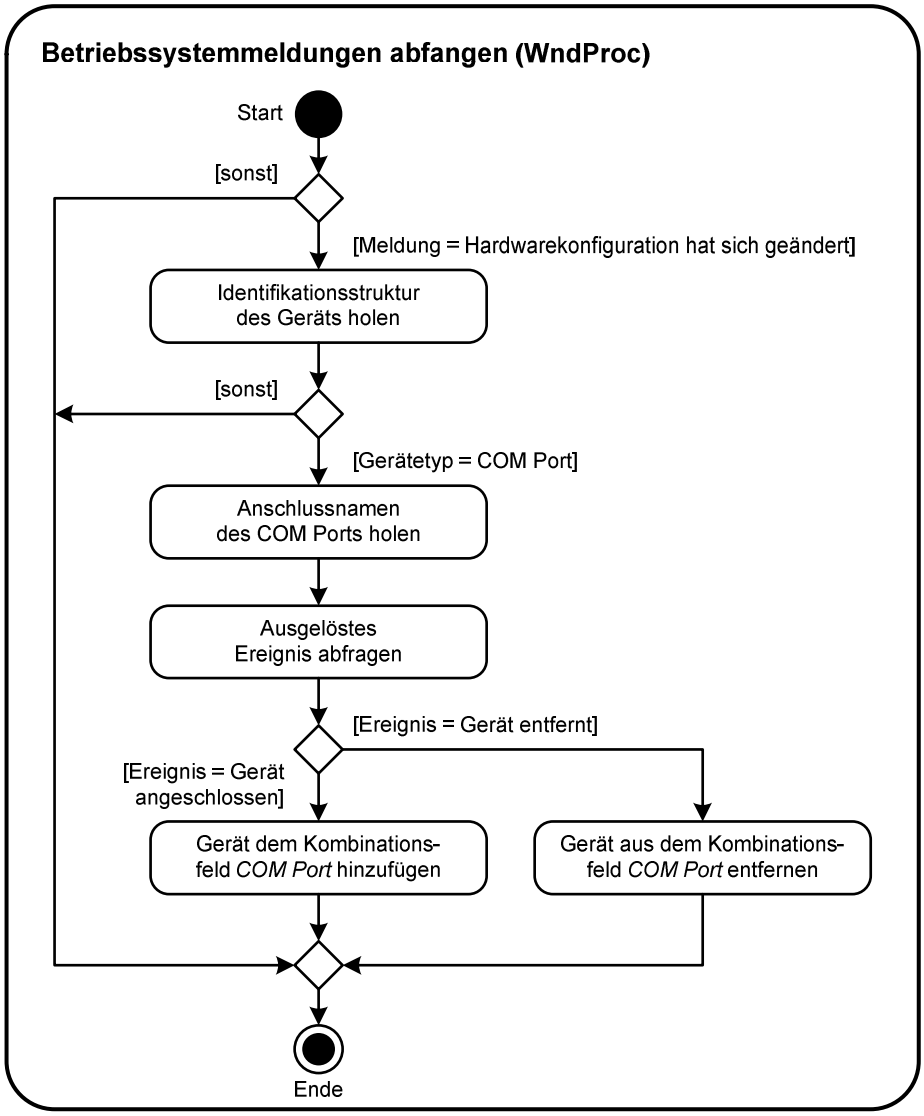

Bild 6.19: Aktivitätsdiagramm zum Abfangen der Betriebssystemmeldungen

7 Testphase

In diesem Abschnitt werden nur die wesentlichen Kernelemente beschrieben.

Die Testsoftware wurde auf zwei Rechnern getestet. Ein Rechner ist mit dem Betriebssystem Windows XP und der andere mit Windows 7 ausgestattet. Bei beiden Rechnern sind keine seriellen RS-232 Anschlüsse vorhanden. Als serielle Schnittstelle wurde der USB-zu-Seriell-Adapter UC232A der Fa. Aten verwendet. Für den Adapter wird ein Treiber benötigt. Dieser kann für das jeweilige Betriebssystem von der Webseite http://de.aten.eu/products/productItem.php?model_no=UC232A des Herstellers heruntergeladen werden.

Es konnten alle Funktionen der Testsoftware mit Ausnahme der Münzausgabefunktion einer verschlüsselten Version des Hoppers erfolgreich geprüft werden. Die verschlüsselte Version des Hoppers lag nicht vor. Die Verschlüsselung konnte jedoch anhand einer Simulation innerhalb der Software getestet werden. Die für die Simulation benötigten Daten wurden dem Beispiel DES-Verschlüsselung aus dem Kapitel 3.5.3 entnommen. Das Ergebnis der Simulation war, dass die Münzausgabe einer verschlüsselten Version des Hoppers ebenfalls mit der Testsoftware geprüft werden kann.

Eine wichtige Erkenntnis ist, dass bei einer aus der Testsoftware geöffneten seriellen Schnittstelle durch das Abziehen des USB-zu-Seriell-Adapters die Rechenleistung stark beeinflusst wurde. Es hat eine sukzessive Erhöhung der Prozessorauslastung und des Arbeitsspeichers stattgefunden bis anschließend ein sogenannter Blue Screen[9] erfolgt ist. Das Problem bestand darin, dass beim Abziehen des USB-zu-Seriell-Adapters der COM-Port offen blieb. Dieses Problem wurde behoben, indem die Betriebssystemmeldung über Änderungen an der Hardwarekonfiguration des Rechners von der Testsoftware abgefangen wird. Es wird dann das im Kapitel 6.3.15 beschriebene Verfahren angewendet. Sollte es sich um einen aus der Anwendung heraus geöffneten COM-Port handeln, der entfernt wurde, wird dieser geschlossen.

[9] Beim Blue Screen wird die Bedienoberfläche des Betriebssystems durch einen blauen Bildschirm ersetzt, auf dem die Fehlerinformationen in weißer Schrift erscheinen.

8 Schlussbetrachtung

8.1 Zusammenfassung

Die vorliegende Arbeit hat sich mit der Entwicklung einer Testsoftware zur Ansteuerung des Zusatzrestgeldspeichers *Serial Compact Hopper MK2* mittels des seriellen Kommunikationsprotokolls *ccTalk* befasst.

Zu Beginn der Arbeit wurde zunächst ein Einblick in die Funktionsweise der Bargeldverarbeitung des Fahrausweisautomaten *almex.compact* der Fa. Höft & Wessel AG gewährt. Dabei wurden die Komponenten der Bargeldverarbeitung beschrieben und deren Zusammenspiel schematisch dargestellt. Die Komponente, für die die Testsoftware geschrieben wurde, ist der Zusatzrestgeldspeicher (Hopper) *Serial Compact Hopper MK2* der Fa. Money Controls™. Der Hopper ist für die Wechselgeldausgabe zuständig.

Der *Serial Compact Hopper MK2* wurde im Kapitel 2 beschrieben. Dort befindet sich auch eine ausführliche Beschreibung zur Funktionsweise der Münzausgabe. Anschließend folgt eine Beschreibung des Kommunikationsprotokolls *ccTalk*, mit dem der Hopper anhand von Befehlen angesteuert werden kann.

Den *Serial Compact Hopper MK2* gibt es in den Ausführungen mit und ohne Verschlüsselung. Bei dem Hopper mit Verschlüsselung wird der Befehl zur Münzausgabe mittels des DES-Verschlüsselungsalgorithmus gesendet. Die Funktionsweise des Algorithmus wurde im Kapitel 3.5 beschrieben und anhand eines Beispiels verdeutlicht.

Vor der Entwicklung der eigenen Testsoftware wurde eine Marktanalyse durchgeführt. Es wurde festgestellt, dass eine Testsoftware der Fa. Money Controls™ existiert, welche im Kapitel 5 vorgestellt wurde. Da die bereits existierende Software die Anforderungen der Aufgabenstellung nicht erfüllen konnte, wurde eine eigene Software implementiert. Die Entwicklung bzw. das Design der entwickelten Software wurde mittels UML Aktivitätsdiagrammen modelliert und im Kapitel 6 ausführlich vorgestellt.

8.2 Fazit

Die Testsoftware erfüllt die Anforderungen der Aufgabenstellung. Sie kann die vom Automatenhersteller Höft & Wessel AG eingespeicherte ID-Nr. im *Serial Compact Hopper MK2* auslesen und verändern. Die Münzausgabefunktion des Hoppers kann geprüft werden. Durch die Statusmeldungen der Sensoren über den Füllstand des Hoppers können die Sensoren auf Funktion geprüft werden. Mit dem Auslesen der Gesamtanzahl der bis dato ausgegebenen Münzen werden, im Falle des Überschreitens bestimmter Schwellenwerte, Hinweise zur Durchführung von Wartungs-/Instandsetzungsarbeiten am Hopper ausgegeben.

Aufgrund der integrierten Hilfefunktion nebst Beispielen wird der Einstieg in die Bedienung der Anwendung erleichtert. Die Anwendung kann auch als Analysator für das *ccTalk*-Protokoll betrachtet werden. Durch den festen Rahmenaufbau der Datenpakete kann die Befehlsliste beliebig erweitert werden. Befehle können einfach hinzugefügt werden.

Durch die Modellierung der Software mittels UML konnten die benötigten Schnittstellen der Anwendung präzise festgelegt werden, sodass die anschließende Implementierung des Quellcodes erfolgreich durchgeführt werden konnte.

8.3 Ausblick

Die Software wird auch nach Abgabe dieser Arbeit gepflegt und ggf. erweitert. Einige Anregungen zur Erweiterung der Testsoftware werden hier vorgestellt.

Die *ccTalk*-Analysefunktion und die Testfunktionen für den Hopper könnten voneinander getrennt werden. Die Bedienelemente zur Ansteuerung des Hoppers könnten dann z. B. in einer separaten Registerkarte platziert werden.

Des Weiteren könnte der Umgang mit den *ccTalk*-Befehlen in Form einer lokalen Datenbank umgesetzt werden. Es würde sich anbieten, Befehle in Gruppen zusammenzufassen. Die in der Testsoftware vorhandene Befehlsliste könnte in Grundbefehle, welche für alle *ccTalk*-Geräte gelten und in Befehle für den Hopper aufgeteilt werden. Weitere Gruppen, z. B. Befehle für einen Münzprüfer, könnten hinzugefügt werden. Eine Benutzerschnittstelle zwischen Datenbank und Anwender ließe sich als Menüelement in die Anwendung integrieren. Auf diese Weise würde dem Anwender die Möglichkeit angeboten werden mittels einer Eingabemaske auf die Datenbank zuzugreifen, um Befehle hinzuzufügen, zu ändern oder zu entfernen.

Dadurch, dass die Anwendung modular programmiert wurde, können verschiedene Bargeldverarbeitungsgeräte in die Anwendung integriert werden. Voraussetzung hierfür ist, dass die Geräte das serielle Kommunikationsprotokoll *ccTalk* unterstützen. Mögliche Bargeldverarbeitungsgeräte könnten z. B. Münzprüfer oder Banknotenverarbeitung sein. Die Erweiterung der Testsoftware zur Prüfung von weiteren Bargeldverarbeitungsgeräten könnte im Rahmen einer weiteren Arbeit untersucht werden.

9 Danksagung

Mit dieser Arbeit wird das Studium Informations- und Elektrotechnik abgeschlossen und deshalb möchte ich mich an dieser Stelle bei ausgewählten Menschen bedanken.

Zunächst möchte ich mich bei meinem Betreuer Herrn Prof. Dr. Robert Fitz bedanken, der die Betreuung der Arbeit herzlich angenommen hat, mir mit konstruktiven Vorschlägen immer zur Seite stand und mit seiner hohen Hilfsbereitschaft und Engagement mich sehr unterstützt hat. Ebenfalls möchte ich mich auch bei Herrn Prof. Dr. Robert Heß bedanken, der sich bereit erklärt hat, die Aufgabe des Zweitgutachters zu übernehmen.

Einen großen Dank richte ich auch an die Firma HOCHBAHN AG, die mir das Studium ermöglicht und mich finanziell unterstützt hat. Insbesondere an die Mitarbeiter der Abteilung Zugsicherungs- und Kommunikationsanlagen, die mir bei der Entstehung der Arbeit geholfen haben.

Dass das Studium erfolgreich abgeschlossen werden konnte, habe ich den Professor/-innen und Kommilitonen zu verdanken. Insbesondere möchte ich mich bei den Kommilitonen Phillip Durdaut, Michael Meinzer und Holger Ullrich bedanken, mit denen ich den Großteil der Laborübungen in einer sehr angenehmen Arbeitsatmosphäre durchgeführt habe. Weiterhin möchte ich mich bei den Kommilitonen Florian Krause und Tobias Köster für das Korrekturlesen dieser Arbeit bedanken.

Ein besonderer Dank gilt meiner Freundin Annika Goldmann, die immer an mich geglaubt hat und mich auch in dieser Phase mit unendlicher Geduld und liebevoller Fürsorge auf dem richtigen Weg gehalten hat.

Literaturverzeichnis

[1] Crane Payment Solutions. http://www.craneps.com/en/brands/view/4/Money+Controls, Abruf: 30.04.2013

[2] Ertel, Wolfgang. *Angewandte Kryptographie.* München : Carl Hanser Verlag, 2007. ISBN 978-3-446-41195-1

[3] Höft & Wessel AG, Flege Dirk: *Pflichtenheft HHAG1.01*. 13.02.2006

[4] Kainka, Burkhard und Berndt, Hans-Joachim. *PC-Schnittstellen unter Windows - Messen, Steuern und Regeln über die Standard-Ports*. 5. Auflage. Aachen : Elektor-Verlag, 2003. ISBN 3-89576-086-2

[5] Kecher, Christoph. *UML 2 Das umfassende Handbuch*. 4., aktualisierte und erweiterte Auflage. Bonn : Galileo Press, 2011. ISBN 978-3-8362-1752-1

[6] Maxim Integrated. *MAX200–MAX209/MAX211/MAX213 +5V, RS-232 Transceivers with 0.1µF External Capacitors.* http://datasheets.maximintegrated.com/en/ds/MAX200-MAX213.pdf, Abruf: 10.05.2013

[7] Maxim Integrated. *MAX220–MAX249 +5V-Powered, Multichannel RS-232 Drivers/Receivers*. http://datasheets.maximintegrated.com/en/ds/MAX220-MAX249.pdf, Abruf: 10.05.2013

[8] Messe Berlin GmbH - Innotrans. http://www.virtualmarket.innotrans.de/index.php5?id=1068325&compact=0&Action=showProduct, Abruf: 27.04.2013

[9] Money Controls GmbH. *Testsoftware für Serielle Hopper*. http://www.money-controls.de/manuals/prog_vorauswahl.htm, Abruf: 03.05.2013

[10] Money Controls™. 2010. *ccTalk Serial Communication Protocol - Generic Specification - Issue 4.6 - Part 1*. http://www.craneps.com/en/products/view/151, Abruf: 11.05.2013

[11] Money Controls™. 2010. *ccTalk Serial Communication Protocol - Generic Specification - Issue 4.6 - Part 2*. http://www.craneps.com/en/products/view/151, Abruf: 11.05.2013

[12] Money Controls™. 2010. *ccTalk Serial Communication Protocol - Generic Specification - Issue 4.6 - Part 3*. http://www.craneps.com/en/products/view/151, Abruf: 11.05.2013

[13] Money Controls™. 2011. *DES Encryption for Hoppers - ccTalk Protocol - Issue 2.1 - 28th January 2011*. http://www.craneps.com/en/products/view/151, Abruf: 11.05.2013

[14] Money Controls™. 2011. *Serial Compact Hopper 2 Technical Manual - Issue 3.1*. http://www.moneycontrols.com/en/users/login/, Abruf: 11.05.2013

[15] Sauter, Benedikt. *Messen, Steuern und Regeln mit USB*. Poing : Franzis Verlag, 2010. ISBN 978-3-7723-5878-4

[16] Schenk, Joachim und Rigoll, Gerhard. *Mensch-Maschine-Kommunikation Grundlagen von sprach- und bildbasierten Benutzerschnittstellen*. Berlin Heidelberg : Springer-Verlag, 2010. ISBN 978-3-642-05456-3

[17] Wittgruber, Friedrich. *Digitale Schnittstellen und Bussysteme - Einführung für das technische Studium*. 2., überarbeitete und erweiterte Auflage. Braunschweig/ Wiesbaden : Vieweg-Verlag, 2002. ISBN 3-528-17436-6

Anhang

A Bedienungsanleitung

A.1 Überblick

Die Anwendung ***Testsoftware for Serial Compact Hopper MK2*** dient zur Ansteuerung des Zusatzrestgeldspeichers *Serial Compact Hopper MK2*, nachfolgend Hopper genannt, der Fa. Money Controls™.

Der Hopper befindet sich im Fahrausweisautomaten *almex.compact* der Fa. Höft & Wessel AG, im Folgenden Automat genannt, und ist für die Auszahlung von Wechselgeld bzw. Restgeld zuständig. Ein Hopper ist nur mit einer bestimmten Münzsorte gefüllt. Damit die Automatensoftware erkennen kann, für welche Münzwertigkeit der Hopper zuständig ist, wird dieser vom Automatenhersteller mit einer ID-Nr. versehen.

Die Anwendung hat die folgenden zwei Hauptaufgaben:

1. Die eingespeicherte ID-Nr. des Automatenherstellers auslesen und ggf. verändern.
2. Münzausgabefunktion des Hoppers prüfen.

A.2 Bedienoberfläche

Die Anwendung ***Testsoftware for Serial Compact Hopper MK2*** besteht aus den zwei Fenstern:

1. Hauptfenster (Main Window)
2. Informationsfenster (Info Window)

A.2.1 Hauptfenster (Main Window)

Das Hauptfenster beinhaltet alle benötigten Steuerelemente zur Ansteuerung des Hoppers.

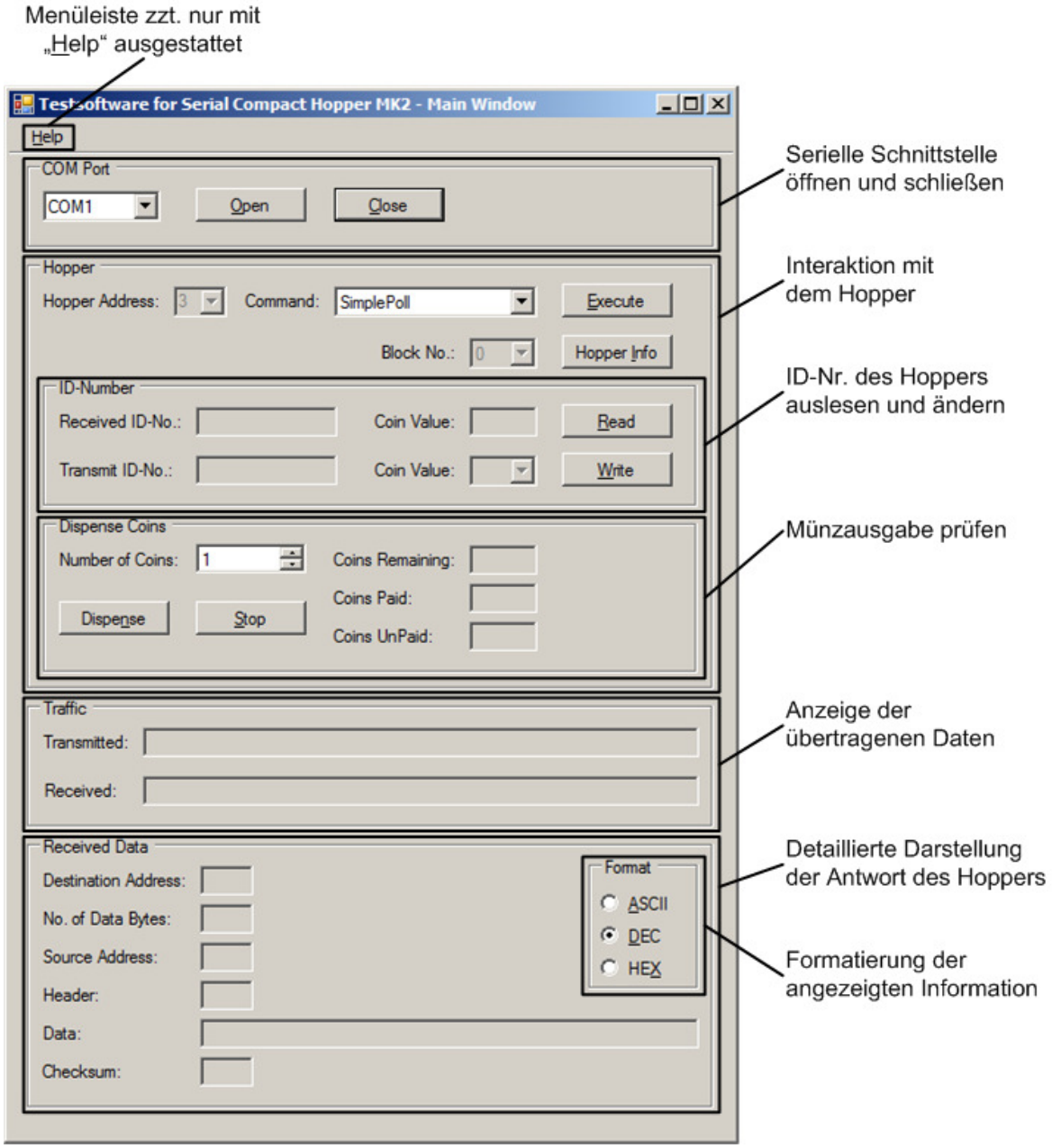

Bild A.1: Hauptfenster der Testsoftware mit allgemeiner Funktionsbeschreibung

Das Hauptfenster besteht im Wesentlichen aus den vier Gruppen:

1. COM Port
2. Hopper
    a. ID-Number
    b. Dispense Coins
3. Traffic
4. Received Data
    a. Format

Hinweis: Die Gruppen *Hopper*, *Traffic* und *Received Data* und deren Untergruppen sind zunächst deaktiviert und werden erst aktiviert, nachdem eine serielle Schnittstelle in der Gruppe *COM Port* geöffnet wurde.

A.2.2 Informationsfenster (Info Window)

Das Informationsfenster dient zur Anzeige von Informationen sowie von Status-/Fehlermeldungen des Hoppers.

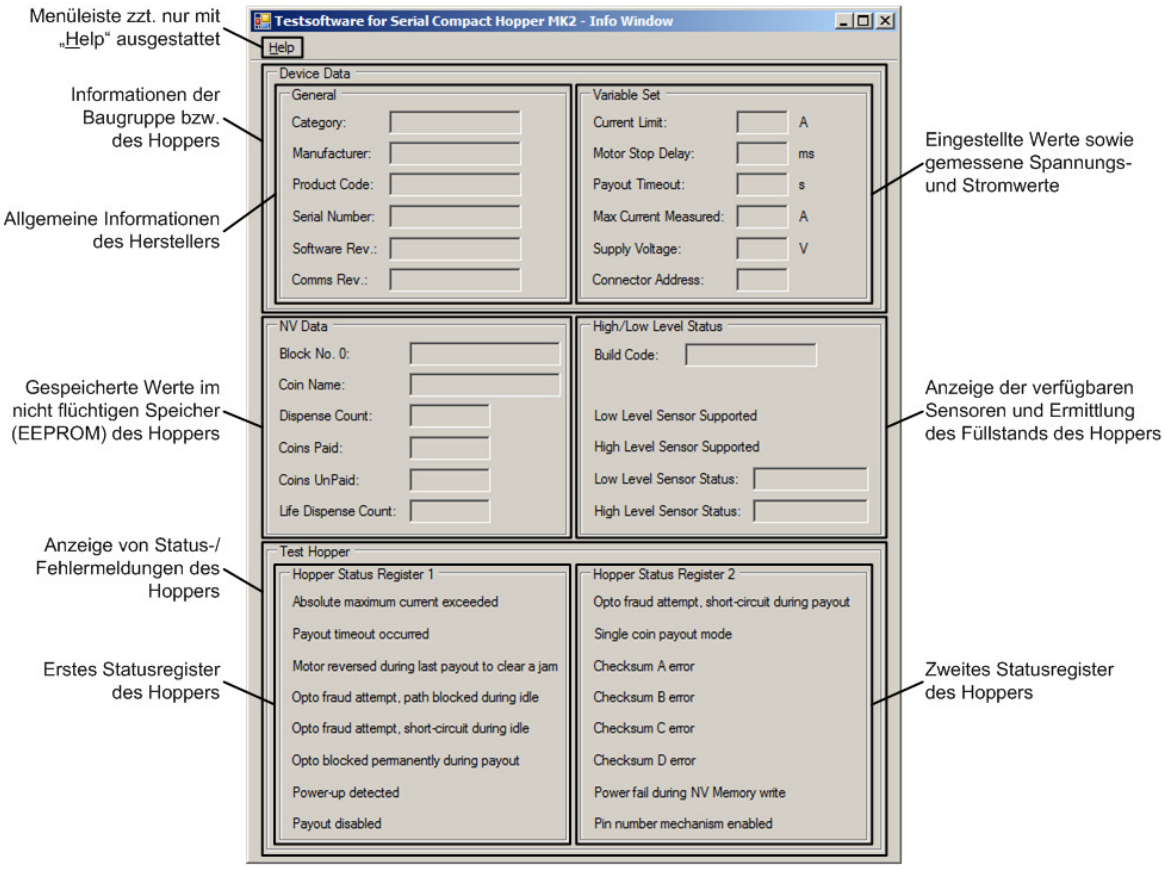

Bild A.2: Informationsfenster der Testsoftware mit allgemeiner Funktionsbeschreibung

Das Informationsfenster besteht aus den vier Gruppen mit ihren jeweiligen Untergruppen:

1. Device Data
    a. General
    b. Variable Set
2. NV (Non Volatile) Data
3. High/Low Level Status
4. Test Hopper
    a. Hopper Status Register 1
    b. Hopper Status Register 2

A.3 Allgemeine Informationen

A.3.1 Tastenkürzel

Das Programm kann entweder mit der Maus oder mit der Tastatur bedient werden. Für eine erleichterte Bedienung über die Tastatur ist jeweils der unterstrichene Buchstabe des Tastenbezeichners maßgeblich. Sollten die Unterstriche nicht zu sehen sein, muss die Taste *Alt* betätigt werden. Es ist **nicht** erforderlich, die Taste *Alt* gedrückt zu halten, da es sich nicht um eine Tastenkombination wie z. B. *Strg+c* handelt.

Tabelle A.1: Tastenkürzel und deren Funktionsbeschreibung

Tastenkürzel	Funktionsbeschreibung
A	Ändert die Darstellungsform der empfangenen bzw. angezeigten Daten der Gruppe *Received Data* in ASCII Zeichen um.
C	Schließt die serielle Schnittstelle.
D	Ändert die Darstellungsform der empfangenen bzw. angezeigten Daten der Gruppe *Received Data* in Dezimal um.
E	Führt den aus dem Kombinationsfeld Command der Gruppe *Hopper* ausgewählten Befehl aus.
H	Ruft die allgemeine Hilfe des Programms auf.
I	Führt eine Reihe an Befehlen aus, um die Informationen sowie die Status-/Fehlermeldungen des Hoppers anzuzeigen. Es wird das Informationsfenster (Info Window) geöffnet, in dem alle Daten angezeigt werden. Bei einer erneuten Betätigung der Taste werden die angezeigten Inhalte gelöscht und anschließend die neuen Daten angezeigt.
N	Führt den Befehl zur Münzausgabe aus.
O	Öffnet die serielle Schnittstelle.
R	Liest die im Hopper eingespeicherte ID-Nr. des Automatenherstellers aus.
S	Führt den Befehl zur sofortigen Abschaltung der Münzausgabe aus.
W	Schreibt die neue ID-Nr. in den Hopper.
X	Ändert die Darstellungsform der empfangenen bzw. angezeigten Daten der Gruppe *Received Data* in Hexadezimal um.

Hinweis: Bei den Tastenkürzeln wird zwischen Groß- und Kleinschreibung **nicht** unterschieden.

A.4 Gruppen des Hauptfensters (Main Window)

A.4.1 COM Port

Die Gruppe *COM Port* dient zur Interaktion mit einer oder mehreren am Rechner vorhandenen seriellen Schnittstellen. Die vorhandenen Schnittstellen werden im Kombinationsfeld der Gruppe aufgelistet.

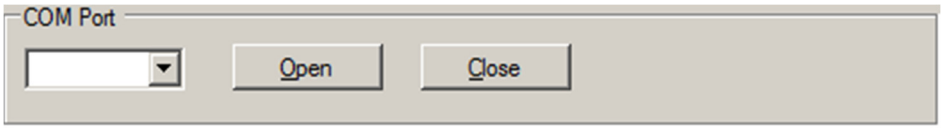

Bild A.3: Gruppe *COM Port* - Ausschnitt aus dem Hauptfenster

Mit der Taste **Open** wird die im Kombinationsfeld ausgewählte serielle Schnittstelle geöffnet. Mit der Taste **Close** wird eine geöffnete serielle Schnittstelle geschlossen.

Wird eine serielle Schnittstelle erfolgreich geöffnet, werden das Kombinationsfeld und die Taste **Open** deaktiviert. Zusätzlich werden alle anderen Gruppen des Hauptfensters aktiviert. Dadurch, dass die Taste **Open** deaktiviert wird, kann zunächst keine weitere Schnittstelle geöffnet werden. Um eine andere serielle Schnittstelle zu öffnen, muss die bereits geöffnete Schnittstelle vorher geschlossen werden.

Die Konfigurations- bzw. Protokollparameter der seriellen Schnittstelle sind durch das Kommunikationsprotokoll *ccTalk* vorgegeben. Die erforderlichen Einstellungen erfolgen automatisch im Programm und müssen nicht manuell vorgenommen werden. Die Protokollparameter lauten: 8N1 (8 Datenbits, keine Parität, 1 Stoppbit), Bitrate: 9600 Bits/s und keine Flusssteuerung.

A.4.1.1 Beispiel: Serielle Schnittstelle öffnen

Die Testsoftware hat die serielle Schnittstelle COM 1 erkannt und zeigt diese im Kombinationsfeld an. Es wurde noch keine Schnittstelle geöffnet und deshalb ist die Taste **Close** deaktiviert. Um die Schnittstelle zu öffnen kann entweder die Taste **Open** mit der Maus oder die Taste *o* von der Tastatur aus betätigt werden.

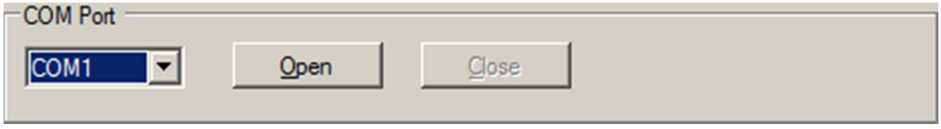

Bild A.4: Gruppe *COM Port* mit der erkannten seriellen Schnittstelle COM 1

Nachdem die serielle Schnittstelle COM 1 erfolgreich geöffnet wurde, werden das Kombinationsfeld und die Taste **Open** deaktiviert. Es kann nur noch die Taste **Close** betätigt werden, um die bereits geöffnete Schnittstelle COM 1 zu schließen.

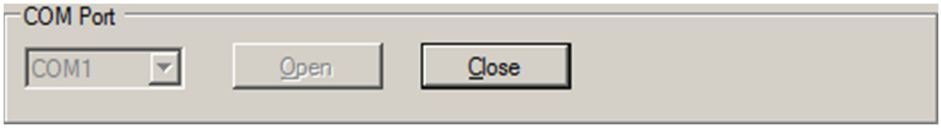

Bild A.5: Gruppe *COM Port* mit der geöffneten seriellen Schnittstelle COM 1

A.4.1.2 Informations-/Fehlermeldungen

Sollte eine serielle Schnittstelle physikalisch entfernt werden, z. B. durch das Abziehen eines USB-zu-Seriell-Adapters, erscheint folgende Meldung:

Bild A.6: Fehlermeldung - serielle Schnittstelle COM 1 wurde entfernt

In der Fehlermeldung wird der Name der seriellen Schnittstelle, hier z. B. COM 1, angezeigt. Dabei wird die serielle Schnittstelle aus dem Kombinationsfeld entfernt. Handelt es sich um eine geöffnete Schnittstelle, werden die Gruppen *Hopper*, *Traffic* und *Received Data* und deren Untergruppen deaktiviert. Wird die Schnittstelle wieder hinzugefügt, muss sie erst wieder geöffnet werden.

Hinweis: Eine nicht mehr vorhandene serielle Schnittstelle am Rechner darf nicht mehr als Objekt im Speicher existieren. Deshalb wird diese entfernt. Die, z. B. durch das Abziehen des USB-zu-Seriell-Adapters, entfernte Schnittstelle muss nach dem Wiedereinstecken erst wieder geöffnet werden.

Wird eine serielle Schnittstelle während der Programmlaufzeit hinzugefügt, wird diese im Kombinationsfeld mit aufgelistet und es erscheint folgende Meldung:

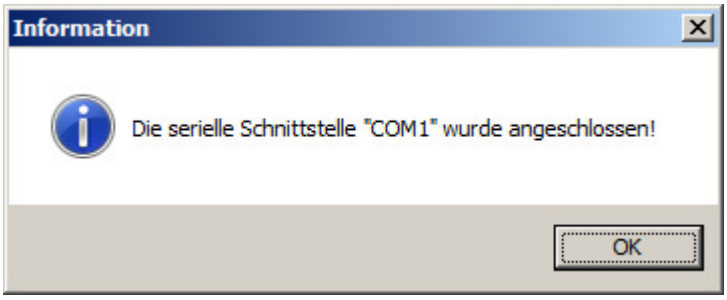

Bild A.7: Information - serielle Schnittstelle COM 1 wurde angeschlossen

In der Meldung wird der Name der seriellen Schnittstelle, hier z. B. COM 1, angezeigt.

Sollte eine serielle Schnittstelle bereits durch eine andere Anwendung verwendet werden, erscheint folgende Meldung:

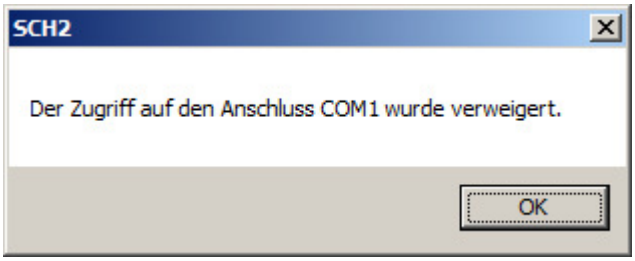

Bild A.8: Meldung - Der Zugriff auf die serielle Schnittstelle COM 1 wurde verweigert

Sollte diese Schnittstelle dennoch verwendet werden, muss entweder die andere Anwendung oder zumindest die geöffnete Schnittstelle geschlossen werden.

A.4.2 Hopper

Die Gruppe *Hopper* dient zur Interaktion mit dem *Serial Compact Hopper MK2* und besteht aus den Kombinationsfeldern **Hopper Address**, **Command**, **Block No.**, den Tasten **Execute** und **Hopper Info** sowie aus den zwei Gruppen *ID-Number* und *Dispense Coins*.

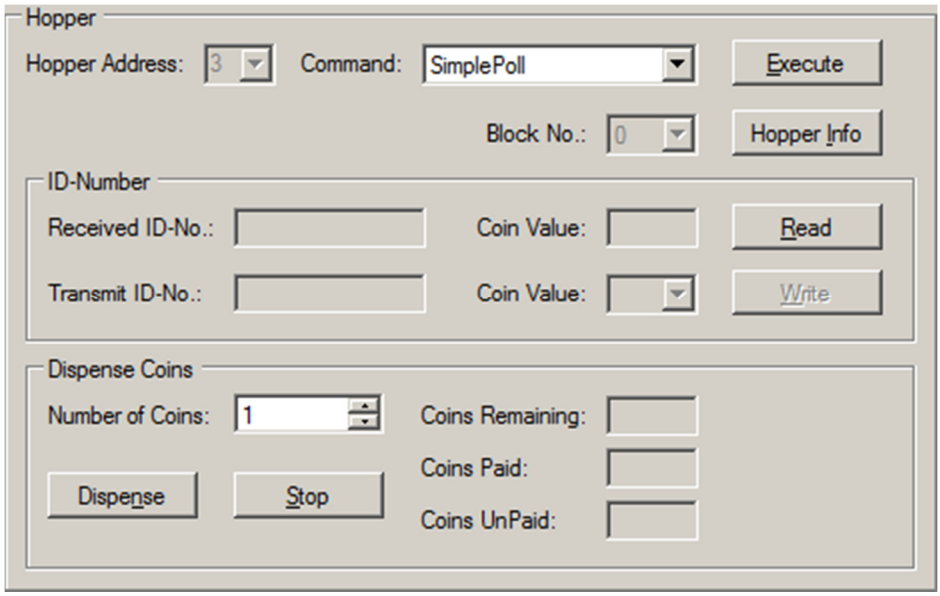

Bild A.9: Gruppe *Hopper* - Ausschnitt aus dem Hauptfenster

A.4.2.1 Kombinationsfeld Hopper Address

Die Hopper verfügen über drei Adresspins, mit denen ihre Adresse hardwaremäßig eingestellt werden kann. Deshalb können an den *ccTalk*-Bus bis zu acht Hopper angeschlossen werden. Hardwaremäßig bedeutet dies, dass ein festes Potenzial bzw. eine Spannung von 0 V für eine logische 0 oder 24 V für eine logische 1 am Pin angelegt wird. Durch diese drei Pins können die Adressen 000_2 bis 111_2 bzw. 0_{10} bis 7_{10} vergeben werden. Im *ccTalk*-Standard ist die Adresse 0_{10} als Broadcastadresse und die Adresse 1_{10} als Standardadresse für den Hostcontroller reserviert und deshalb entspricht die am Hopper eingestellte Adresse 0_{10} der Adresse 3_{10} und die Adresse 7_{10} entspricht der Adresse 10_{10}. Diese Adressenzuordnung findet innerhalb des Hoppers statt. Allgemein ist bei der hardwaremäßigen Adresszuordnung mit einem Adressoffset von drei zu rechnen. Sollten diese drei Adresspins freigelassen werden, hat der Hopper die Adresse 3_{10}.

Hinweis: Die Adresse des Hoppers ist zunächst auf den Wert 3 voreingestellt, welcher der Adresse eines Hoppers entspricht, dem keine Adresse hardwaremäßig (alle drei Adresspins sind frei) zugeordnet wurde. Die in der Testsoftware voreingestellte Adresse kann zzt. nicht verändert werden. Mit der aktuellen Version der Testsoftware und des vorhandenen Seriell-zu-*ccTalk*-Adapters kann zzt. immer nur ein einziger Hopper angesteuert werden.
Begründung: Die Anzahl der zzt. vorhandenen Hopper liegt im zweistelligen Bereich und deshalb ist es zum aktuellen Zeitpunkt nicht erforderlich, mehrere Hopper am *ccTalk*-Bus anzuschließen. Dies wird in einem zukünftigen Update der Software berücksichtigt. Hierbei ist auch zu beachten, dass eine Anpassung der Hardware bzw. der Anschlussbelegung der anzuschließenden Hopper notwendig wäre, wenn die Einstellung der Adressen der Hopper hardwaremäßig erfolgen sollte.

A.4.2.2 Kombinationsfeld Command

Im Kombinationsfeld **Command** sind alle zweckmäßigen Befehle für das manuelle Testen des Hoppers aufgelistet. Durch die Auswahl eines Befehls und der Betätigung der Taste **Execute** wird der ausgewählte Befehl ausgeführt bzw. über die serielle Schnittstelle an den Hopper gesendet. I. d. R. wird, wenn kein Übertragungsfehler oder sonstige Fehler auftreten, die Antwort des Hoppers empfangen und ausgewertet. Das gesendete sowie das empfangene Datenpaket werden in der Gruppe *Traffic* angezeigt. Eine detaillierte Darstellung des empfangenen Datenpakets erfolgt in der Gruppe *Received Data*.

Hinweis: Die gesendeten Daten sind die Daten, die tatsächlich am *ccTalk*-Bus angekommen sind. Es wird das Echo angezeigt und nicht die Daten, die den Rechner verlassen haben.

A.4.2.3 Kombinationsfeld Block No.

Das Kombinationsfeld **Block No.** ist zunächst deaktiviert und wird nur aktiviert, wenn der Befehl *Read Data Block* im Kombinationsfeld **Command** ausgewählt wurde. Mit dem Befehl *Read Data Block* und der jeweiligen **Block No.** können dann die gespeicherten Bytes aus dem EEPROM des Hoppers ausgelesen werden. Das EEPROM hat vier acht Byte große Register. Diese entsprechen den Block-Nrn. 0 bis 3.

Das Kombinationsfeld **Block No.** besitzt eine Doppelfunktion. Wenn aus der Befehlsliste der Befehl *Dispense Hopper Coins* ausgewählt wird, ändert sich die Bezeichnung **Block No.** in **Coins No.** und es kann mittels des Kombinationsfelds eine Anzahl an Münzen ausgewählt werden, welche der Hopper ausgeben soll.

Hinweis: Der Befehl *Dispense Hopper Coins* kann nicht direkt ausgeführt werden, da zunächst weitere Befehle vorab gesendet werden müssen.

A.4.2.4 Taste Execute

Die Taste **Execute** führt den aus dem Kombinationsfeld **Command** ausgewählten Befehl aus.

A.4.2.5 Taste Hopper Info

Die Taste **Hopper Info** führt eine Reihe an Befehlen aus, um die Informationen sowie die Status-/Fehlermeldungen des Hoppers anzuzeigen. Es wird das Informationsfenster (Info Window) gezeigt, in dem alle Daten angezeigt werden. Bei einer erneuten Betätigung der Taste werden die bisherigen Inhalte gelöscht und anschließend die neuen Daten angezeigt.

Hinweis: Während der Befehlsausführungen der Taste **Hopper Info** können keine anderen Befehle getätigt werden. Deshalb werden die Taste **Hopper Info** und alle anderen Tasten dieser Gruppe deaktiviert.

A.4.3 ID-Number

Mit der Gruppe *ID-Number* kann die vom Automatenhersteller eingestellte ID-Nr. des Hoppers ausgelesen und ggf. geändert werden.

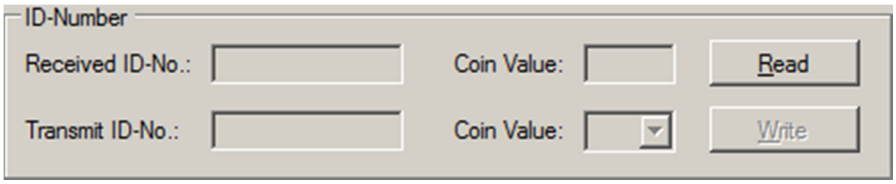

Bild A.10: Gruppe *ID-Number* - Ausschnitt aus dem Hauptfenster

Die Gruppe *ID-Number* besteht aus zwei Zeilen. In der ersten Zeile kann mittels der Taste **Read** die im Hopper eingespeicherte ID-Nr. ausgelesen werden. Die ID-Nr. wird im Textfeld **Received ID-No.** angezeigt. Anhand der ausgelesenen ID-Nr. wird die Münzwertigkeit, für die der Hopper eingestellt wurde, erkannt. Diese wird im Textfeld **Coin Value** angezeigt.

Hinweis: Die angezeigte ID-Nr. und die Münzwertigkeit des Hoppers sollten den Angaben auf dem aufgeklebten Barcode am Gehäuse des Hoppers entsprechen.

Die ID-Nr. eines Hoppers kann nur geändert werden, wenn die im Hopper eingespeicherte ID-Nr. vorher eingelesen wurde. Aus Sicherheitsgründen ist es nicht möglich, die ID-Nr. direkt im Textfeld **Transmit ID-No.** einzugeben. Es kann nur die gewünschte Münzwertigkeit aus dem dafür vorgesehenen Kombinationsfeld **Coin Value** ausgewählt werden.

A.4.3.1 Beispiel: ID-Nr. auslesen und ändern (Hopper mit ID-Nr.)

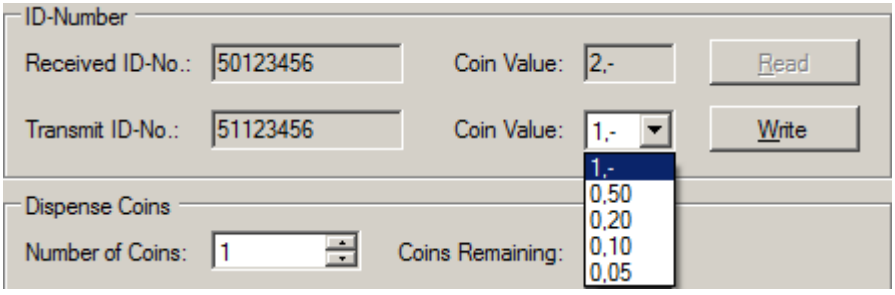

Bild A.11: Gruppe *ID-Number* mit der aus dem Hopper ausgelesenen ID-Nr. *50123456*

Zuvor wurde mittels der Taste **Read** die ID-Nr. des Hoppers ausgelesen. Die ausgelesene ID-Nr. wird im Textfeld **Received ID-No.** angezeigt, lautet *50123456* und entspricht der Münzwertigkeit 2 €.

Die Münzsorte, für die ein Hopper zuständig ist, wird aus der ID-Nr. ermittelt.

Aufbau der ID-Nr. 50123456

Die erste Ziffer der ID-Nr. dient zur Identifikation einer Baugruppe im Automaten. Die *5* bedeutet, dass es sich um ein Auszahlungsgerät bzw. Hopper handelt. Die zweite Ziffer der ID-Nr. dient zur Identifikation der Münzsorte, mit der ein Hopper gefüllt wurde. Ein Hopper kann nur mit einer bestimmten Münzsorte gefüllt werden. Im Beispiel hat der Hopper die Ziffer *0* und ist deshalb für die Münzsorte 2 € zuständig. Die restlichen sechs Ziffern dienen als laufende Serien-Nr. des Hoppers und dürfen nicht geändert werden. Die laufende Serien-Nr. darf nur einmalig vorkommen.

Tabelle A.2: Zuordnungstabelle zur Identifikation der Münzsorte aus der ID-Nr.

Zweite Ziffer der ID-Nr.	Münzsorte in Euro
0	2
1	1
2	0,50
3	0,10
4	0,05
7	0,20

Nun kann die ID-Nr. des Hoppers geändert werden, damit dieser mit einer anderen Münzsorte gefüllt werden kann. Aus der Liste des Kombinationsfelds kann die neue Münzsorte ausgewählt werden und mittels der Taste **Write** wird die neue ID-Nr. im Hopper eingespeichert. Im Beispiel wurde die Münzsorte 1 € ausgewählt.

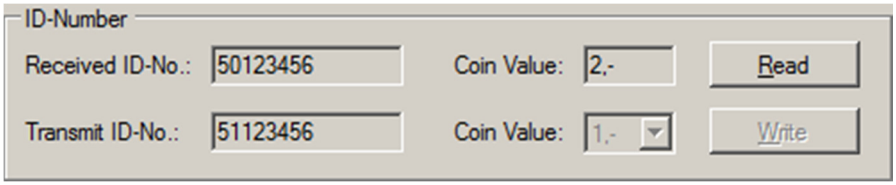

Bild A.12: Gruppe *ID-Number* mit der neu im Hopper eingespeicherten ID-Nr. *51123456*

Nachdem die Taste **Write** betätigt wurde, ist die neue ID-Nr. im Hopper eingespeichert. Aus Sicherheitsgründen wird die Taste **Write** deaktiviert und erst nach einem erneuten Lesevorgang wieder aktiviert. Die zuvor gespeicherte ID-Nr. bleibt noch solange angezeigt, bis die Taste **Read** betätigt wird. Dies hat den Vorteil, dass der Anwender die vorher eingespeicherte ID-Nr. noch angezeigt bekommt. Um sicherzustellen, dass die neue ID-Nr. tatsächlich im Hopper eingespeichert wurde, kann die Taste **Read** betätigt werden. Nun wird die neue ID-Nr. im Textfeld **Received ID-No.** angezeigt.

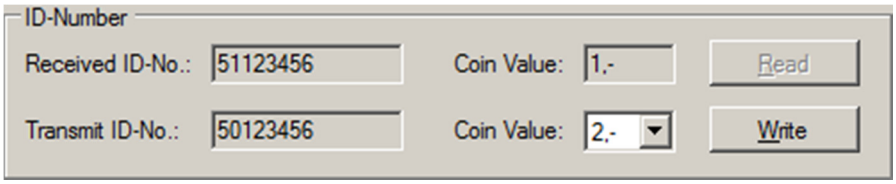

Bild A.13: Gruppe *ID-Number* mit der aus dem Hopper ausgelesenen ID-Nr. *51123456*

Hinweis: Nachdem die ID-Nr. eines Hoppers geändert wurde, muss dies unbedingt mit einem Aufkleber auf dem Gehäuse gekennzeichnet werden, um zu verhindern, dass ein Hopper fälschlicherweise mit einer anderen als der vorgesehenen Münzsorte gefüllt wird.

A.4.3.2 Beispiel: ID-Nr. auslesen und ändern (Hopper ohne ID-Nr.)

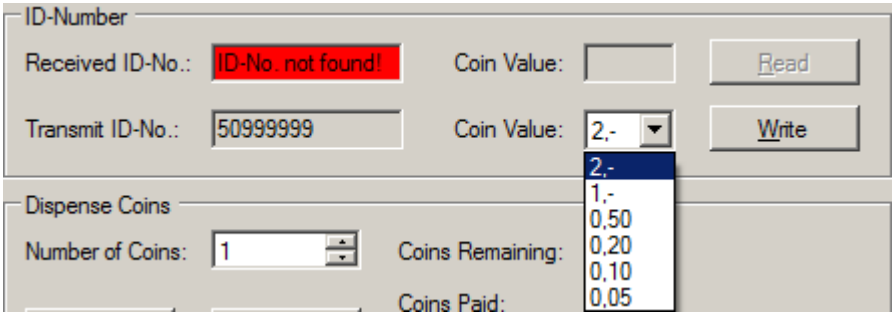

Bild A.14: Gruppe *ID-Number* mit einer im Hopper nicht vorhandenen ID-Nr.

Zuvor wurde mittels der Taste **Read** die ID-Nr. des Hoppers ausgelesen. In diesem Fall wurde dem Hopper noch keine ID-Nr. zugewiesen. Aus diesem Grund erscheint im Textfeld **Received ID-No.** der Text *„ID-No. not found!“*.

Bei der Vergabe der ID-Nrn. wurde erkannt, dass die hinteren sechs Ziffern aufwärts gezählt werden. Damit keine doppelten ID-Nrn. in den Umlauf kommen, wurde entschieden, diejenigen Hopper, die keine ID-Nr. vom Automatenhersteller erhalten haben, ab der ID-Nr. 5x999999 abwärts zu vergeben. Wird eine ID-Nr. vergeben, muss sie gespeichert werden, damit der nächste Hopper ohne ID-Nr. die nächste bzw. die ID-Nr. 5x999998 erhält. Wird eine ID-Nr. in einen Hopper eingespeichert, erfolgt die Speicherung in der Textdatei *„ID-Number_list.txt“*.

Inhalt der Textdatei *„ID-Number_list.txt“:* 2013_04_14 15:23:35 50999999

Es werden lediglich das Datum, die aktuelle Systemzeit und die vergebene ID-Nr. abgespeichert.

A.4.4 Dispense Coins

Mit der Gruppe *Dispense Coins* kann die Münzausgabe des Hoppers geprüft werden.

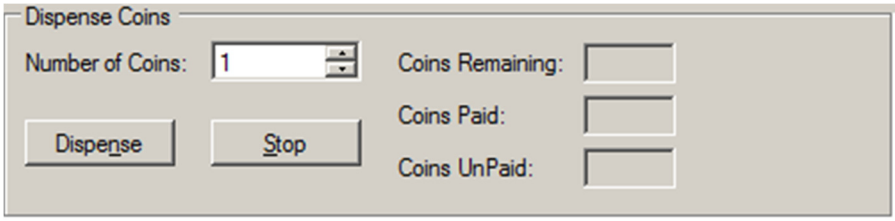

Bild A.15: Gruppe *Dispense Coins* - Ausschnitt aus dem Hauptfenster

Die Gruppe *Dispense Coins* besteht aus mehreren Steuerelementen, welche im Folgenden näher erläutert werden.

Das numerische Feld **Number of Coins** dient zur Eingabe der Anzahl der Münzen, die ausgegeben werden sollen. Das Feld besitzt eine Auf- und eine Abschaltfläche mit denen der Wert erhöht bzw. verringert werden kann. Der minimale einstellbare Wert ist eine Münze und es können maximal 255 Münzen angegeben werden. Die Einstellung der Anzahl der Münzen kann auch mittels Tastatur erfolgen.

Die Taste **Dispense** dient zur Aktivierung der Münzausgabe und die Taste **Stop** zur sofortigen Abschaltung der Münzausgabe.

In den Textfeldern **Coins Remaining**, **Coins Paid** und **Coins UnPaid** wird der aktuelle Status der Münzausgabe angezeigt. Um dies zu realisieren, wird während der Münzausgabe der Ausgabestatus des Hoppers in einem Intervall von 200 ms abgefragt.

Tabelle A.3: Statusmeldungen der Münzausgabe der Gruppe *Dispense Coins*

Status	Beschreibung
Coins Remaining	Zeigt die Anzahl der noch auszugebenden Münzen an. Der Wert wird alle 200 ms aktualisiert.
Coins Paid	Zeigt die Anzahl der bereits ausgegebenen Münzen an. Der Wert wird alle 200 ms aktualisiert.
Coins UnPaid	Zeigt die Anzahl der Münzen an, die nicht ausgegeben werden konnten. Dieser Wert wird erst nach Ablauf des Befehls bzw. nach Ablauf des Payout Timeouts angezeigt.

A.4.5 Traffic

In der Gruppe *Traffic* werden die gesendeten sowie die empfangenen Datenpakete angezeigt.

Bild A.16: Gruppe *Traffic* - Ausschnitt aus dem Hauptfenster

Die Gruppe *Traffic* besteht aus den Textfeldern **Transmitted** und **Received**. Im Textfeld **Transmitted** wird das gesendete und im Textfeld **Received** das empfangene Datenpaket angezeigt.

Bei dem gesendeten Datenpaket handelt es sich um die tatsächlich an dem *ccTalk*-Bus anliegenden bzw. angekommenen Daten, da das gesendete Datenpaket technisch bedingt als ein Echo ebenfalls im Eingangspuffer der seriellen Schnittstelle ankommt.
Die Entscheidung, das Echo anstatt der Daten am Ausgangsport der seriellen Schnittstelle anzuzeigen, dient zur Überprüfung bzw. Eingrenzung von evtl. auftretenden Fehlern. An dieser Stelle kann der Anwender feststellen, ob ein Fehler auf dem Bus vorliegt. Ein Fehler liegt demnach vor, wenn die gesendeten Daten nicht denen entsprechen, die eigentlich gesendet werden sollen. Diese Überprüfung findet zzt. in der Anwendung nicht statt.

Bei dem empfangenen Datenpaket handelt es sich um die Antwort des Hoppers auf einen bestimmten zuvor gesendeten Befehl. Das empfangene Datenpaket wird in der nachfolgend erklärten Gruppe *Received Data* in seine einzelnen Bestandteile aufgeteilt.

A.4.6 Received Data

In der Gruppe *Received Data* wird das empfangene Datenpaket bzw. die Antwort des Hoppers auf einen zuvor gesendeten Befehl in seine einzelnen Bestandteile zerlegt. Die empfangene Antwort wird im Textfeld **Received** der Gruppe *Traffic* zeilenweise angezeigt.

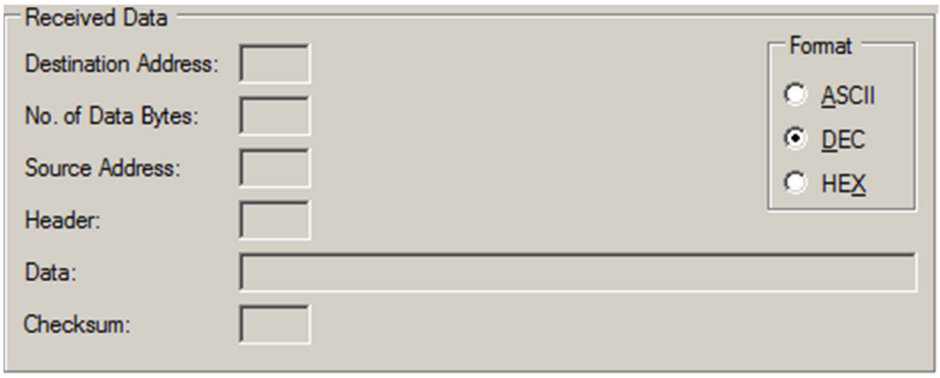

Bild A.17: Gruppe *Received Data* - Ausschnitt aus dem Hauptfenster

Ein Datenpaket besteht aus der Zieladresse, der Anzahl der gesendeten Datenbytes, der Quelladresse, dem Header, evtl. aus Datenbytes und der Prüfsumme.

Wenn keine Datenbytes gesendet wurden, steht im Feld *Anzahl der gesendeten Datenbytes* der Wert 0.

Tabelle A.4: Darstellung eines Datenpakets mit Datenbytes

Ziel-adresse	Anzahl Datenbytes	Quell-adresse	Header	Datenbyte 1	…	Datenbyte N	Prüf-summe

Tabelle A.5: Darstellung des minimalen Datenpakets

Ziel-adresse	0	Quell-adresse	Header	Prüf-summe

Die Gruppe *Received Data* beinhaltet die Gruppe *Format,* mit der die angezeigten Daten bei Bedarf zwischen den drei Darstellungsarten ASCII, Dezimal (DEC) und Hexadezimal (HEX) umgeschaltet werden können. Standardmäßig erfolgt die Darstellung in Dezimal.

A.4.6.1 Beispiel: Der Rechner sendet den Befehl *Simple Poll*

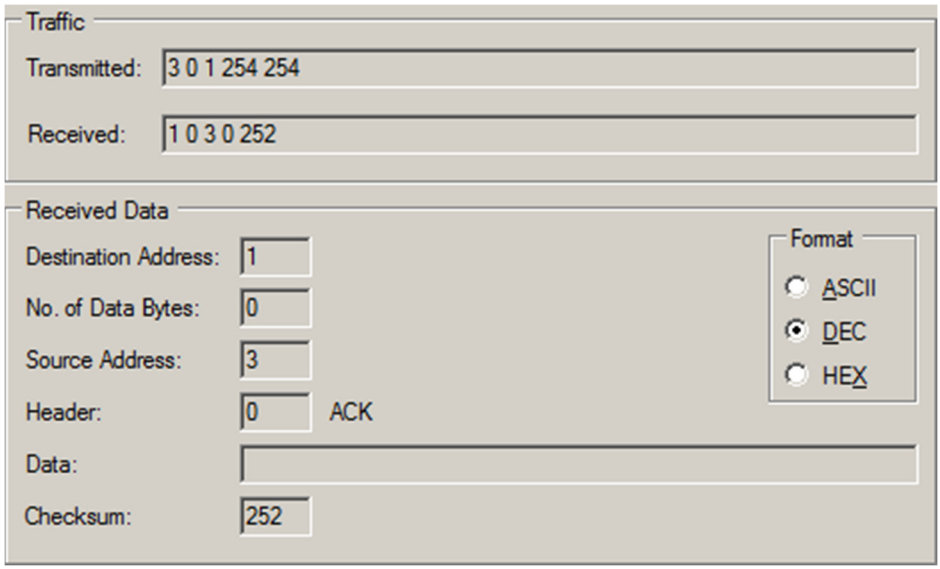

Bild A.18: Gruppen *Traffic* und *Received Data* nach dem gesendeten Befehl *Simple Poll*

Das gesendete Datenpaket des Rechners sieht wie folgt aus:

Tabelle A.6: Gesendetes Datenpaket des Rechners zum Befehl *Simple Poll*

Ziel-adresse	Anzahl Datenbytes	Quell-adresse	Header	Prüf-summe
3	0	1	254	254

Die Antwort bzw. Quittung des Hoppers auf den zuvor gesendeten Befehl bzw. das empfangene Datenpaket am Rechner sieht folgendermaßen aus:

Tabelle A.7: Antwort des Hoppers zum Befehl *Simple Poll*

Ziel-adresse	Anzahl Datenbytes	Quell-adresse	Header	Prüf-summe
1	0	3	0	252

A.5 Gruppen des Informationsfensters (Info Window)

A.5.1 Device Data

Die Gruppe *Device Data* beinhaltet die Gruppen *General* und *Variable Set*. In der Gruppe *General* werden die allgemeinen Informationen der Baugruppe bzw. des Hoppers angezeigt. In der Gruppe *Variable Set* werden eingestellte Parameterwerte sowie gemessene Spannungs- und Stromwerte angezeigt.

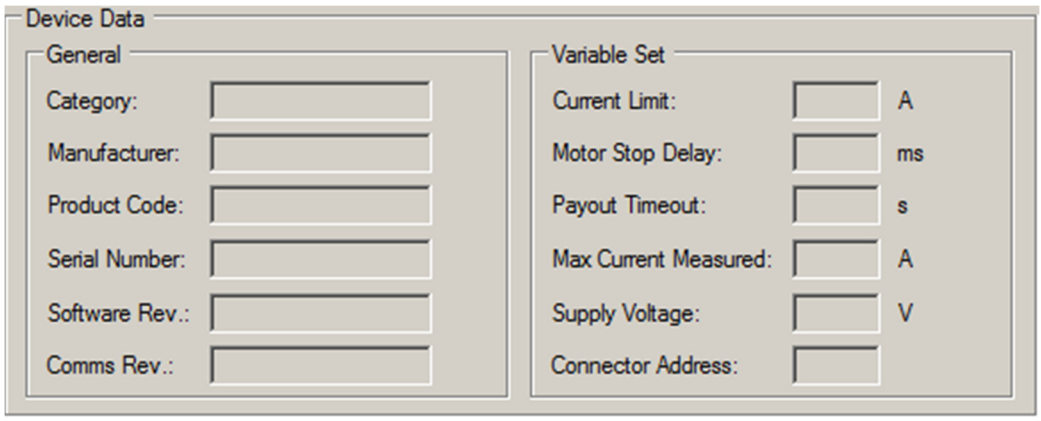

Bild A.19: Gruppe *Device Data* - Ausschnitt aus dem Informationsfenster

A.5.1.1 Gruppe *General*

Die Gruppe *General* besteht aus den Textfeldern **Category**, **Manufacturer**, **Product Code**, **Serial Number**, **Software Rev.** und **Comms Rev.**

Tabelle A.8: Beschreibung der Textfelder der Gruppe *General*

Textfeld	Beschreibung
Category	Beschreibt die Funktion des Geräts. Handelt es sich um ein Auszahlungsgerät bzw. Hopper, wird im Textfeld *Payout* angezeigt. Weitere Geräte könnten z. B. Münz- oder Banknotenprüfer sein. In diesen Fall werden jeweils die Texte *Coin Validator* bzw. *Bill Validator* angezeigt.
Manufacturer	Der Herstellername des Geräts. Beim *Serial Compact Hopper MK2* der Fa. Money Controls erscheint der Text *Money Controls*.
Product Code	Der Produktcode gibt die Bezeichnung des Geräts an und unterscheidet zwischen der verschlüsselten und der unverschlüsselten Version des Hoppers. Bei der unverschlüsselten Version erscheint der Text *SCH2-NOENCRYPT* und bei der verschlüsselten Version *SCH2*.

Serial Number	Die Serien-Nr. des Hopperherstellers. Die Serien-Nr. kommt nur einmalig vor und darf nicht mit der ID-Nr., welche vom Automatenhersteller im Hopper gespeichert wurde, verwechselt werden. Die Serien-Nr. kann, im Gegensatz zu der ID-Nr., nicht verändert werden.
Software Rev.	Die Revisionsnummer der im Hopper befindlichen Software. Zzt. hat diese den Stand 2.4. Im Textfeld erscheint *SCH2-V2.4*.
Comms Rev.	Die Revisionsnummer des Kommunikationsprotokolls *ccTalk*. Es erscheint *132* im Textfeld. Die erste Ziffer beschreibt das Level. Die zwei nachkommenden Ziffern sind die Revisionsnummer. Der Text *132* bedeutet, dass der Hopper die erste Ausgabe (Level) des *ccTalk*-Protokolls mit der Spezifikation 3.2 unterstützt.

A.5.1.2 Gruppe Variable Set

Die Gruppe *Variable Set* besteht aus den Textfeldern **Current Limit**, **Motor Stop Delay**, **Payout Timeout**, **Max Current Measured**, **Supply Voltage** und **Connector Address**.

Tabelle A.9: Beschreibung der Textfelder der Gruppe *Variable Set*

Textfeld	Beschreibung
Current Limit	Zeigt den eingestellten maximalen Stromwert an, welcher als Strombegrenzung für den im Hopper eingebauten Motor dient. Es ist der maximale Stromwert der bei einer Drehung des Motors im Rechtslauf im Fall einer Münzverklemmung erreicht werden kann. Der maximale Stromwert ist zzt. auf 2 A festgelegt und im Textfeld erscheint *1,99* A.
Motor Stop Delay	Die Zeit, in der sich der Motor noch dreht nachdem eine Münze den Hopper verlassen hat. Mittels einer Lichtschranke, welche sich direkt am Ausgangsschlitz des Hoppers befindet, wird erkannt, dass die Münze den Hopper verlassen hat. Der Wert ist zzt. auf 0 Sekunden eingestellt und im Textfeld erscheint *0* Sekunden.
Payout Timeout	Die Zeit, in der sich der Motor noch dreht, nachdem die letzte Münze den Hopper verlassen hat und noch mindestens eine Münze ausgegeben werden muss. Dieser Wert ist zzt. auf 10 Sekunden eingestellt und im Textfeld erscheint der Wert *9,99* Sekunden. Im Falle einer Münzverklemmung wird dem Hopper auf diese Weise noch Zeit zur Verfügung gestellt, die Münzverklemmung zu lösen. Nach Ablauf dieser Zeit wird der Motor gestoppt.

Max Current Measured	Zeigt den maximal gemessenen Stromwert des Motors in Ampere an. Der Hopper misst in zeitlichen Abständen die Stromaufnahme des Motors. Die Messung erfolgt nur bei einem laufenden Motor.
Supply Voltage	Zeigt die am Hopper anliegende Versorgungsspannung in Volt an. Die Spannung wird kontinuierlich gemessen, außer in der Zeit, in der eine Münzausgabe erfolgt. Die erlaubte Versorgungsspannung liegt im Bereich +19 V bis +26 V. Der typische Wert liegt bei +24 V. Liegt die Spannungsversorgung innerhalb des erlaubten Bereichs, wird der Hintergrund des Textfelds grün eingefärbt. Liegt die Spannungsversorgung außerhalb des erlaubten Bereichs, wird der Hintergrund des Textfelds rot eingefärbt.
Connector Address	Zeigt den Adresswert des Hoppers an. Der Adresswert entspricht der hardwaremäßig eingestellten Adresse am Hopper. Wenn die Adresspins des Hoppers freigelassen werden, hat dieser die Adresse 0. Im Textfeld erscheint der Wert 0. Innerhalb des Hoppers findet eine Adresszuordnung statt, weil im *ccTalk*-Protokoll die Adresse 0 bereits reserviert ist. Die hardwaremäßig eingestellte Adresse 0 entspricht der Adresse 3 in *ccTalk*. Grundsätzlich ist bei der hardwaremäßigen Adresszuordnung mit einem Adressoffset von drei zu rechnen.

A.5.1.3 Beispiel: Gruppe *Device Data* mit ausgefüllten Werten

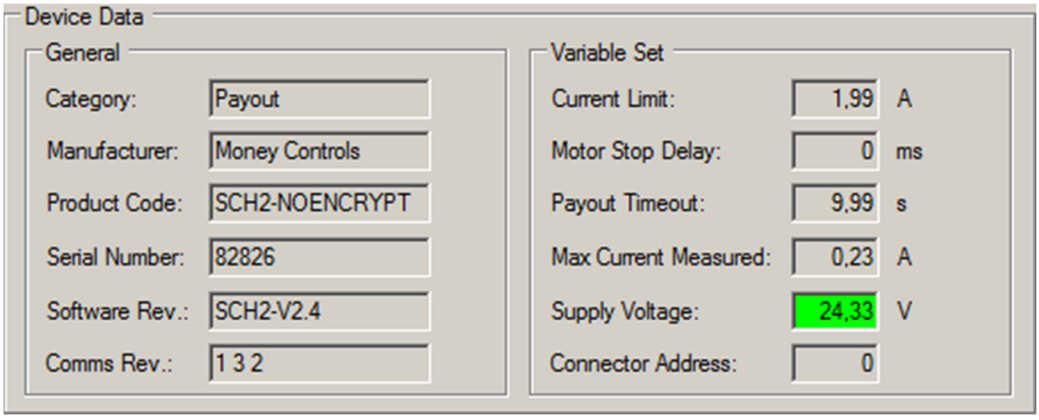

Bild A.20: Gruppe *Device Data* nach der Betätigung der Taste *Hopper Info*

A.5.2 NV (Non Volatile) Data

In der Gruppe *NV Data* werden die im EEPROM des Hoppers gespeicherten Werte angezeigt.

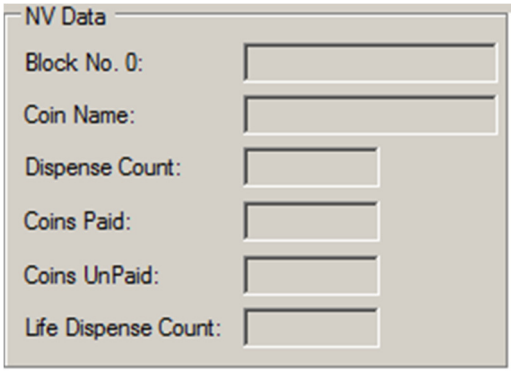

Bild A.21: Gruppe *NV Data* - Ausschnitt aus dem Informationsfenster

Die Gruppe *NV Data* besteht aus den Textfeldern **Block No. 0**, **Coin Name**, **Dispense Count**, **Coins Paid**, **Coins UnPaid** und **Life Dispense Count**.

Tabelle A.10: Beschreibung der Textfelder der Gruppe *NV Data*

Textfeld	Beschreibung
Block No. 0	In der Block-Nr. 0 ist die ID-Nr. des Hoppers vom Automatenhersteller gespeichert. Ein Block besteht aus acht Bytes. Für die ID-Nr. werden nur die vier niederwertigen Bytes benötigt. Die restlichen vier Bytes haben den Wert 0. Die empfangenen Bytes werden LSB beginnend angezeigt.
Coin Name	Hier wird der Name der Münze angezeigt, für die der Hopper zuständig ist. Diese Möglichkeit wurde nicht verwendet, sodass immer die vom Hersteller des Hoppers eingetragenen Werte 45 45 45 45 45 45 4 6 angezeigt werden.
Dispense Count	Die Gesamtanzahl der ausgegebenen Münzen nachdem der Zähler das letzte Mal zurückgesetzt wurde. Dieser Wert wird erst nach einem Hardware- oder Software-Reset aktualisiert. Dieser Zähler kann mittels des Befehls *Write Data Block* zurückgesetzt werden.
Coins Paid	Die Anzahl der zuletzt ausgegebenen Münzen.
Coins UnPaid	Die Anzahl der Münzen, welche nicht ausgegeben werden konnten.
Life Dispense Count	Die Gesamtanzahl aller ausgegebenen Münzen seit dem der Hopper hergestellt wurde.

A.5.2.1 Beispiel: Gruppe *NV Data* mit ausgefüllten Werten

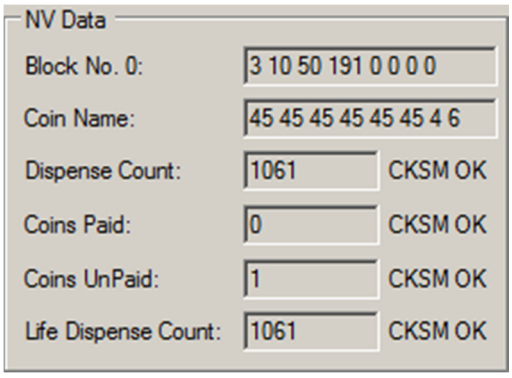

Bild A.22: Gruppe *NV Data* nach der Betätigung der Taste *Hopper Info*

Neben den Textfeldern **Dispense Count**, **Coins Paid**, **Coins UnPaid** und **Life Dispense Count** erscheint jeweils der Text *CKSM OK*. Jeder dieser Werte besitzt eine Prüfsumme, um die Datenintegrität zu gewährleisten. Ist die Prüfsumme korrekt, erscheint, wie im Beispiel zu sehen ist, der Text *CKSM OK*. Andernfalls erscheint der Text *CKSM nOK*. In diesem Fall könnte es sein, dass der angezeigte Wert nicht stimmt.

A.5.3 High/Low Level Status

In der Gruppe *High/Low Level Status* werden die im Hopper verfügbaren Sensoren und der Füllstand des Hoppers ermittelt und angezeigt.

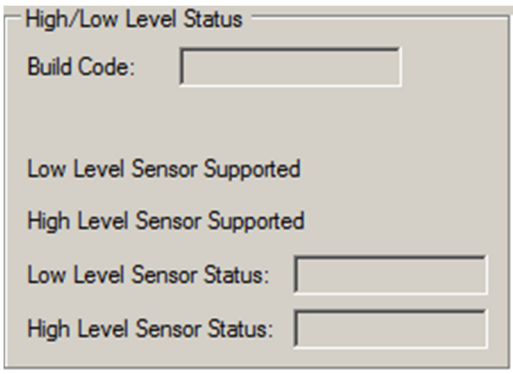

Bild A.23: Gruppe *High/Low Level Status* - Ausschnitt aus dem Informationsfenster

Die Gruppe *High/Low Level Status* besteht aus den Textfeldern **Build Code**, **Low Level Sensor Status, High Level Sensor Status** und den zwei Bezeichnern **Low Level Sensor Supported** und **High Level Sensor Supported**.

Tabelle A.11: Beschreibung der Textfelder der Gruppe *High/Low Level Status*

Textfeld/Bezeichner	Beschreibung
Build Code	Hier wird angezeigt, welche Sensoren eingebaut bzw. vorhanden sind. Zzt. besitzen die Hopper nur den Low Level Sensor. In diesem Fall erscheint im Textfeld `Lev Lo`. Weitere mögliche Texte wären: `Lev HiLo` Der Hopper besitzt beide Sensoren. `Lev Hi` Der Hopper besitzt nur den High Level Sensor. `Standard` für das Standard Model (SCH1).
Low Level Sensor Supported	Wenn der Hopper den Low Level Sensor unterstützt, wird der Bezeichner angezeigt. Andernfalls wird der Text grau dargestellt.
High Level Sensor Supported	Wenn der Hopper den High Level Sensor unterstützt, wird der Bezeichner angezeigt. Andernfalls wird der Text grau dargestellt.
Low Level Sensor Status	Im Textfeld wird je nach Füllstand des Hoppers entweder *NEARLY EMPTY*, wenn der Füllstand sich unterhalb des Sensors befindet oder *NEARLY FULL*, wenn der Füllstand sich oberhalb des Sensors oder auf gleicher Höhe mit dem Sensor befindet, angezeigt. In Abhängigkeit des Füllstands wird der Hintergrund des Textfelds geändert. Ist der Hopper fast leer, wird der Hintergrund des Textfelds orange eingefärbt. Sollte der Hopper fast voll sein, wird der Hintergrund des Textfelds grün eingefärbt.
High Level Sensor Status	Im Textfeld wird je nach Füllstand des Hoppers entweder *NEARLY EMPTY*, wenn der Füllstand sich unterhalb des Sensors befindet oder *NEARLY FULL*, wenn der Füllstand sich oberhalb des Sensors oder auf gleicher Höhe mit dem Sensor befindet, angezeigt. In Abhängigkeit des Füllstands wird der Hintergrund des Textfelds geändert. Ist der Hopper fast leer, wird der Hintergrund des Textfelds orange eingefärbt. Sollte der Hopper fast voll sein, wird der Hintergrund des Textfelds grün eingefärbt.

A.5.3.1 Beispiel: Gruppe *High/Low Level Status* mit ausgefüllten Werten

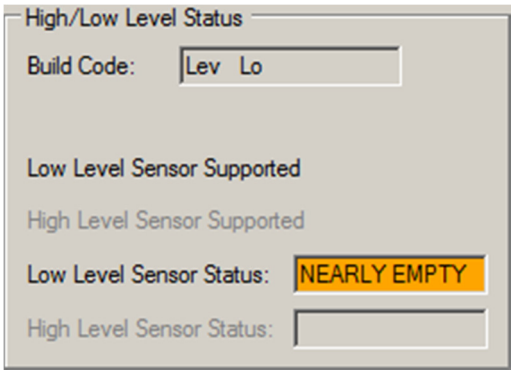

Bild A.24: Gruppe *High/Low Level Status* nach der Betätigung der Taste *Hopper Info*

Aus dem Textfeld **Build Code** sowie aus dem angezeigten Bezeichner **Low Level Sensor Supported** ist zu erkennen, dass der angeschlossene Hopper den Low Level Sensor unterstützt.
Im Textfeld **Low Level Sensor Status** wird der Text *NEARLY EMPTY* angezeigt und der Hintergrund wird in der Farbe Orange dargestellt. Der Füllstand des Hoppers befindet sich also unterhalb des Sensors.

Nach dem Füllen des Hoppers und einer erneuten Betätigung der Taste **Hopper Info** ändert sich der Textinhalt des **Low Level Sensor Status** in *NEARLY FULL* und der Hintergrund wird in der Farbe Grün dargestellt.

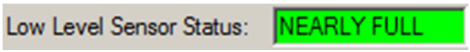

Bild A.25: Gruppe *High/Low Level Status* mit Low Level Sensor Status = *NEARLY FULL*

Hinweis: Der Status der Sensoren wird innerhalb des Hoppers nur alle 2 Sekunden abgefragt. Aus diesem Grund kann es vorkommen, dass der angezeigte Status nicht aktualisiert wird. Es ist nicht bekannt, wann der Hopper die Abfrage macht. Deshalb ist die Verzögerung variabel und beträgt maximal 2 Sekunden. Sollte der Zustand sich nicht geändert haben, muss die Taste **Hopper Info** erneut betätigt werden. Am besten ist es, nachdem der Hopper gefüllt wurde, die Zeit von 2 Sekunden abzuwarten, bis die Taste betätigt wird.

Hinweis: Der *Serial Compact Hopper MK2* unterstützt zzt. nur den Low Level Sensor.

A.5.4 Test Hopper

Die Gruppe *Test Hopper* beinhaltet die Gruppen *Hopper Status Register 1* und *Hopper Status Register 2*. In der Gruppe *Test Hopper* werden die Status-/Fehlermeldungen der Baugruppe bzw. des Hoppers angezeigt.

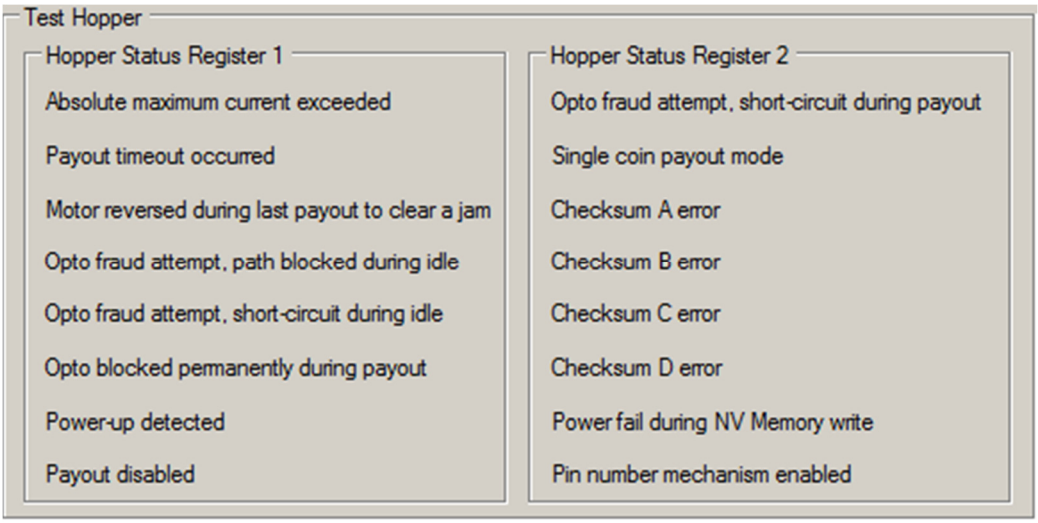

Bild A.26: Gruppe *Test Hopper* - Ausschnitt aus dem Informationsfenster der Testsoftware

A.5.4.1 Gruppe Hopper Status Register 1

Die Gruppe *Hopper Status Register 1* besteht aus den Bezeichnern:

- **Absolute maximum current exceeded**
- **Payout timeout occurred**
- **Motor reversed during last payout to clear a jam**
- **Opto fraud attempt, path blocked during idle**
- **Opto fraud attempt, short-circuit during idle**
- **Opto blocked permanently during payout**
- **Power-up detected**
- **Payout disabled**

Tabelle A.12: Beschreibung der Bezeichner der Gruppe *Hopper Status Register 1*

Bezeichner	Beschreibung
Absolute maximum current exceeded	Die Münzausgabe wurde gestoppt, weil der maximale Stromwert überschritten wurde. Dies ist das Softwareäquivalent zu einer Sicherung. Wenn diese Meldung erscheint, konnte entweder eine Münzverklemmung nicht gelöst werden oder es liegt ein Motorfehler vor. Die Meldung muss durch einen Software-Reset gelöscht werden. Sollte der Fehler nicht behoben werden, bleibt die Meldung bestehen und es kann keine Münzausgabe erfolgen. Die Meldung erscheint, wenn der angezeigte Stromwert im Textfeld *Current Limit* der Gruppe *Variable Set* überschritten wurde. Der im Textfeld *Current Limit* angezeigte Stromwert dient als Strombegrenzung für den im Hopper eingebauten Motor. Der maximale Stromwert ist zzt. auf 2 A festgelegt.
Payout timeout occurred	Diese Meldung weist auf das Beenden der Münzausgabe nach Ablauf eines Zeitwerts hin, wenn durch Zeitablauf nicht alle Münzen ausgegeben werden konnten. Grund hierfür könnte z. B. eine Münzverklemmung sein oder der Hopper verfügt über keine Münzen mehr. Die Meldung erscheint, wenn der Zeitwert im Textfeld *Payout Timeout* der Gruppe *Variable Set* abgelaufen ist. Der Zeitwert ist zzt. auf 10 s festgelegt.
Motor reversed during last payout to clear a jam	Diese Meldung erscheint, wenn eine Münzverklemmung vorliegt und der Motor seine Drehrichtung ändern musste, um sie zu lösen. Die Meldung bleibt solange bestehen, bis ein Software-Reset erfolgt.
Opto fraud attempt, path blocked during idle	Die Lichtschranke am Münzausgangsschlitz des Hoppers wurde während des Leerlaufs bzw. während der Zeit, in der keine Münzausgabe erfolgte, blockiert. Dies kann vorkommen, wenn eine Münze am Münzausgang stecken bleibt.
Opto fraud attempt, short-circuit during idle	Die Lichtschranke am Münzausgangsschlitz des Hoppers wurde während des Leerlaufs bzw. während der Zeit, in der keine Münzausgabe erfolgte, kurzgeschlossen. Dies kann vorkommen, wenn eine direkte Lichteinstrahlung am Münzausgang erfolgt. Die Meldung dient zur Verhinderung einer Manipulation am Hopper.

Opto blocked Permanently during payout	Die Lichtschranke am Münzausgangsschlitz des Hoppers wurde während einer Münzausgabe blockiert. Dies kann vorkommen, wenn eine Münze am Münzausgang stecken bleibt.
Power-up detected	Diese Meldung erscheint, wenn der Hopper zwischenzeitlich von der Spannungsversorgung getrennt war. Sie erscheint nur bei einem Hardware-Reset und kann durch ein Software-Reset mittels des Befehls *Reset Device* gelöscht werden.
Payout disabled	Die Münzausgabe ist deaktiviert. Um die Münzausgabe zu aktivieren, muss der Befehl *Enable Hopper* ausgeführt werden. Diese Meldung erscheint immer nach einem Hardware- bzw. Software-Reset.

A.5.4.2 Gruppe Hopper Status Register 2

Die Gruppe *Hopper Status Register 2* besteht aus den Bezeichnern:

- **Opto fraud attempt, short-circuit during payout**
- **Single coin payout mode**
- **Checksum A error**
- **Checksum B error**
- **Checksum C error**
- **Checksum D error**
- **Power fail during NV Memory write**
- **Pin number mechanism enabled**

Tabelle A.13: Beschreibung der Bezeichner der Gruppe *Hopper Status Register 2*

Bezeichner	Beschreibung
Opto fraud attempt, short-circuit during payout	Die Lichtschranke am Münzausgangsschlitz des Hoppers wurde während einer Münzausgabe kurzgeschlossen. Dies kann vorkommen, wenn eine direkte Lichteinstrahlung am Münz-ausgang erfolgt. Die Meldung dient zur Verhinderung einer Manipulation am Hopper.
Single coin payout mode	Diese Meldung erscheint, wenn der Hopper im Modus *Single coin payout* eingestellt ist. In diesem Modus kann der Befehl für die Münzausgabe nur eine einzige Münze ausgeben. Der Standard Modus ist der *Multi coin payout* mit dem der Befehl für die Münzausgabe bis zu 255 Münzen ausgeben kann.

Checksum A error	Wird angezeigt, wenn die Prüfsumme für den Wert *Dispense Count* nicht stimmt. Die Prüfsumme dient zur Datenintegrität. Der Wert *Dispense Count* lässt sich aus drei Bytes berechnen. Diese drei Bytes werden inkl. der Prüfsumme aufaddiert. Das Ergebnis der Restdivision (Modulo) zwischen der aufaddierten Summe und dem Wert 256 muss den Wert 0 ergeben. Andernfalls hat evtl. eine Datenmanipulation stattgefunden.
Checksum B error	Wird angezeigt, wenn die Prüfsumme für den Wert *Coins Paid* nicht stimmt. Der Wert *Coins Paid* besteht nur aus einem Byte. Dieser wird mit der Prüfsumme addiert. Das Ergebnis der Restdivision zwischen der aufaddierten Summe und dem Wert 256 muss den Wert 0 ergeben. Andernfalls hat evtl. eine Datenmanipulation stattgefunden.
Checksum C error	Wird angezeigt, wenn die Prüfsumme für den Wert *Coins UnPaid* nicht stimmt. Der Wert *Coins UnPaid* besteht nur aus einem Byte. Dieser wird mit der Prüfsumme addiert. Das Ergebnis der Restdivision zwischen der aufaddierten Summe und dem Wert 256 muss den Wert 0 ergeben. Andernfalls hat evtl. eine Datenmanipulation stattgefunden.
Checksum D error	Wird angezeigt, wenn die Prüfsumme für den Wert *Life Dispense Count* nicht stimmt. Die Prüfsumme dient zur Datenintegrität. Der Wert *Life Dispense Count* lässt sich aus drei Bytes berechnen. Diese drei Bytes werden inkl. der Prüfsumme aufaddiert. Das Ergebnis der Restdivision zwischen der aufaddierten Summe und dem Wert 256 muss den Wert 0 ergeben. Andernfalls hat evtl. eine Datenmanipulation stattgefunden.
Power fail during NV Memory write	Die Meldung erscheint, wenn während eines Schreibzugriffs auf den EEPROM des Hoppers die Spannungsversorgung unterbrochen wurde.
Pin number mechanism enabled	Diese Meldung erscheint, wenn der PIN Mechanismus aktiviert ist. In diesem Fall muss, bevor der Befehl zur Münzausgabe erfolgt, der PIN mittels des Befehls *EnterPINnumber* eingegeben werden. Der PIN Mechanismus ist eine optionale Sicherheitseigenschaft und deshalb standardmäßig deaktiviert.

B Klassen und Module

In den nachfolgenden Tabellen sind die verwendeten Klassen und Module sowie deren Methoden/Ereignisse alphabetisch aufgelistet.

B.1 Klassen

Tabelle B.1: Auflistung der Klassen und deren Beschreibung

Klasse	Beschreibung
Command	Die Klasse Command dient als Vorlage für den Aufbau eines zu sendenden Befehls.
InfoWindow	Die Klasse *InfoWindow* beinhaltet Methoden mit denen die Informationen sowie Status-/Fehlermeldungen des Hoppers auf der grafischen Oberfläche des Informationsfensters angezeigt werden.
MainWindow	Die Klasse *MainWindow* beinhaltet diverse Methoden mit denen die Interaktion zwischen den Steuerelementen des Hauptfensters und den restlichen Modulen der Anwendung stattfindet. Sie ist folglich das Bindeglied zwischen der grafischen Oberfläche des Hauptfensters und der eigentlichen Programmausführung. Betätigt der Anwender z. B. die Taste **Open** löst diese Aktion ein Ereignis aus, welches durch eine Behandlungsroutine der Klasse abgefangen wird. Diese führt dann die weiteren Programmschritte aus.
MyApplication	Die Klasse *MyApplication* stellt Eigenschaften, Methoden und Ereignisse bereit, die sich auf die aktuelle Anwendung beziehen. Sie beinhaltet die Methode *Startup*, welche beim Starten der Anwendung noch vor dem Erstellen des Startformulars (Hauptfensters) ausgelöst wird.
Received	Die Klasse *Received* dient als Vorlage für den Aufbau eines zu empfangenen Datenpakets.
Transmit	Die Klasse *Transmit* ist für den Aufbau eines zu sendenden Datenpakets zuständig.

Tabelle B.2: Beschreibung der Methoden der Klasse *Command*

Klasse *Command*	
Methode	**Beschreibung**
ToString	Die Methode *ToString* wird überschrieben und gibt die Befehlsbezeichnung als Zeichenkette zurück.
	Rückgabewert: Die Befehlsbezeichnung als String.

Tabelle B.3: Beschreibung der Methoden/Ereignisse der Klasse *InfoWindow*

Klasse *InfoWindow*	
Methode/Ereignis	**Beschreibung**
CheckReceivedData OfTransmitCommands ForDeviceData	Mit dieser Methode werden die empfangenen Datenpakete bzw. die Antwort des Hoppers auf einen zuvor gesendeten Befehl weiterverarbeitet. Entspricht die Antwort des Hoppers der erwarteten Antwort, wird der nächste zu sendende Befehl ausgeführt. Andernfalls erscheint eine Meldung.
Clear_All_Group Boxes_Textboxes	Den Inhalt der Textfelder von allen Gruppen des Informationsfensters löschen.
GroupBox_DeviceData _Clear_Textboxes	Den Inhalt der Textfelder der Gruppe *Device Data* bzw. der Gruppen *General* und *Variable Set* löschen.
GroupBox_General _Clear_Textboxes	Den Inhalt der Textfelder der Gruppe *General* löschen.
GroupBox_HighLow LevelStatus_Clear	Den Inhalt und die Hintergrundfarbe der Textfelder der Gruppe *High/Low Level Status* löschen. Zusätzlich alle Bezeichner (Labels) deaktivieren.
GroupBox_Hopper StatusRegister1_Clear	Die Bezeichner der Gruppe *Hopper Status Register 1* deaktivieren.
GroupBox_Hopper StatusRegister2_Clear	Die Bezeichner der Gruppe *Hopper Status Register 2* deaktivieren.
GroupBox_NVData _Clear_Textboxes	Den Inhalt der Textfelder der Gruppe *NV Data* löschen.
GroupBox_ TestHopper_Clear	Die Bezeichner der Gruppe *Test Hopper* bzw. der Gruppen *Hopper Status Register 1* und *2* deaktivieren.
GroupBox_Variable Set_Clear_Textboxes	Den Inhalt der Textfelder der Gruppe *Variable Set* löschen.

NVData	Die Methode *NVData* füllt die Textfelder der Gruppe *NV Data* mit den jeweiligen Werten aus.
	Parameter: Empfangene Datenbytes des Befehls *ReadData Block* und das gesendete Datenbyte für die Block-Nr.
RegisterHighLow LevelStatus	Die Methode *RegisterHighLowLevelSensor* stellt den Status der Sensoren für die Ermittlung des Füllstands des Hoppers in der Gruppe *High/Low Level Status* dar.
	Parameter: Empfangene Datenbytes des Befehls *ReqPayout HighLowStatus*.
RegisterTestHopper	Die Methode *RegisterTestHopper* aktiviert bzw. deaktiviert die Bezeichner der Gruppe *Test Hopper* bzw. der Gruppen *Hopper Status Register 1* und *2* in Abhängigkeit der empfangenen Datenbytes des Befehls *TestHopper*.
	Parameter: Empfangene Datenbytes des Befehls *TestHopper.*
TextBox_ SupplyVoltage_ TextChanged	Ändert sich der Inhalt des Textfelds **Supply Voltage** der Gruppe *Variable Set*, wird die Hintergrundfarbe des Textfelds entsprechend geändert.
ToolStripMenu ItemHelp_Click	Mit der Methode *ToolStripMenuItemHelp_Click* wird beim Betätigen des Menüelements **Help** das dazugehörige Hilfe-Fenster angezeigt.
TransmitCommands ForDeviceData	Die Methode *TransmitCommandsForDeviceData* beinhaltet die Befehle, welche für das Abfragen der Informationen bzw. der Status-/Fehlermeldungen des Hoppers zuständig sind. Dabei beginnt die Befehlskette mit dem Befehl *SimplePoll*, um zu prüfen, ob der Hopper überhaupt am Bus angeschlossen ist. Alle weiteren Befehle werden von der Methode *CheckReceived DataOfTransmitCommandsForDeviceData* ausgegeben, da die Antworten des Hoppers zunächst auf Richtigkeit geprüft und zusätzlich ausgewertet werden müssen.
VariableSet	Den Inhalt der Textfelder der Gruppe *Variable Set* mit Werten ausfüllen.
	Parameter: Empfangene Datenbytes des Befehls *ReqBuild Code*.

Tabelle B.4: Beschreibung der Methoden/Ereignisse der Klasse *MainWindow*

Klasse *MainWindow*	
Methode/Ereignis	**Beschreibung**
Button_Close SerialPort_Click	Die im Kombinationsfeld der Gruppe *COM Port* ausgewählte serielle Schnittstelle schließen.
Button_Command Execute_Click	Die Methode sendet über die serielle Schnittstelle mittels der Methode *TransmitCommand* den aus dem Kombinationsfeld **Command** ausgewählten Befehl an den Hopper.
Button_Dispense Coins_Click	Mit *Button_DispenseCoins_Click* wird der erste Befehl aus der Befehlskette *TransmitCommandDispenseCoins* gesendet. Alle weiteren Befehle werden von der Methode *CheckReceived DataOfTransmitCommandDispenseCoins* ausgegeben, da die Antworten des Hoppers zunächst auf Richtigkeit geprüft und ausgewertet werden müssen.
Button_Dispense Coins_Enable	Die Methode wird benötigt, um die Taste **Dispense** aus einem anderen Thread zu aktivieren. Die Taste wird bei ihrer Betätigung deaktiviert und, nachdem die Befehlskette *Transmit CommandDispenseCoins* durchgelaufen ist, wieder aktiviert.
Button_Dispense Coins_Enabled	Die Methode *Button_DispenseCoins_Enabled* wird zum Abfragen des Status der Taste **Dispense** benötigt.
	Rückgabewert: Wahr, wenn die Taste aktiviert ist, sonst falsch.
Button_Emergency Stop_Click	Die Methode *Button_EmergencyStop_Click* sendet über die serielle Schnittstelle mittels der Methode *TransmitCommand* den Befehl *EmergencyStop* an den Hopper. Der Befehl dient zur sofortigen Abschaltung der Münzausgabe.
Button_Hopper Info_Click	Mit *Button_HopperInfo_Click* wird das Informationsfenster angezeigt, in dem alle wichtigen Informationen des Hoppers angezeigt werden.
Button_Hopper Info_Enable	Die Methode wird benötigt, um die Taste **Hopper Info** aus einem anderen Thread zu aktivieren. Die Taste wird bei ihrer Betätigung deaktiviert und, nachdem die Befehlskette *Transmit CommandsForDeviceData* durchgelaufen ist, wieder aktiviert.
Button_Hopper Info_Enabled	Die Methode *Button_HopperInfo_Enabled* wird zum Abfragen des Status der Taste **Hopper Info** benötigt.
	Rückgabewert: Wahr, wenn die Taste aktiviert ist, sonst falsch.

Button_Open SerialPort_Click	Die im Kombinationsfeld der Gruppe *COM Port* ausgewählte serielle Schnittstelle öffnen. Zuvor wird diese noch folgendermaßen konfiguriert: Baud- bzw. Bitrate=9600bit/s, 8N1, kein Handshake.
Button_Read_ IDNumber_Click	Die Methode *Button_Read_IDNumber_Click* sendet den Befehl *ReadDataBlock* an den Hopper.
Button_Write_ IDNumber_Click	Die Methode *Button_Write_IDNumber_Click* sendet den Befehl *WriteDataBlock* an den Hopper und speichert ggf. die zu sendende ID-Nr. in der Datei *ID-Number_list.txt* ab.
CheckReceived DataOf TransmitCommand DispenseCoins	Mit dieser Methode werden die empfangenen Datenpakete bzw. die Antwort des Hoppers auf einen zuvor gesendeten Befehl weiterverarbeitet. Entspricht die Antwort des Hoppers der erwarteten Antwort, wird der nächste zu sendende Befehl ausgeführt. Andernfalls erscheint eine Meldung.
Clear_All_Group Boxes_Textboxes	Den Inhalt der Textfelder von allen Gruppen des Hauptfensters löschen.
ComboBox_ BlockNumber_ SelectedIndexChanged	Ändert sich der Index des Kombinationsfelds **Block No.**, wird die Eigenschaft *data* bzw. das zu sendende Datenbyte des Objekts (Befehls) *ReadDataBlock* entsprechend geändert.
ComboBox_ Command_ SelectedIndexChanged	Ändert sich der Index des Kombinationsfelds **Command**, werden für einige Befehle bestimmte Aktionen ausgeführt. Z. B. werden die benötigten Datenbytes in Abhängigkeit von anderen Steuerelementen gesetzt.
ComboBox_ SerialPort_Update	Die Methode aktualisiert die Einträge im Kombinationsfeld der Gruppe *COM Port* und aktiviert bzw. deaktiviert die Tasten **Open** und **Close** sowie die Gruppen des Hauptfensters entsprechend.
	Parameter: COM-Anschlussnamen und Operation. Die Operation *add*, um den Anschlussnamen im Kombinationsfeld hinzuzufügen und *remove*, um den Anschlussnamen zu entfernen.
ComboBox_TransmitIDNumberInEuro_ SelectedIndexChanged	Ändert sich der Index des Kombinationsfelds **Coin Value**, indem eine andere Münzsorte ausgewählt wurde, wird in Abhängigkeit der ausgewählten Münzsorte die an den Hopper zu sendende ID-Nr. angepasst.
DEV_ BROADCAST_PORT	DEV_BROADCAST_PORT beinhaltet Informationen über die serielle Schnittstelle. Z. B. den COM-Anschlussnamen.

GroupBox_IDNumber _Clear_Textboxes	Den Inhalt der Textfelder der Gruppe *ID-Number* löschen.
GroupBox_Received Data_Clear_Textboxes	Den Inhalt der Textfelder der Gruppe *Received Data* löschen.
GroupBox_Traffic _Clear_Textboxes	Den Inhalt der Textfelder der Gruppe *Traffic* löschen.
GroupBoxes_Enable	Die Methode *GroupBoxes_Enable* aktiviert bzw. deaktiviert die Gruppen *Hopper*, *Traffic* und *Received Data*.
	Parameter: Wahr, um die Gruppen zu aktivieren. Falsch, um die Gruppen zu deaktivieren.
MainWindow_ Activated	Wird das Hauptfenster angezeigt, wird die Taste *Alt* ausgeführt. Damit für den Anwender deutlich wird, dass die Anwendung auch über Tastenbefehle gesteuert werden kann, erscheinen Unterstriche unter dem Buchstaben des jeweiligen Steuerelements. **Hinweis:** Die Anwendung kann bereits von vornherein über die Steuerelemente bedient werden.
MainWindow_ FormClosing	Anweisungen, welche durchgeführt werden müssen, bevor das Hauptfenster geschlossen wird. Z. B. eine geöffnete serielle Schnittstelle schließen.
MainWindow_KeyUp	Die Steuerelemente des Hauptfensters anhand von Tastenbefehlen ausführen.
MainWindow_Load	Anweisungen, welche durchgeführt werden, bevor das Hauptfenster angezeigt wird.
OnTimedEvent	Das Ereignis, welches durch den Timer ausgelöst wird, um den zuletzt gesendeten Befehl erneut zu senden.
RadioButton_ASCII _CheckedChanged	*RadioButton_ASCII_CheckedChanged* stellt die Anzeige der Textfelder der Gruppe *Received Data* in ASCII um.
RadioButton_DEC _CheckedChanged	*RadioButton_DEC_CheckedChanged* stellt die Anzeige der Textfelder der Gruppe *Received Data* in Dezimal um.
RadioButton_HEX _CheckedChanged	*RadioButton_HEX_CheckedChanged* stellt die Anzeige der Textfelder der Gruppe *Received Data* in Hexadezimal um.
ReceivedSerialData	Die Methode *ReceiveSerialData* stellt die empfangenen Daten in den Gruppen *Traffic* und *Received Data* dar.
SetReceivedData	Die Methode *SetReceivedData* setzt die zuletzt empfangenen Daten in die Textfelder der Gruppe *Received Data* ein.

TextBox_Header _TextChanged	Den Inhalt des Textfelds **Header** der Gruppe *Received Data* interpretieren.
TextBox_ ReceivedIDNumber _TextChanged	Ändert sich der Inhalt des Textfelds **Received ID-No.**, indem eine neue ID-Nr. empfangen wurde, wird das Textfeld der anzuzeigenden Münzsorte aktualisiert. Des Weiteren wird das Kombinationsfeld für die Auswahl der zu verändernden Münzsorte des Hoppers aktualisiert. Es werden alle Münzsorten hinzugefügt – mit Ausnahme der bereits eingestellten.
TextBox_ TransmitIDNumber _TextChanged	Ändert sich der Inhalt des Textfelds **Transmit ID-No.**, wird die Eigenschaft *data* bzw. die zu sendenden Datenbytes des Objekts (Befehls) *WriteDataBlock* entsprechend geändert.
ToolStripMenu ItemHelp_Click	Mit der Methode *ToolStripMenuItemHelp_Click* wird beim Betätigen des Menüelements **Help** das dazugehörige Hilfe-Fenster angezeigt.
TransmitCommand DispenseCoins	Die Methode *TransmitCommandDispenseCoins* beinhaltet die Befehle, die bis zur Ausführung des eigentlichen Befehls zur Auszahlung von Münzen *DispenseHopperCoins* gesendet werden müssen. Dabei beginnt die Befehlskette mit dem Befehl *SimplePoll*, um zu prüfen, ob der Hopper überhaupt am Bus angeschlossen ist. Alle weiteren Befehle werden von der Methode *CheckReceivedDataOfTransmitCommandDispense Coins* ausgegeben, da die Antworten des Hoppers zunächst auf Richtigkeit geprüft und ausgewertet werden müssen. Z. B. Auswertung, ob eine Verschlüsselung erforderlich ist. Wenn eine unverschlüsselte Version des Hoppers vorliegt, wird eine Verschlüsselung der Datenbytes des Befehls *Dispense HopperCoins* nicht benötigt.
	Parameter: Befehl als Zeichenkette.
WndProc	Die Methode *WndProc* wird benötigt, um Betriebssystemmeldungen zu behandeln. Hier wird zzt. nur die Meldung, dass der USB-zu-Seriell-Adapter entfernt bzw. angeschlossen wurde, behandelt. Die Methode aktualisiert außerdem das Kombinationsfeld der Gruppe *COM Port.*

Tabelle B.5: Beschreibung der Methoden/Ereignisse der Klasse *MyApplication*

Klasse *MyApplication*	
Ereignis	**Beschreibung**
Startup	Das Ereignis *Startup* wird ausgelöst während die Anwendung startet und zwar noch bevor das Startformular (Hauptfenster) erstellt wird. Beim Start wird geprüft, ob serielle Schnittstellen vorhanden sind. Wird keine serielle Schnittstelle gefunden, erscheint ein Hinweis und der Anwender hat zwei Möglichkeiten. Entweder kann eine serielle Schnittstelle angeschlossen und anschließend die Taste *Wiederholen* betätigt werden, damit die Anwendung gestartet wird oder die Taste *Abbrechen* wird betätigt, um die Anwendung zu beenden.

Tabelle B.6: Beschreibung der Methoden der Klasse *Received*

Klasse *Received*	
Methode	**Beschreibung**
Calculate SerialNumber	Die Methode *CalculateSerialNumber* berechnet aus den empfangenen Datenbytes die Seriennummer des Hoppers.
	Rückgabewert: Die Seriennummer des Hoppers.
Databytes ToASCII	Die Methode *DatabytesToASCII* wandelt die empfangenen Datenbytes in Zeichen um und fasst diese in einer Zeichenkette zusammen.
	Rückgabewert: Die empfangenen Datenbytes in ASCII-Format.
Databytes ToString	Die Methode *DatabytesToString* liefert die empfangenen Datenbytes in Form einer Zeichenkette, deren Elemente durch Leerzeichen getrennt sind.
	Rückgabewert: Die empfangenen Datenbytes als String.
Transmitted ToString	Die Methode *TransmittedToString* liefert das gesendete Datenpaket, welches als Echo empfangen wird, in Form einer Zeichenkette, deren Elemente durch Leerzeichen getrennt sind.
	Rückgabewert: Das Echo des gesendeten Datenpakets als String.
ToString	Die Methode *ToString* fasst das empfangene Datenpaket bzw. die Antwort des Hoppers in Form einer Zeichenkette zusammen, deren Elemente durch Leerzeichen getrennt sind.
	Rückgabewert: Das empfangene Datenpaket in Form einer Zeichenkette.

Tabelle B.7: Beschreibung der Methoden der Klasse *Transmit*

Klasse *Transmit*	
Methode	**Beschreibung**
Calculate Checksum	Die Methode *CalculateChecksum* berechnet für das zu sendende Datenpaket die benötigte Prüfsumme.
	Rückgabewert: Die Prüfsumme als Byte.
Length	Die Methode *Length* ermittelt die Länge des zu sendenden Datenpakets.
	Rückgabewert: Die Länge des Datenpakets als Byte.
ToBytearray	Die Methode *ToBytearray* fasst alle zu sendenden Bytes in einem Bytearray bzw. Datenpaket zusammen.
	Rückgabewert: Das zu sendende Datenpaket in Form eines Bytearrays.
ToString	Die Methode *ToString* fasst das zu sendende Datenpaket in Form einer Zeichenkette zusammen, deren Elemente durch Leerzeichen getrennt sind.
	Rückgabewert: Das zu sendende Datenpaket in Form einer Zeichenkette.

B.2 Module

Tabelle B.8: Auflistung der Module und deren Beschreibung

Modul	Beschreibung
Calculate	Das Modul *Calculate* beinhaltet Methoden, welche zur Berechnung der ID-Nr. des Automatenherstellers und der Prüfsumme benötigt werden.
CommandList	Das Modul *CommandList* beinhaltet Methoden zum Umgang mit der Befehlsliste.
Cryptography	Das Modul *Cryptography* beinhaltet die Methode *DES_crypt* mit der ein 64-Bit Datenblock entweder ver- oder entschlüsselt werden kann. Diese Methode wird benötigt, um den Auszahlungsbefehl *Dispense Hopper Coins* mit den zu sendenden Datenbytes, welche zuvor mittels des DES-Algorithmus verschlüsselt werden, an einen Hopper zu senden.
FileHandling	Das Modul *FileHandling* beinhaltet Methoden, mit denen die Bearbeitung von bzw. der Schreib- und Lesezugriff auf Dateien behandelt wird.
GlobalConst	Das Modul *GlobalConst* beinhaltet die globalen Konstanten des gesamten Projekts.
GlobalVar	Das Modul *GlobalVar* beinhaltet die globalen Variablen des gesamten Projekts.
MyMessages	Das Modul *MyMessages* beinhaltet die globalen Konstanten für die Meldungen.
MySerialPort	Das Modul *MySerialPort* beinhaltet Methoden zum Öffnen, Schließen, Konfigurieren und Erfassen von Pegeländerungen der seriellen Schnittstellen.
Transceive	Das Modul *Transceive* beinhaltet zwei Methoden. Einerseits die Methode zur Aufbereitung der zu sendenden Daten, in dem die zu sendenden Bytes dem Objekt *telegram_TxD* der Klasse *Transmit* zugewiesen werden. Anschließend erfolgt die Weiterleitung des Objekts zur seriellen Schnittstelle. Andererseits die Methode zur Abholung der am Empfangspuffer der seriellen Schnittstelle anliegenden Bytes und deren Zuweisung am Objekt *telegram_RxD* der Klasse *Received*.

Tabelle B.9: Beschreibung der Methoden des Moduls *Calculate*

Modul *Calculate*	
Methode	**Beschreibung**
BytesToIDNr	Die Methode *BytesToIDNr* berechnet aus den ersten vier der acht empfangenen Datenbytes die ID-Nr. und liefert diese als String zurück. Berechnung: ID-Nr. = [1es Byte] * 2^{24} + [2es] * 2^{16} + [3es] * 2^{8} + [4es] * 2^{0} Die empfangenen Datenbytes befinden sich im NV Memory Block-Nr. 0, welche mittels des Befehls *ReadDataBlock* empfangen werden.
	Parameter: Empfangene Datenbytes.
	Rückgabewert: Berechnete ID-Nr. als String oder ID_NOT_FOUND.
Checksum	Die Methode *Checksum* berechnet aus den zu sendenden oder empfangenen Daten die Prüfsumme und liefert diese als ein Byte zurück.
	Parameter: Daten als Bytearray.
	Rückgabewert: Prüfsumme als Byte.
IDNrToBytes	Die Methode *IDNrToBytes* berechnet aus der ID-Nr. die vier zu sendenden Bytes und liefert diese als ein Bytearray zurück. Berechnung: [1es Byte] = ID-Nr. \ 2^{24} [2es Byte] = (ID-Nr. mod 2^{24}) \ 2^{16} [3es Byte] = (ID-Nr. mod 2^{16}) \ 2^{8} [4es Byte] = (ID-Nr. mod 2^{8}) \ 2^{0}
	Parameter: ID-Nr. als Integer.
	Rückgabewert: Vier berechnete Datenbytes als ein Bytearray.

Tabelle B.10: Beschreibung der Methoden des Moduls *CommandList*

Modul *CommandList*	
Methode	**Beschreibung**
GetList	Mit der Methode *GetList* werden die Befehle für den Hopper aus der Datei *command_list* geladen. Wenn die Datei *command_list* existiert, werden die Objekte bzw. die Befehle aus der Datei geladen. Andernfalls werden die zuvor festgelegten Objekte erstellt und in der Datei gespeichert. Befehle aus einer Datei zu laden hat den Vorteil, dass diese bei Bedarf aus der Anwendung heraus verändert oder neue Befehle hinzugefügt bzw. alte entfernt werden können.

Tabelle B.11: Beschreibung der Methoden des Moduls *Cryptography*

Modul *Cryptography*	
Methode	**Beschreibung**
DES_crypt	Die Methode *DES_crypt* liefert in Abhängigkeit des Parameters *mode* entweder den ver- oder den entschlüsselten 64-Bit Datenblock. Dabei wird der 64-Bit Datenblock bzw. der Parameter *dataBlock64* mit dem 64-Bit Schlüssel (*key64)* entweder ver- oder entschlüsselt.
	Parameter: Den 64-Bit Schlüssel (*key64*) als ein acht Byte langes Bytearray. Den zu ver- bzw. entschlüsselnden 64-Bit Datenblock (*dataBlock64*) als ein acht Byte langes Bytearray. Den Modus (*mode*). Für Verschlüsselung *encrypt* und für Entschlüsselung *decrypt*.
	Rückgabewert: Den ver- bzw. entschlüsselten 64-Bit Datenblock als ein acht Byte langes Bytearray.

Tabelle B.12: Beschreibung der Methoden des Moduls *FileHandling*

Modul *FileHandling*	
Methode	**Beschreibung**
ReadFrom IDNumberList	Die Methode *ReadFromIDNumberList* liest die zuletzt eingetragene ID-Nr. aus der Datei *ID-Number_list.txt* aus.
	Rückgabewert: Die zuletzt eingetragene bzw. vergebene ID-Nr.
WriteInto IDNumberList	Die Methode *WriteIntoIDNumberList* schreibt den übergebenen Text bzw. die ID-Nr. in die Datei *ID-Number_list.txt*. Die Speicherung der ID-Nr. in eine Datei dient dazu, dass Hopper, die keine ID-Nr. zugewiesen bekommen haben, eine eindeutige ID-Nr. zugewiesen bekommen. Es dürfen keine doppelten ID-Nrn. vergeben werden.
	Parameter: Die ID-Nr., welche in die Datei geschrieben werden soll.
WriteInto Logfile	Die Methode *WriteIntoLogfile* schreibt den übergebenen Text in eine Datei. Die Datei dient als Logdatei, in der bestimmte Programmausführungen dokumentiert werden. Als Dateiname dient das aktuelle Datum des Betriebssystems im Format *yyyy_MM_dd* mit der Dateierweiterung *.log*. Ist die Datei bereits vorhanden, wird bei jedem Aufruf der Methode nur eine neue Zeile hinzugefügt. Da die Datei als Logdatei dient, wird in jeder neuen Zeile die aktuelle Systemzeit im Format *HH:mm:ss fffff* vorweggeschrieben. Format *yyyy_MM_dd*: Jahr_Monat_Tag Format *HH:mm:ss fffff*: Stunde(0-23):Minuten:Sekunden Millisekunden
	Parameter: Der Text, der in die Logdatei geschrieben werden soll.

Tabelle B.13: Beschreibung der Methoden des Moduls *MySerialPort*

Modul *MySerialPort*	
Methode	**Beschreibung**
Close	Mit *Close* wird der Port der seriellen Schnittstelle, sofern diese geöffnet ist, geschlossen.
Config	Mit *Config* wird die serielle Schnittstelle konfiguriert.
	Parameter: COM-Anschlussnamen.
GetPortNames	Die Methode *GetPortNames* liefert alle Namen der am Rechner vorhandenen seriellen Schnittstellen.
	Rückgabewert: Die Anschlussnamen als ein String Array.
OnTimedEvent	Das Ereignis *OnTimedEvent* wird ausgelöst, wenn die eingestellte Zeit (Intervall) des Timers abgelaufen ist. Dieses Ereignis wird benötigt, um den zuletzt gesendeten Befehl erneut zu senden.
Open	Mit *Open* wird der Port der seriellen Schnittstelle geöffnet, sofern diese nicht bereits geöffnet wurde.
	Rückgabewert: Wahr, wenn der Port geöffnet ist oder geöffnet werden konnte. Falsch, wenn der Port nicht geöffnet werden kann.
SerialPort_ DataReceived	Das Ereignis *SerialPort_DataReceived* wird ausgelöst, wenn am Empfangspuffer der seriellen Schnittstelle die zuvor festgelegte Anzahl an Bytes empfangen wurde. Die Anzahl der zu empfangenen Bytes wird mittels der Methode *ReceivedBytesThreshold* festgelegt, bevor ein Befehl gesendet wird.
SerialPort_ PinChanged	Das Ereignis *SerialPort_PinChanged* wird ausgelöst, wenn eine Pegeländerung an einem Eingangspin der seriellen Schnittstelle stattgefunden hat. Dieses Ereignis wird für das Abfragen der Signalleitung DSR (Data Set Ready) benötigt. Die Signalleitung DSR (Pin 6 am COM-Port) wird als Eingang benötigt, um festzustellen, ob der Adapter angeschlossen ist. Wird der Adapter während der Programmlaufzeit entfernt, erscheint eine Meldung. Wird der Adapter angeschlossen, erscheint ebenfalls eine Meldung.
TransmitData	Die Methode *TransmitData* schreibt das zu sendende Datenpaket in Form eines Bytearrays am Ausgangspuffer der seriellen Schnittstelle.
	Parameter: Das zu sendende Datenpaket als Bytearray. Die Anzahl der zu empfangenen Datenbytes, um die Anzahl der Bytes festzulegen, welche am Empfangspuffer vorhanden sein müssen, bevor das *DataReceived*-Ereignis ausgelöst wird.

Tabelle B.14: Beschreibung der Methoden des Moduls *Transceive*

Modul *Transceive*	
Methode	**Beschreibung**
ReadData	Die Methode *ReadData* dient zur Abholung der am Empfangspuffer der seriellen Schnittstelle anliegenden Bytes und deren Zuweisung am Objekt *telegram_RxD* der Klasse *Received*.
Transmit Command	Die Methode *TransmitCommand* dient zur Aufbereitung des zu sendenden Datenpakets und übergibt dieses der Methode *TransmitData* des Moduls *MySerialPort*.
	Parameter: Den zu sendenden Befehl.

C RS232-zu-*ccTalk*-Adapter

C.1 Schaltplan

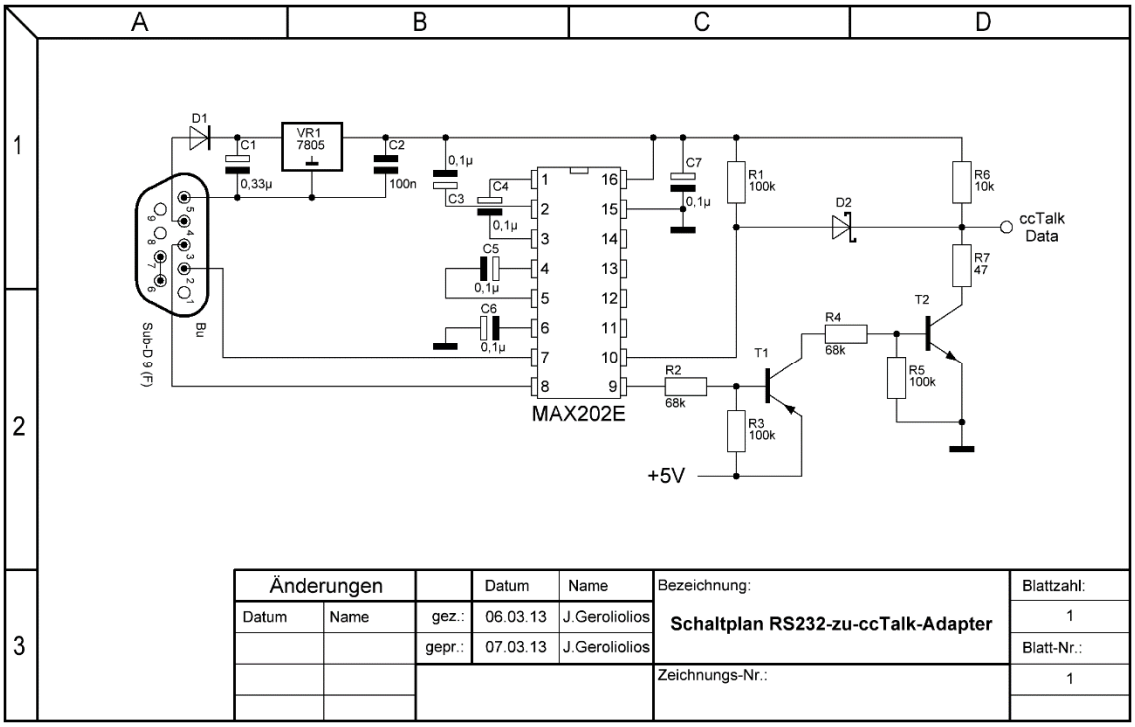

Bild C.1: Schaltplan des RS232-zu-*ccTalk*-Adapters

C.2 Stückliste

Tabelle C.1: Stückliste für die Platine des RS232-zu-*ccTalk*-Adapters

Pos.	Bauteil	Bezeichnung	Menge
1	Bu	Sub-D 9 (F), Buchse	1
2	C1	Elko (alternativ Tantal), 0,33 µF, 35 V, RM=2,54	1
3	C2	Folienkondensator, 100 nF, RM=2,54	1
4	C3, C4, C5, C6, C7	Tantal-Kondensator, 0,1 µF, 35 V, RM=2,54	5
5	D1	Diode, BAT85	1
6	D2	Schottky-Diode, BAT54	1
7	IC	+5 V RS-232 Transceiver, MAX202E	1
8	R1, R3, R5	Widerstand 0204, 100 kΩ	3
9	R2, R4	Widerstand 0204, 68 kΩ	2
10	R6	Widerstand 0204, 10 kΩ	1
11	R7	Widerstand 0204, 47 Ω	1
12	T1	BC327, PNP-Transistor	1
13	T2	BC546, NPN-Transistor	1

C.3 Leiterplattenlayout - Bestückungsseite

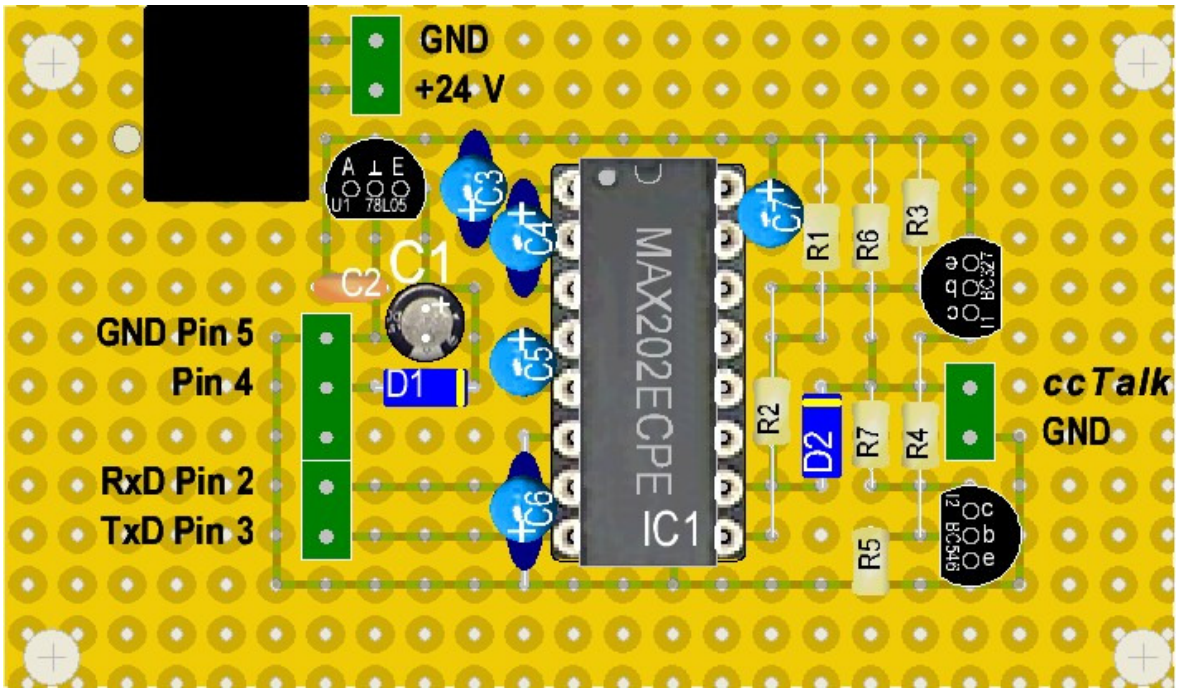

Bild C.2: Leiterplattenlayout des RS232-zu-*ccTalk*-Adapters - Bestückungsseite

C.4 Leiterplattenlayout - Lötseite (gewendet)

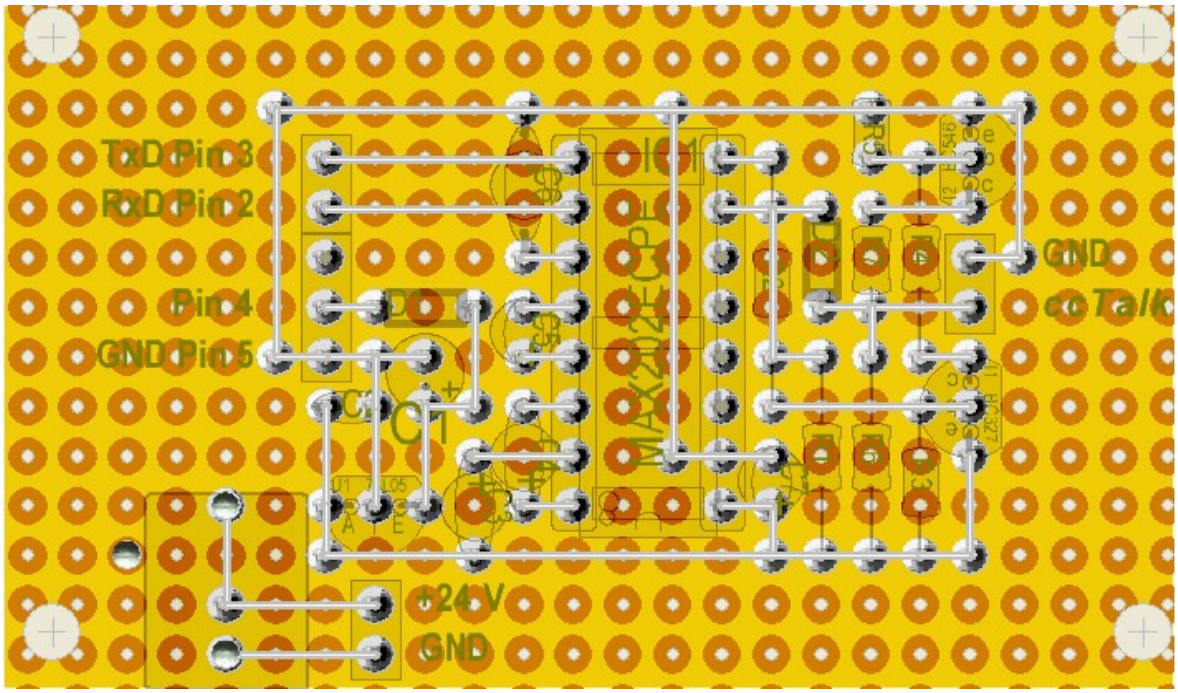

Bild C.3: Leiterplattenlayout des RS232-zu-*ccTalk*-Adapters - Lötseite (gewendet)

Versicherung über die Selbstständigkeit

Hiermit versichere ich, dass ich die vorliegende Arbeit im Sinne der Prüfungsordnung nach §16(5) APSO-TI-BM ohne fremde Hilfe selbstständig verfasst und nur die angegebenen Hilfsmittel benutzt habe. Wörtlich oder dem Sinn nach aus anderen Werken entnommene Stellen habe ich unter Angabe der Quellen kenntlich gemacht.

Hamburg, 23. Mai 2013

Ort, Datum Unterschrift