

Kalibriertool für Drucktransmitter



Diplomarbeit 2004
Informatik TS

Marco Di Menco

Inhaltsverzeichnis

1	Einführung	1
2	Planung	2
2.1	Aufgabenstellung	2
2.2	Anforderungsspezifikation.....	3
2.2.1	Einführung.....	3
2.2.2	Ziele	3
2.2.3	Einsatz.....	5
2.2.4	Funktionen	6
2.2.5	Schnittstellen	7
2.2.6	Leitungsanforderung	7
2.2.7	Daten/Datenbank.....	7
2.2.8	Weitere Merkmale	7
2.3	Zeitplan.....	8
2.3.1	Übersicht	8
2.3.2	Soll-Ist Vergleich	9

3	Analyse	10
3.1	Übersicht	10
3.2	Systemgrenze	10
3.3	Use Cases	11
3.3.1	Use Case Diagramm	11
3.3.2	Beschreibung der Aktoren	13
3.3.3	Use Case UploadGeraetedaten	13
3.3.4	Use Case DownloadGeraetedaten	14
3.3.5	Use Case ErstelleNeuesGeraet	14
3.3.6	Use Case AendereGeraet	15
3.3.7	Use Case LoescheGeraet	15
3.3.8	Use Case KonfiguriereHW	16
3.3.9	Use Case EroeffneProjekt	16
3.3.10	Use Case LadeProjekt	17
3.3.11	Use Case SpeichereProjekt	17
3.3.12	Use Case SchliesseProjekt	18
3.3.13	Use Case zeigeKalibrierzertifikat	18
3.3.14	Use Case aktualisiereKalibrierzertifikate	19
3.4	System-Sequenzdiagramme	20
3.4.1	erstelleNeuesGeraet	20
3.4.2	aendereGeraet	20
3.4.3	loescheGeraet	21
3.4.4	konfiguriereHW	21
3.4.5	uploadGeraetedaten	22

3.4.6	downloadeGeraetedaten	22
3.4.7	eroeffneProjekt	23
3.4.8	ladeProjekt	23
3.4.9	speichereProjekt	24
3.4.10	schliesseProjekt	24
3.4.11	zeigeKalibrierzertifikat	25
3.4.12	aktualisiereKalibrierzertifikat	25
3.5	Konzeptionelles Modell	26
3.6	Kontrakte	27
3.6.1	erstelleNeuesGeraet	27
3.6.2	aendereGeraet	27
3.6.3	loescheGeraet	28
3.6.4	uploadeGeraetedaten	28
3.6.5	downloadeGeraetedaten	29
3.6.6	konfiguriereHW	29
3.6.7	eroeffneProjekt	30
3.6.8	ladeProjekt	30
3.6.9	speichereProjekt	30
3.6.10	schliesseProjekt	31
3.6.11	zeigeKalibrierzertifikat	31
3.6.12	aktualisiereKalibrierzertifikat	31
3.7	Review / Entscheidungen	32
3.7.1	Entscheidung	32

4	Design	33
4.1	Systemarchitektur	33
4.1.1	Zielsystem	33
4.1.2	Schnittstellen	33
4.1.3	Entwicklung	33
4.2	User Interface	34
4.2.1	Hauptfenster	34
4.2.2	Menu	34
4.2.3	Konfigurationsdialog	34
4.2.4	Gerätedialoge	35
4.3	Interaktionsdiagramme	36
4.3.1	erstelleDruckschalter	36
4.3.2	erstelleDrucktransmitter	36
4.3.3	aendereGeraet	37
4.3.4	downloadeGeraetedaten	38
4.3.5	uploadeGeraetedaten	38
4.3.6	loescheGeraet	39
4.3.7	eroeffneProjekt	39
4.3.8	oeffneProjekt	40
4.3.9	speichereProjekt	41
4.3.10	konfiguriereHW	42
4.3.11	initialisiereHW	43
4.3.12	Package Diagramm	44

4.4	Design Klassendiagramm	44
4.4.1	GUI	44
4.4.2	Application Logic	45
4.4.3	Hardware	46
4.4.4	Datenhaltung	46
5	Implementation	47

5.1	Serielle Schnittstelle	47
5.1.1	Threads	47
5.1.2	Thread-Synchronisierung	47
6	Test	48

6.1	Testmethode	48
6.2	Erreichung der Ziele	48
6.3	Einsatz	49
6.4	Funktionen	50
6.5	Schnittstellen	51
6.6	Daten/Datenbank	52
6.7	Weitere Merkmale	52
7	Schlusswort	53
8	Literatur- und Linkverzeichnis	54

8.1	Softwareengineering	54
8.2	Programmierung C++/MFC	54
8.3	Internetlinks	54

9	Anhang A	I
----------	-----------------	----------

9.1	CD.....	I
9.2	Arbeitspakete	II
9.2.1	Planung.....	II
9.2.2	Analyse	II
9.2.3	Design	III
9.2.4	Implementation.....	IV
9.2.5	Tests	IV
9.2.6	Dokumentation	V
9.3	Bestimmungen für die Diplomarbeit.....	VI
9.4	Protokolle der Betreuersitzungen	IX

10	Anhang B	XIII
-----------	-----------------	-------------

10.1	Dokumentation Programmieradapter „Digipack“	XIII
-------------	--	-------------

Anhang C	Sourcecode	zusätzliches Dokument
-----------------	-------------------	------------------------------

1 Einführung

Das digitale Zeitalter schreitet immer weiter. Es ist schon soweit, dass auch ich den kleinsten Geräten, in welchen bis vor kurzem noch reine Analogtechnik und Potentiometer in Feinmechanik eingesetzt wurden, heute ein Mikrokontroller steckt.

Dies ist auch bei der Firma Trafag so. Um den steigenden Anforderungen gerecht zu werden, mussten auch sie in ihren Geräten „Intelligenz“ für die Kompensationen einbauen. Diese Mikrokontroller werden hauptsächlich in der Produktion für die Eichung verwendet.

Diese Technologie möchte die Firma Trafag jetzt auch ihren Kunden zugänglich machen, dass diese auch kleine Korrekturen oder Einstellungen an den Geräten vornehmen können.

Als ich mit meiner Anfrage für eine Diplomarbeit bei der Firma Trafag auf offene Ohren stiess, entschloss ich mich die Herausforderung anzunehmen.

Es gilt eine Applikation zu entwickeln, die den Bedürfnissen der Firma Trafag und ihren Kunden gerecht würden. Demnach den mechanischen Schraubenzieher durch einen digitalen zu ersetzen.

2 Planung

2.1 Aufgabenstellung



Herr
Di Menco Marco
Bahnhofwiese 2
8712 Stäfa

Wald, 28. März 2004

Diplomarbeit 2004 (Informatik)

Sehr geehrter Herr Di Menco

Sie erhalten folgende Diplomarbeit zugeteilt, die Sie selbständig lösen möchten.

Kalibriertool für Drucktransmitter

Es soll ein Windows-Tool für die Rekalibrierung von Drucktransmittern der Firma Trafag AG erstellt werden, mit dem Kunden oder Vertretungen kleinere Korrekturen oder Einstellungen vornehmen können. Die Rekalibrierung erfolgt über einen von der Firma Trafag AG entwickelten Programmieradapter, der über die serielle Schnittstelle angesteuert wird.

Die Aufgabe umfasst folgende Punkte:

- Projektplanung
- Pflichtenheft
- Entwurf der Analyse
- Entwurf des Designs
- Implementieren der Problem domain
- Testen anhand des Pflichtenheftes
- Dokumentation aller Arbeitsschritte

Optional:

- Anzeigen von Digitalen Kalibrierzertifikaten (Mittels Seriennummer)
- Herunterladen neuer digitaler Kalibrierzertifikate über das Web (Live Update)

Eingesetzte Mittel:

- Betriebssysteme: Windows NT, Windows 2000
- Visual Studio.Net
- Programmieradapter der Firma Trafag AG

Ausgabe der Aufgabenstellung: 7. April 2004
Abgabe der 2 Dokumentationen: 17. August 2004

Mit freundlichen Grüßen



D. Venosta
Betreuer

2.2 Anforderungsspezifikation

2.2.1 Einführung

2.2.1.1 Zweck

Das Softwareprodukt wird eingesetzt um Druckmessgeräte im Feld, von Kunden oder Vertretungen der Firma Trafag AG, zu konfigurieren.

2.2.1.2 Referenzen

Dokumentation der Firma Trafag:

- Dokumentation des Programmieradapters („Digipack“)
- Geräte Dokumentationen EPS (Druckschalter)
- Geräte Dokumentationen ECO/EPT (Analoge Drucktransmitter¹)

2.2.2 Ziele

2.2.2.1 Allgemeine Beschreibung

Das zu entwickelnde Softwareprodukt ermöglicht seinen Benutzern Druckmessgeräte der Firma Trafag zu konfigurieren oder nachzukalibrieren.

Das Programm kann, mit Hilfe eines Programmieradapters („Digipack“), die Konfiguration der Druckmessgeräte lesen oder beschreiben.

2.2.2.2 Muss-Kriterien

- Erstellung einer grafischen Benutzeroberfläche
- Ansteuerung des Programmieradapters
 - o Die Software muss ein austauschbares Modul für die Programmieradapter-Kommunikation enthalten. Dieses Modul sollte so gekapselt sein, dass sie in anderen SW weiterverwendet werden kann.
 - o Implementierung des Modbus² Protokoll über serielle Schnittstelle, gemäss Spezifikationen der Trafag AG (Anhang B Dokumentation Programmieradapter „Digipack“)

¹ Drucktransmitter sind Geräte die Druck mittels einer Messzelle in ein elektrisches oder digitales Signal umformen.

² Modbus ist ein offenes serielles Kommunikationsprotokoll, das auf der Master-Slave-Architektur basiert.

- Implementierung verschiedener Geräteprofile
 - o Druckschalter
 - Uploaden der Schaltpunkt-Parameter
 - Anzeigen der Schaltpunkt-Parameter
 - Ändern/Berechnen der neuen Schaltpunkt-Parameter
 - Downloaden/Überprüfen der neuen Schaltpunkt-Parameter
 - Die Schaltpunktparameter sind:
 - Schalterpunkt 1 ON (0...xxbar)
 - Schalterpunkt 1 OFF (0...xxbar)
 - Schalterpunkt 1 Delay (1ms 10s)
 - Schalterpunkt 2 ON (0...xxbar)
 - Schalterpunkt 2 OFF (0...xxbar)
 - Schalterpunkt 2 Delay (1ms 10s)
 - o Analoge Drucktransmitter „Feinkalibrierung“
 - Uploaden der Kalibrierwerte
 - Anzeigen der Kalibrierwerte
 - Schrittweise Ändern/Berechnen neuen Kalibrierwerte
 - Downloaden/Überprüfen der neuen Kalibrierwerte
 - Die änderbaren Kalibrierwerte sind:
 - Nullpunkt Korrektur (1...498)
 - Spanne Korrektur (1...498) (Verstärkung)
 - Die anzuzeigenden Kalibrierwerte sind:
 - Seriennummer
 - Kalibriertemperatur
 - Temperaturkoeffizient des Nullpunktes (siehe Gerätedokumentation)
 - Temperaturkoeffizient der Spanne (siehe Gerätedokumentation)
 - Grobeinstellung der Spanne (30%,40%,70%,100%)
- Auslesen von Gerätedaten wie Seriennummer, Produktionsdatum usw.
 - o Die Seriennummer jedes Gerätes muss ausgelesen und angezeigt werden können.
- Speichern/Laden der Geräteparameter in eine/von einer Datei

2.2.2.3 Kann-Kriterien

- Anzeigen von digitalen Kalibrierzertifikaten (mittels Seriennummer).
- Herunterladen neuer, aktueller, digitaler Kalibrierzertifikate über das Web (eine Art Live Update).

2.2.2.4 Abgrenzungskriterien

- Das bestehende Kommunikations-Protokoll muss verwendet werden.
- Die Software des Programmieradapters ist nicht Bestandteil des Projektes und kann nicht geändert oder angepasst werden.

2.2.3 Einsatz

2.2.3.1 Anwendungsbereiche

Die Anwendung erfolgt in der Industrie oder im Labor und wird in den meisten Fällen durch technisches Personal bedient.

2.2.3.2 Zielgruppen

Die Zielgruppe sind Kunden und Vertretungen der Firma Trafag, die Druckschalter oder Drucktransmitter einsetzen.

2.2.3.3 Software

Die Applikation sollte auf einem Standard Windowssystem ohne zusätzliche Software (wie z.B. Microsoft Office) auskommen.

Die Menus und GUI¹-Elemente müssen auf Grund der international verbreiteten Anwendergruppe in Englisch verfasst werden.

Mit Ausnahme der Systemmenüs (Öffnen, Speichern, OK, Abbrechen) diese Funktionen können in der jeweils eingestellten Sprache des Betriebssystems erscheinen.

2.2.3.4 Hardware

Die Software muss auf einem Desktop-PC oder auf einem Notebook mit Intelarchitektur (mind. Pentium II 300MHz) laufen.

Voraussetzung sind die Betriebssysteme Windows 2000/XP.

2.2.3.5 Sicherheit

Keine spezifischen Anforderungen.

¹ GUI = Graphical User Interface (Benutzeroberfläche)

2.2.4 Funktionen

2.2.4.1 Konfigurieren der Kommunikationsschnittstelle

Es muss die Möglichkeit bestehen die Kommunikationsschnittstelle (Com-Port) auszuwählen.

Diese Einstellungen müssen abgespeichert werden.

2.2.4.2 Initialisieren des Programmieradapters („Digipack“)

Nach der Auswahl des Com-Ports muss getestet werden, ob der Programmieradapter am ausgewählten Port angeschlossen ist.

2.2.4.3 Erzeugen eines neuen Gerätes

Mit dieser Funktion wird ein Gerät der Arbeitsfläche hinzugefügt. Der Benutzer muss beim Erzeugen eines neuen Gerätes den Gerätetyp auswählen können. Mit dieser Auswahl wird die richtige Programmiermethode und das GUI mit dem Parametern ausgewählt.

2.2.4.4 Löschen von Geräten

Die erzeugten Geräte müssen wieder von der Arbeitsfläche entfernt werden können.

2.2.4.5 Upload von Gerätedaten

Mit dieser Funktion werden über den Programmieradapter die Geräteparameter aus dem Gerät gelesen.

2.2.4.6 Änderung von Gerätedaten

Mit dieser Funktion werden die Gerätedaten des gewählten Gerätes geändert.

2.2.4.7 Download von Gerätedaten

Diese Funktion dient dazu, die Geräteparameter über den Programmieradapter in das Gerät zu laden.

2.2.4.8 Speichern von Gerätedaten

Die Gerätedaten müssen in eine Datei gespeichert werden können.

2.2.4.9 Laden von Gerätedaten

Die abgespeicherten Gerätedaten müssen wieder aus einer Datei geladen werden können.

2.2.5 Schnittstellen

2.2.5.1 Benutzeroberfläche

Es muss eine vollgrafische Windowsapplikation erstellt werden. Das User Interface muss den gängigen Standards für Windows-Applikationen genügen.

2.2.5.2 Software

Keine Schnittstelle zu anderen Softwareprodukten (wie z.B. MS Office) mit Ausnahme der Pakete einer Windows Standardinstallation wie Internet Explorer usw.

2.2.5.3 Hardware

Serielle Schnittstelle, eventuell auch über Bluetooth oder USB (über Windows ComPort -Treiber).

Für die Programmierung wird ein Programmieradapter („DigiPack“) verwendet, der mit einer seriellen Schnittstelle (RS232) ausgerüstet ist.

2.2.5.4 Kommunikationsschnittstellen

Der Programmieradapter wird mit einem Modbus Protokoll gemäss Spezifikationen der Trafag AG angesteuert. Genauere Details sind im Anhang beigefügt.

Der Berechnungsalgorithmus für die CRC16-Checksumme wird von der Firma Trafag AG geliefert.

2.2.6 Leistungsanforderung

Keine speziellen Anforderungen bezüglich Leistungsfähigkeit.

2.2.7 Daten/Datenbank

Gerätedaten oder Gerätekonfigurationen müssen in eine Datei abgespeichert werden können.

2.2.8 Weitere Merkmale

2.2.8.1 Randbedingungen zur Entwicklung

Es ist eine MFC¹(Microsoft Foundation Classes) Anwendung in C++ zu entwickeln.

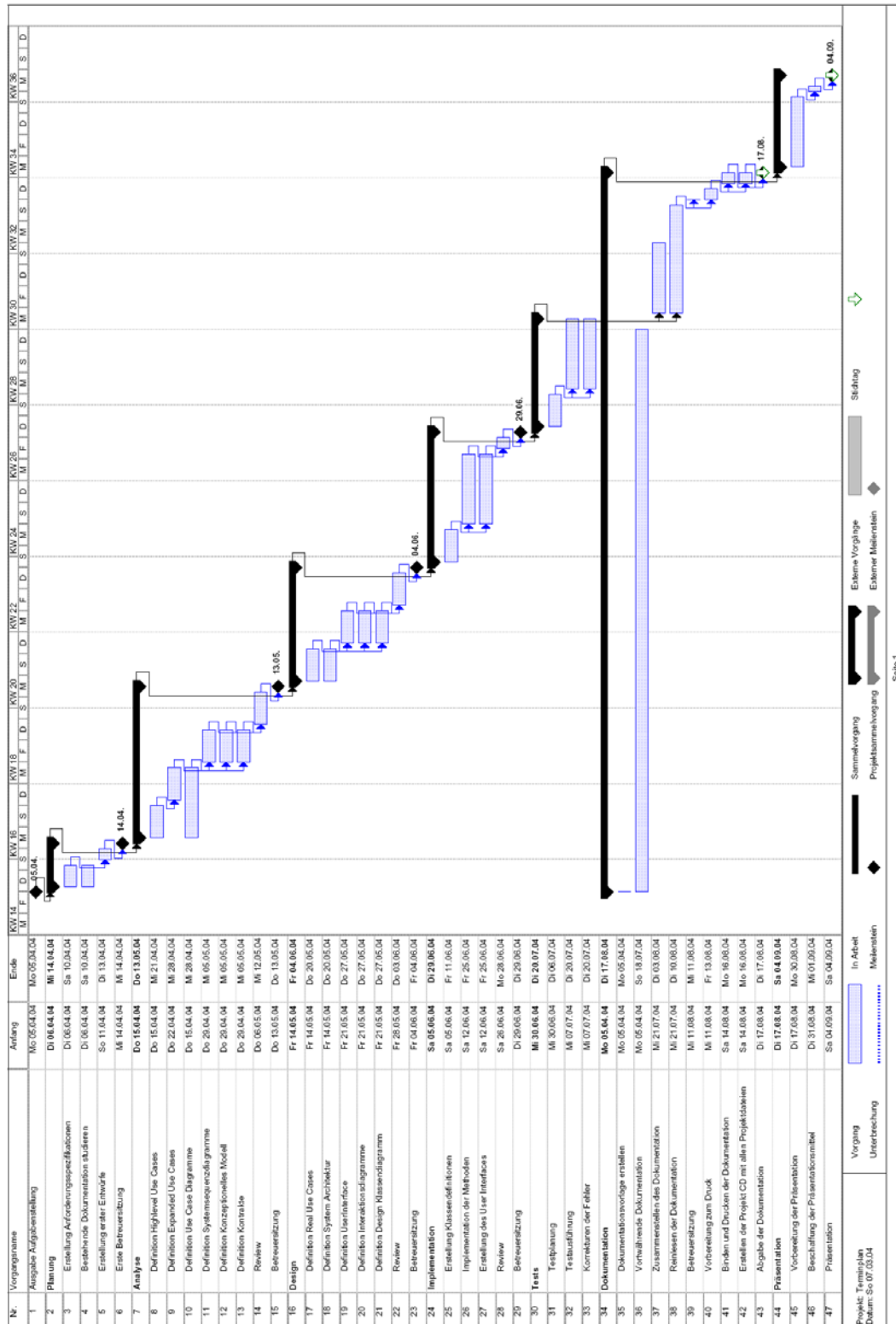
2.2.8.2 Randbedingungen im Betrieb

Der Programmieradapter muss für den Betrieb immer angeschlossen sein.

¹ MFC = Microsoft Foundation Classes ein objektorientiertes Framework auf C++ Basis der Windows API.

2.3 Zeitplan

2.3.1 Übersicht



2.3.2 Soll-Ist Vergleich

Projektphase		Beginn	Ende	Stundenaufwand
Planung	Soll	06.04.2004	14.04.2004	30 h
	Ist	06.04.2004	10.04.2004	25 h
Analyse	Soll	15.04.2004	13.05.2004	50 h
	Ist	12.04.2004	11.05.1004	40h
Design	Soll	14.05.2004	04.06.2004	40 h
	Ist	14.05.2004	17.06.2004	35 h
Implementation	Soll	05.06.2004	29.06.2004	20 h
	Ist	17.06.2004	06.07.2004	40 h
Test	Soll	30.06.2004	20.07.2004	10 h
	Ist	05.07.2004	12.08.2004	5 h
Dokumentation	Soll	05.04.2004	17.08.2004	50 h
	Ist	01.04.2004	17.08.2004	55 h
Total	Ist	-	-	200 h

Eine detaillierte Auflistung der einzelnen Arbeitspakete befindet sich im Anhang (9.2 Arbeitspakete).

3 Analyse

3.1 Übersicht

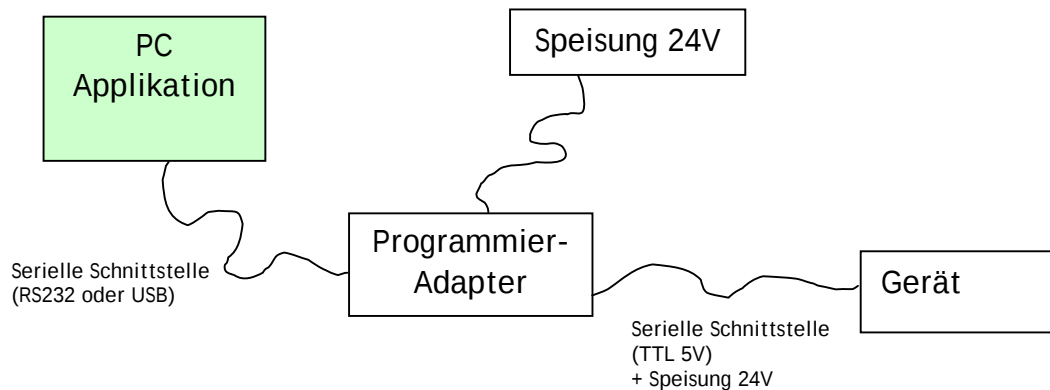


Abbildung 1

3.2 Systemgrenze

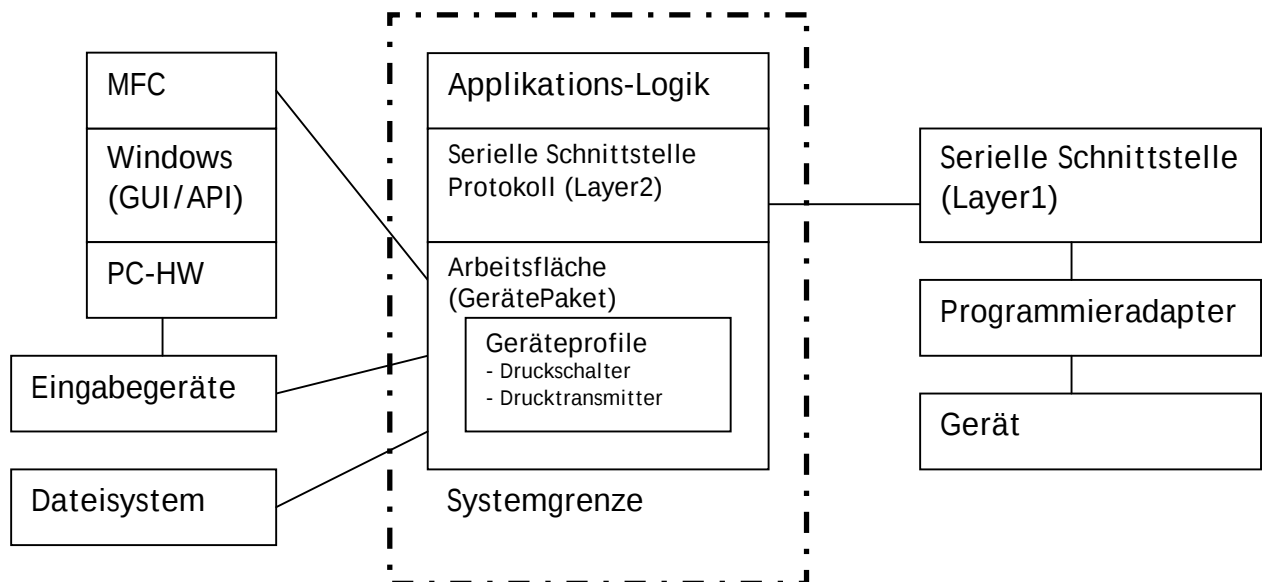


Abbildung 2

Dieses Projekt beschränkt sich auf die Applikationslogik, Geräteprofile, GUI, Datenhaltung und des Protokolls der seriellen Schnittstelle. Die Geräte-Profile bestehen aus zwei Profilen, aus einem Druckschalterprofil und einem Drucktransmitterprofil.

3.3 Use Cases

3.3.1 Use Case Diagramm

3.3.1.1 Geräte Use Cases

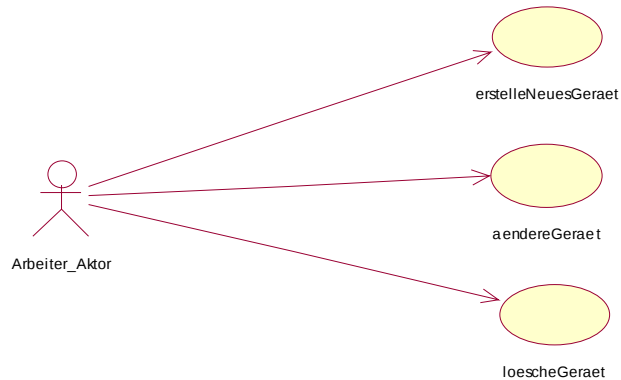


Abbildung 3

3.3.1.2 Hardware Use Cases

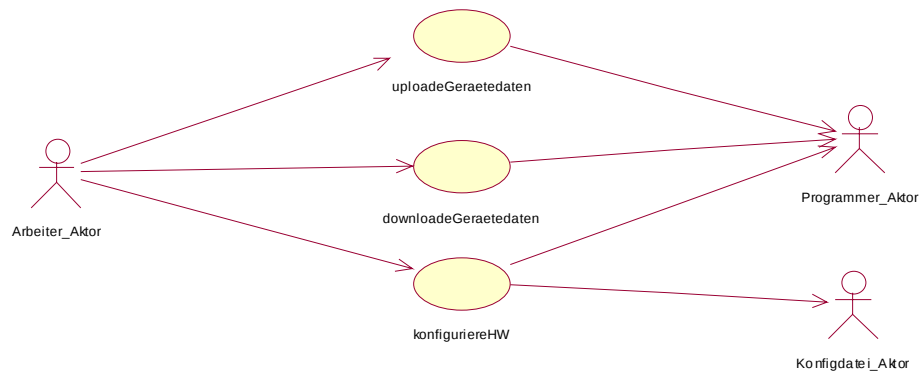


Abbildung 4

3.3.1.3 Projekt Use Cases

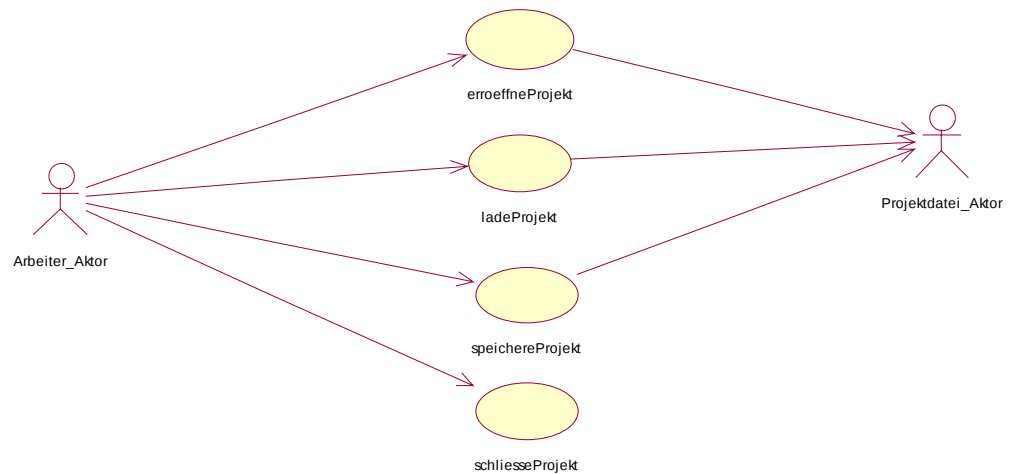


Abbildung 5

3.3.1.4 Optionale Use Cases

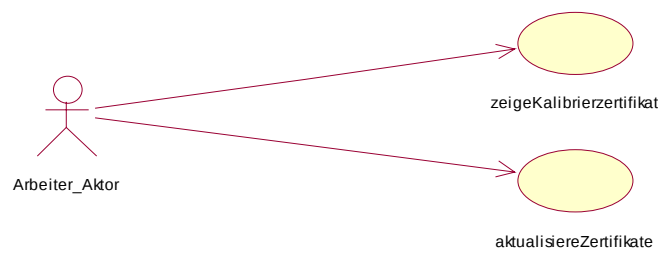


Abbildung 6

3.3.2 Beschreibung der Aktoren

Name	Beschreibung
 Arbeiter_Aktor	Löst das Erstellen, Ändern, Löschen von Geräten aus. Konfiguriert die Hardware und initiiert den Upload und Download von Gerätedaten.
 <i>Programmer_Aktor</i>	<i>Kommuniziert mit dem Gerät, ist für den Up/Download zuständig.</i>
 <i>Projektdatei_Aktor</i>	<i>Dieser Akteur nimmt das Projekt und die Gerätedaten auf.</i>
 <i>Konfigdatei_Aktor</i>	<i>Dieser Akteur nimmt Konfigurationsdaten auf.</i>

Aktiver Akteur (Initiator) / Passiver Akteur

3.3.3 Use Case UploadeGeraetedaten

Name	Beschreibung
Akteur(en)	<i>Arbeiter_Aktor, Programmer_Aktor</i>
Kurzbeschreibung	Dieser Use Case behandelt den Upload der Gerätekonfiguration.
Priorität	Essentiell, Hauptfunktion
Referenzen	2.2.4.5 Upload von Gerätedaten
Ablauf:	
Akteur-Aktionen	Systemantwort
1. uploadeGeraetedaten()	2. leseGeraetedaten()
3. erhalteGeraetedaten()	4. uploadErfolgreich

Aktiver Akteur (Initiator) / Passiver Akteur

3.3.4 Use Case DownloadeGeraeteden

Name	Beschreibung
Aktor(en)	Arbeiter_Aktor, Programmer_Aktor
Kurzbeschreibung	Dieser Use Case behandelt den Download der Gerätekonfiguration.
Priorität	Essentiell, Hauptfunktion
Referenzen	2.2.4.7 Download von Gerätedaten
Ablauf:	
Aktor-Aktionen	Systemantwort
1. downloadeGeraeteden()	2. speichereGeraeteden()
3. speichernErfolgreich()	4. DownloadErfolgreich

Aktiver Aktor (Initiator) / Passiver Aktor

3.3.5 Use Case ErstelleNeuesGeraet

Name	Beschreibung
Aktor(en)	Arbeiter_Aktor
Kurzbeschreibung	Dieser Use Case behandelt das Erstellen eines neuen Gerätes im Projekt.
Priorität	Essentiell, Hauptfunktion
Referenzen	2.2.4.3 Erzeugen eines neuen Gerätes
Ablauf:	
Aktor-Aktionen	Systemantwort
1. addGeraet()	2. GeraeteTypAuswahl
3. setGeraetetyp(Geraetetyp)	4. ErstellenErfolgreich

Aktiver Aktor (Initiator) / Passiver Aktor

3.3.6 Use Case AendereGeraet

Name	Beschreibung
Aktor(en)	Arbeiter_Aktor
Kurzbeschreibung	Dieser Use Case behandelt das Ändern eines Gerätes im Projekt.
Priorität	Essentiell, Hauptfunktion
Referenzen	2.2.4.6 Änderung von Gerätedaten
Ablauf:	
Aktor-Aktionen	Systemantwort
1. aendereGeraet()	2. sendeGeraeteDaten(daten)
3. setNeueGeraetedaten(daten)	4. AendernErfolgreich

Aktiver Aktor (Initiator) / Passiver Aktor

3.3.7 Use Case LoescheGeraet

Name	Beschreibung
Aktor(en)	Arbeiter_Aktor
Kurzbeschreibung	Dieser Use Case behandelt das Entfernen eines Gerätes aus dem Projekt.
Priorität	Essentiell
Referenzen	2.2.4.4 Löschen von Geräten
Ablauf:	
Aktor-Aktionen	Systemantwort
1. loescheGeraet()	2. bestaetigeLoeschen
3. bestaetigeLoeschen(OK)	4. LoeschenErfolgreich

Aktiver Aktor (Initiator) / Passiver Aktor

3.3.8 Use Case KonfiguriereHW

Name	Beschreibung
Aktor(en)	Arbeiter_Aktor, Konfigdatei_Aktor, Programmer_Aktor
Kurzbeschreibung	Dieser Use Case behandelt das Konfigurieren der Kommunikationsschnittstelle für den Programmieradapter und speichert diese in die Konfig-Datei.
Priorität	Essentiell
Referenzen	2.2.4.1 Konfigurieren der Kommunikationsschnittstelle
Ablauf:	
Aktor-Aktionen	Systemantwort
1. konfiguriereHW()	2. EingabeKonfiguration
3. setKonfiguration(Konfig)	4. initialisiereProgrammer(Einstellungen)
5. KonfigurationErfolgreich	6. speichereEinstellung
7. speichernErfolgreich	8. KonfigurationErfolgreich

Aktiver Aktor (Initiator) / Passiver Aktor

3.3.9 Use Case EroeffneProjekt

Name	Beschreibung
Aktor(en)	Arbeiter_Aktor, Projektdatetei_Aktor
Kurzbeschreibung	Dieser Use Case behandelt das Erstellen eines neuen Projektes
Priorität	Essentiell
Referenzen	-
Ablauf:	
Aktor-Aktionen	Systemantwort
1. EroeffneNeuesProjekt(Name)	2. ProjektErfolgreichErstellt()

Aktiver Aktor (Initiator) / Passiver Aktor

3.3.10 Use Case LadeProjekt

Name	Beschreibung
Aktor(en)	Arbeiter_Aktor, Projektdatei_Aktor
Kurzbeschreibung	Dieser Use Case behandelt das Laden eines bestehenden Projektes.
Priorität	Essentiell
Referenzen	2.2.4.9 Laden von Gerätedaten
Ablauf:	
Aktor-Aktionen	Systemantwort
1. LadeProjekt(Dateiname)	2. oeffne(Dateiname)
	3. leseDaten()
4. Projektdaten	5. schliesseDatei()
6. erfolgreich	7. ProjektErfolgreichGeladen()

Aktiver Aktor (Initiator) / Passiver Aktor

3.3.11 Use Case SpeichereProjekt

Name	Beschreibung
Aktor(en)	Arbeiter_Aktor, Projektdatei_Aktor
Kurzbeschreibung	Dieser Use Case behandelt das Speichern eines Projektes.
Priorität	Essentiell
Referenzen	2.2.4.8 Speichern von Gerätedaten
Ablauf:	
Aktor-Aktionen	Systemantwort
1. SpeichereProjekt(Dateiname)	2. oeffne(Dateiname)
	3. schreibeProjektdaten()
4. erfolgreich	5. schliesseDatei()
6. erfolgreich	7. ProjektErfolgreichGespeichert()

Aktiver Aktor (Initiator) / Passiver Aktor

3.3.12 Use Case SchliesseProjekt

Name	Beschreibung
Aktor(en)	Arbeiter_Aktor
Kurzbeschreibung	Dieser Use Case behandelt das Schliessen eines Projektes.
Priorität	Essentiell
Referenzen	-
Ablauf:	
Aktor-Aktionen	Systemantwort
1. schliesseProjekt()	2. [nochnichtgespeichert]ProjektSpeichern?
3. [nochnichtgespeichert] ProjektSpeichern(Auswahl)	4. ProjektErfolgreichGeschlossen

Aktiver Akteur (Initiator) / Passiver Akteur

3.3.13 Use Case zeigeKalibrierzertifikat

Name	Beschreibung
Aktor(en)	Arbeiter_Aktor
Kurzbeschreibung	Dieser Use Case behandelt das Anzeigen des Kalibrierzertifikates eines im Projekt befindlichen Gerätes.
Priorität	Optional
Referenzen	-
Ablauf:	
Aktor-Aktionen	Systemantwort
1. zeigeZertifikat(SerieNr)	2. ladeZertifikat
	3. returnKalibrierZertifikat

Aktiver Akteur (Initiator) / Passiver Akteur

3.3.14 Use Case aktualisiereKalibrierzertifikate

Name	Beschreibung
Aktor(en)	Arbeiter_Aktor
Kurzbeschreibung	Dieser Use Case behandelt das Aktualisieren der Kalibrierzertifikate.
Priorität	Optional
Referenzen	-
Ablauf:	
Aktor-Aktionen	Systemantwort
1. aktualisiereZertifikate ()	2. downloadeZertifikate() 3. aktualisierenErfolgreich

Aktiver Aktor (Initiator) / Passiver Aktor

3.4 System-Sequenzdiagramme

3.4.1 erstelleNeuesGeraet

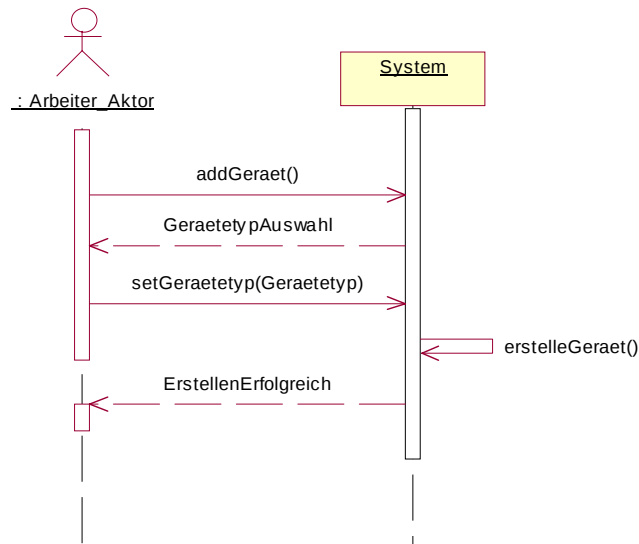


Abbildung 7

3.4.2 aendereGeraet

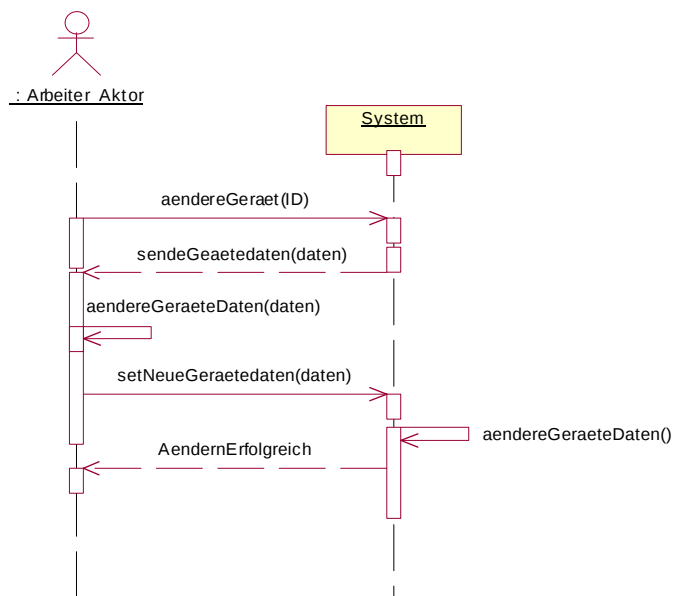


Abbildung 8

3.4.3 loescheGeraet

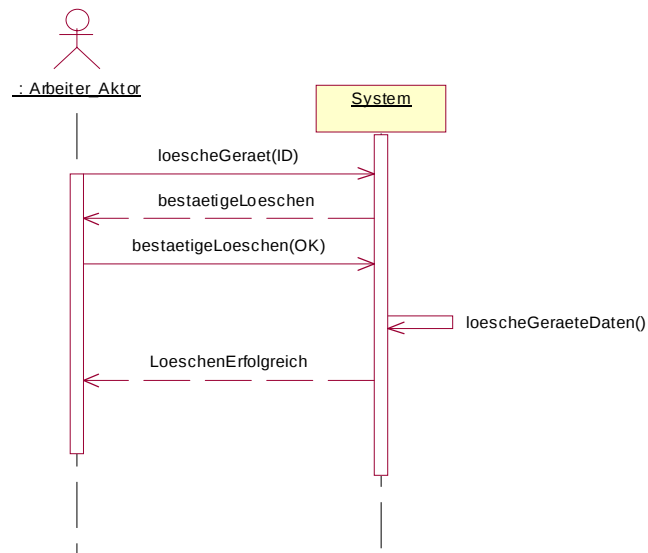


Abbildung 9

3.4.4 konfiguriereHW

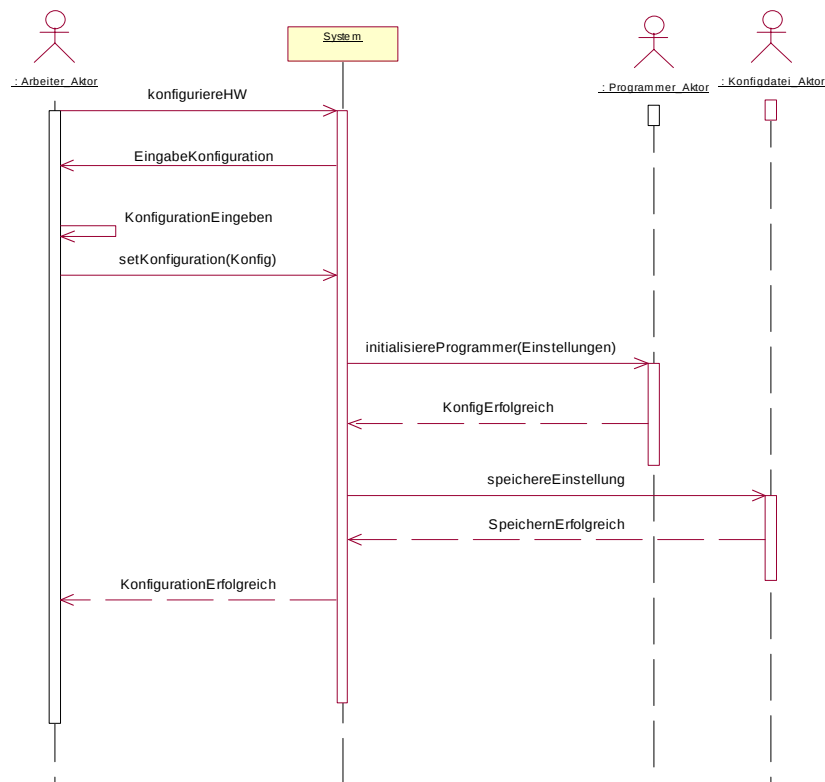


Abbildung 10

3.4.5 uploadeGeraetedaten

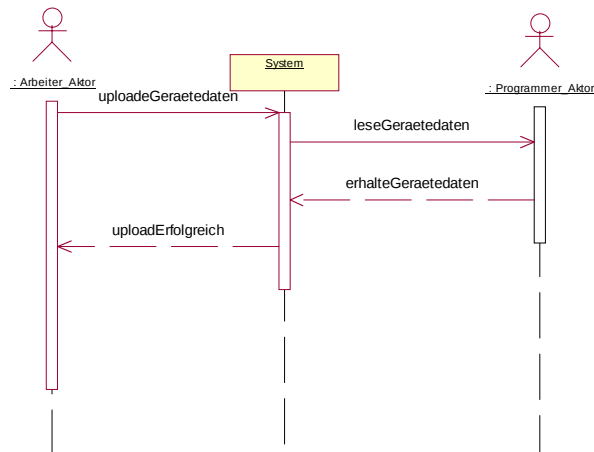


Abbildung 11

3.4.6 downloadeGeraetedaten

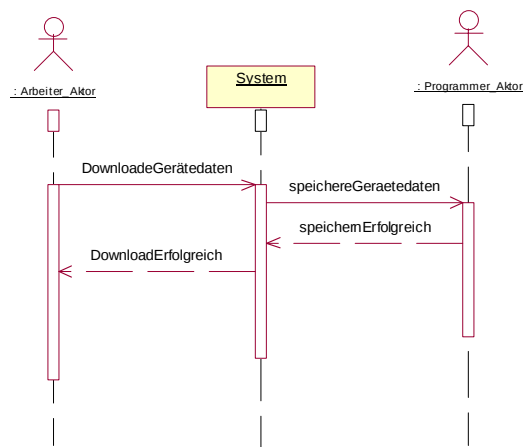


Abbildung 12

3.4.7 eroeffneProjekt

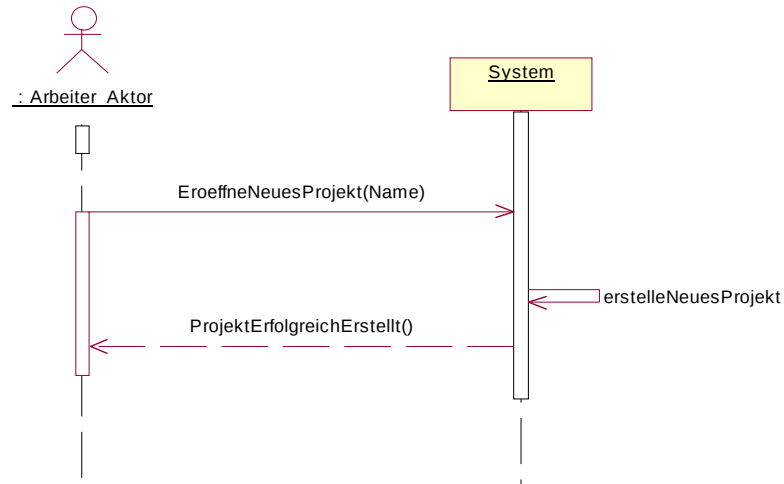


Abbildung 13

3.4.8 ladeProjekt

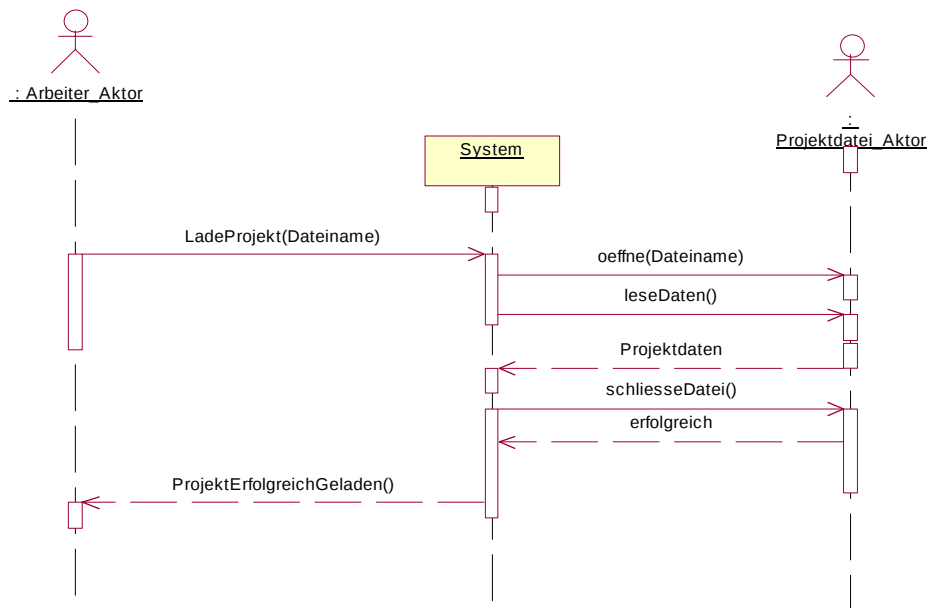


Abbildung 14

3.4.9 speichereProjekt

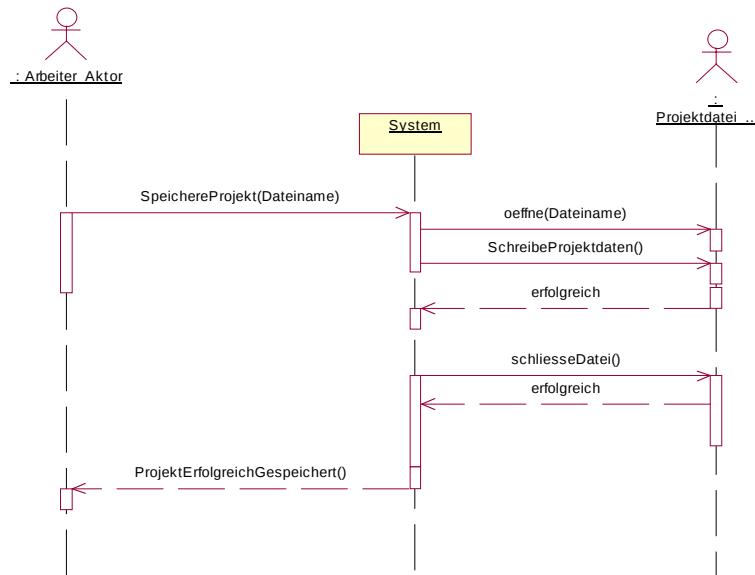


Abbildung 15

3.4.10 schliesseProjekt

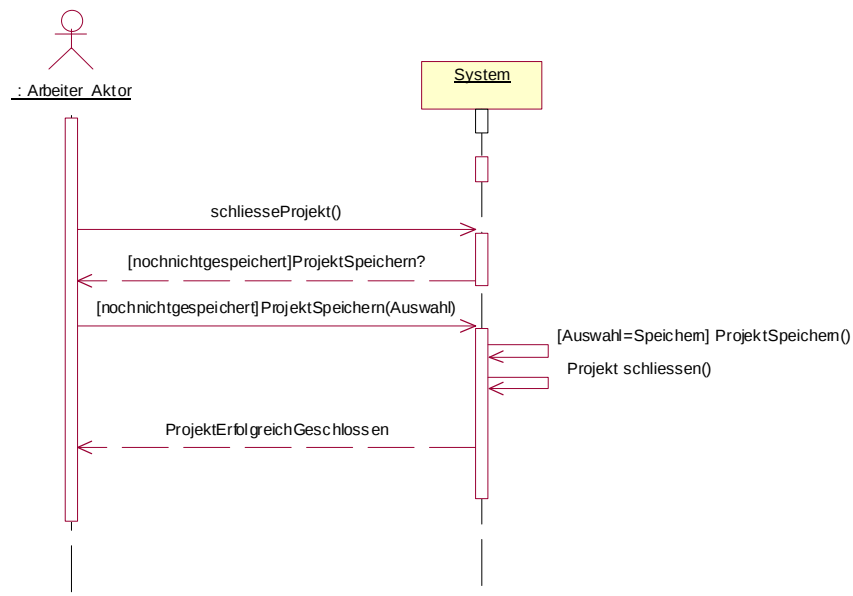


Abbildung 16

3.4.11 zeigeKalibrierzertifikat

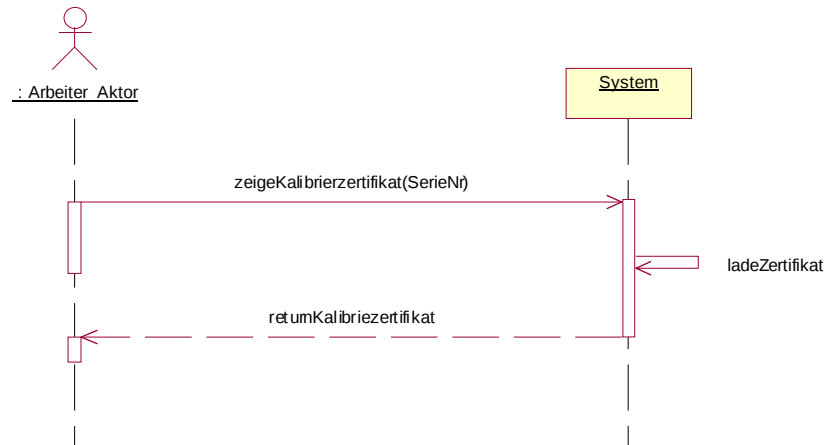


Abbildung 17

3.4.12 aktualisiereKalibrierzertifikat

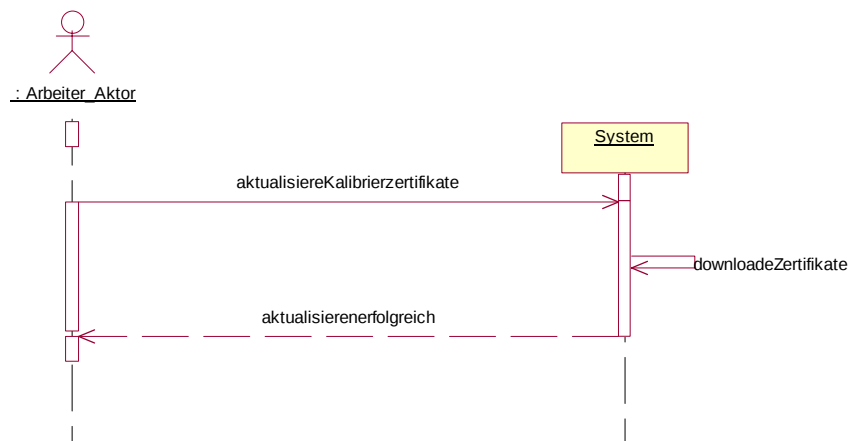


Abbildung 18

3.5 Konzeptionelles Modell

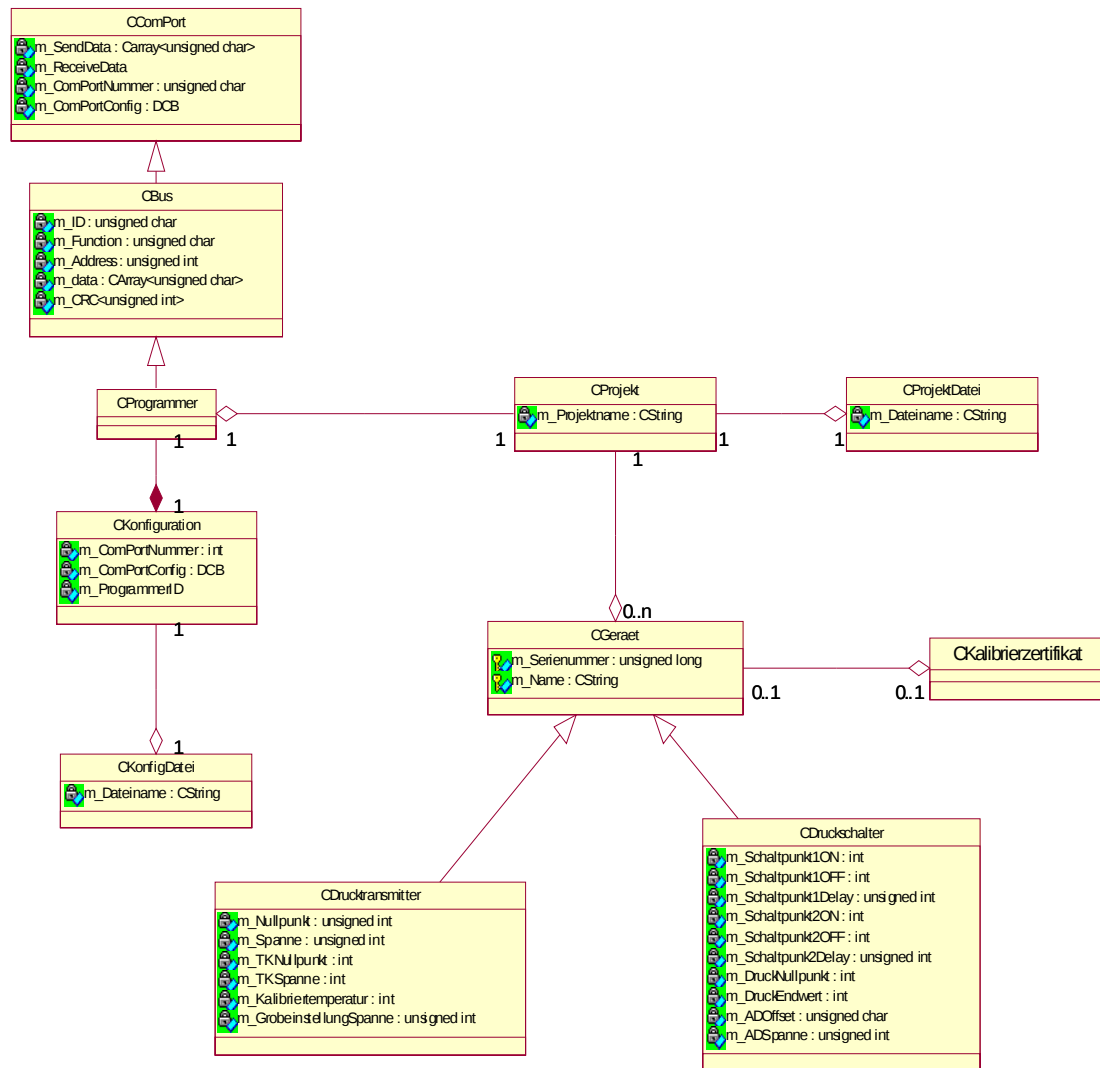


Abbildung 19

3.6 Kontrakte

3.6.1 erstelleNeuesGeraet

Name	erstelleNeuesGeraet
Verantwortlichkeiten	Erstellt neues Gerät und fügt es dem Projekt hinzu.
Cross Referenz	2.2.4.3 Erzeugen eines neuen Gerätes 3.3.5 Use Case ErstelleNeuesGeraet
Notizen	
Ausnahmen(Fehler)	Falls kein Name angegeben -> Fehleranzeige
Output	Ein neues Gerät.
Vorbedingungen	Ein Projekt ist erstellt.
Nachbedingungen	- Ein neues Gerät ist erstellt worden. - Das Gerät ist dem Projekt hinzugefügt worden. - Der Gerätetyp ist definiert worden.

3.6.2 aendereGeraet

Name	aendereGeraet
Verantwortlichkeiten	Ändert die Parameter eines bestehenden Gerätes.
Cross Referenz	2.2.4.6 Änderung von Gerätedaten 3.3.6 Use Case AendereGeraet
Ausnahmen(Fehler)	Falls die Geräteparameter ausserhalb des Wertebereiches -> Fehleranzeige
Output	-
Vorbedingungen	- Das zu editierende Gerät muss vorhanden sein. - Die Seriennummer des zu editierenden Gerätes darf nicht „Leer“ sein (d.h. das Gerät muss nach dem Erstellen zumindest einmal upgeloadet sein).
Nachbedingungen	- Die Parameter des Gerätes sind geändert worden. - Die Parameter stehen zum Download bereit.

3.6.3 loescheGeraet

Name	loescheGeraet
Verantwortlichkeiten	Entfernt das Gerät aus dem aktuellen Projekt.
Cross Referenz	2.2.4.4 Löschen von Geräten 3.3.7 Use Case LoescheGeraet
Ausnahmen(Fehler)	-
Output	-
Vorbedingungen	• Das zu löschende Gerät muss vorhanden sein.
Nachbedingungen	• Das Gerät ist aus dem aktuellen Projekt entfernt worden. • Die Gerätedaten des Gerätes sind gelöscht worden.

3.6.4 uploadeGeraetedaten

Name	uploadeGeraetedaten
Verantwortlichkeiten	Lädt die Gerätedaten aus dem, am Programmieradapter angeschlossenen Gerät.
Cross Referenz	2.2.4.5 Upload von Gerätedaten 3.3.3 Use Case UploadeGeraetedaten
Ausnahmen(Fehler)	• Falls der Programmieradapter oder die serielle Schnittstelle nicht angesprochen werden kann -> Fehleranzeige + Konfigurationsdialog. • Falls das ausgewählte Gerät nicht dem am Programmieradapter angeschlossenen Gerät entspricht -> Fehleranzeige.
Output	-
Vorbedingungen	• Das zu uploadende Gerät muss ausgewählt sein. • Die Seriennummer des ausgewählten Gerätes muss entweder „Leer“ oder mit dem angeschlossenen Gerät identisch sein.
Nachbedingungen	• Die Gerätedaten des ausgewählten Gerätes sind aktualisiert worden. • Die Parameter sind zum Ändern bereit.

3.6.5 downloadeGeraetedaten

Name	downloadeGeraetedaten
Verantwortlichkeiten	Lädt die Gerätedaten in das am Programmieradapter angeschlossenen Gerät.
Cross Referenz	2.2.4.7 Download von Gerätedaten 3.3.4 Use Case DownloadeGeraetedaten
Ausnahmen(Fehler)	- Falls der Programmieradapter oder die serielle Schnittstelle nicht angesprochen werden können -> Konfigurationsdialog. - Falls das ausgewählte Gerät nicht dem am Programmieradapter angeschlossenen Gerät entspricht -> Fehleranzeige.
Output	-
Vorbedingungen	- Das zu downloadende Gerät muss ausgewählt sein. - Die Seriennummer des ausgewählten Gerätes muss mit dem angeschlossenen Gerät identisch und nicht „Leer“ sein.
Nachbedingungen	- Die Gerätedaten des ausgewählten Gerätes sind aktualisiert worden. - Die Parameter sind zum Ändern bereit.

3.6.6 konfiguriereHW

Name	konfiguriereHW
Verantwortlichkeiten	Konfiguriert die serielle Schnittstelle und den angeschlossenen Programmieradapter.
Cross Referenz	2.2.4.1 Konfigurieren der Kommunikationsschnittstelle 3.3.8 Use Case KonfiguriereHW
Ausnahmen(Fehler)	Falls der Programmieradapter oder die serielle Schnittstelle nicht angesprochen werden können -> Fehleranzeige.
Output	Eintrag in Konfigdatei
Vorbedingungen	Keine
Nachbedingungen	- Die Konfiguration wurde geändert und in die Konfigdatei abgespeichert. - Die serielle Schnittstelle wurde mit den neuen Einstellungen geladen. - Der Programmieradapter wurde initialisiert.

3.6.7 eroeffneProjekt

Name	eroeffneProjekt
Verantwortlichkeiten	Erstellt ein neues und leeres Projekt
Cross Referenz	3.3.9 Use Case EroeffneProjekt
Ausnahmen(Fehler)	-
Output	• Neues Projekt
Vorbedingungen	• Bereits geöffnete Projekte müssen zuerst geschlossen werden.
Nachbedingungen	• Ein neues und leeres Projekt wurde erstellt. • Das leere Projekt wird angezeigt.

3.6.8 ladeProjekt

Name	ladeProjekt
Verantwortlichkeiten	Lädt ein bestehendes Projekt aus einer Projektdatei
Cross Referenz	2.2.4.9 Laden von Gerätedaten 3.3.10 Use Case LadeProjekt
Ausnahmen(Fehler)	• Falls die Projektdatei nicht gelesen werden kann -> Fehleranzeige.
Output	• Geladenes Projekt
Vorbedingungen	• Bereits geöffnete Projekte müssen zuerst geschlossen werden.
Nachbedingungen	• Das Projekt wurde aus der Projektdatei geladen. • Alle sich im Projekt befindlichen Geräte wurden aus der Projektdatei geladen. • Das Projekt und die Geräte werden angezeigt.

3.6.9 speichereProjekt

Name	speichereProjekt
Verantwortlichkeiten	Speichert ein bestehendes Projekt in eine Projektdatei.
Cross Referenz	2.2.4.8 Speichern von Gerätedaten 3.3.11 Use Case SpeichereProjekt
Ausnahmen(Fehler)	• Falls die Projektdatei nicht geschrieben werden kann -> Fehleranzeige.
Output	• Projektdatei
Vorbedingungen	• Das Projekt muss Geräte enthalten.
Nachbedingungen	• Das Projekt wurde in eine Projektdatei gespeichert. • Alle sich im Projekt befindlichen Geräte wurden in die Projektdatei gespeichert.

3.6.10 schliesseProjekt

Name	schliesseProjekt
Verantwortlichkeiten	Schliesst das geöffnet Projekt.
Cross Referenz	3.3.12 Use Case SchliesseProjekt
Ausnahmen(Fehler)	-
Output	-
Vorbedingungen	Es muss ein geöffnetes Projekt vorhanden sein.
Nachbedingungen	- Das Projekt wurde geschlossen. - Die Arbeitsfläche ist leer.

3.6.11 zeigeKalibrierzertifikat

Name	schliesseProjekt
Verantwortlichkeiten	Zeigt das Kalibrierzertifikat des ausgewählten Gerätes an.
Cross Referenz	2.2.2.3 Kann-Kriterien 3.3.13 Use Case zeigeKalibrierzertifikat
Notizen	optional
Ausnahmen(Fehler)	-
Output	-
Vorbedingungen	Es muss ein Kalibrierzertifikat für das gewählte Gerät vorhanden sein.
Nachbedingungen	Das Kalibrierzertifikat wird angezeigt.

3.6.12 aktualisiereKalibrierzertifikat

Name	schliesseProjekt
Verantwortlichkeiten	Aktualisiert und ergänzt die Kalibrierzertifikate.
Cross Referenz	2.2.2.3 Kann-Kriterien 3.3.14 Use Case aktualisiereKalibrierzertifikate
Notizen	optional
Ausnahmen(Fehler)	-
Output	-
Vorbedingungen	Keine
Nachbedingungen	Die Kalibrierzertifikate wurden ergänzt oder aktualisiert.

3.7 Review / Entscheidungen

3.7.1 Entscheidung

Ich habe mich Entschieden die Punkte 3.6.11 zeigeKalibrierzertifikat und 3.6.12 aktualisiereKalibrierzertifikat noch nicht zu realisieren, da diese Punkte den vorgegebenen Zeitrahmen sprengen würden.

Ein weiterer Punkt ist, dass die Firma Trafag die Daten nicht in der gewünschten Form vorliegen hat und ich deshalb dort auch noch etliche Zeit investieren müsste.

4 Design

4.1 Systemarchitektur

4.1.1 Zielsystem

Hardware:	Personal Computer
Betriebssystem:	Windows 2000/XP

Ich habe diese Konfiguration gewählt, weil sie den Vorgaben in den Anforderungsspezifikationen entsprechen.

4.1.2 Schnittstellen

Benutzeroberfläche:	Windows
Datenhaltung:	Datei, Registry
Andere Systeme:	Serielle Schnittstelle

Für die Datenhaltung habe ich mich für Datei und Registry entschieden. Die Datei verwende ich für die Projekt- und Gerätedaten und die Speicherung realisiere ich mit der Serialisierung der MFC. Die Registry verwende ich für die Speicherung der Konfigurationsdaten. Die Registry eignet sich hier besonders, da sich bei ihr die Parameter einzeln und indexiert ansprechen lassen. Ein weiterer Vorteil gegenüber einer Datei ist der einheitliche Zugriff und die benutzerbezogene Konfiguration.

4.1.3 Entwicklung

Entwicklungsumgebung:	Visual Studio .NET 2003
Programmiersprache:	VC++
Klassenbibliotheken:	STL, MFC

Ich verwende die Entwicklungsumgebung von Microsoft Visual Studio .NET 2003 da sie die aktuellste Entwicklungsumgebung von Microsoft ist und ich auf ihr, besonders mit VC++ und MFC, relativ viel Erfahrung habe.

Als Klassenbibliotheken verwende ich hauptsächlich die MFC. Für besondere Funktionen wie z.B. Mathematikfunktionen verwende ich die STL.

4.2 User Interface

4.2.1 Hauptfenster

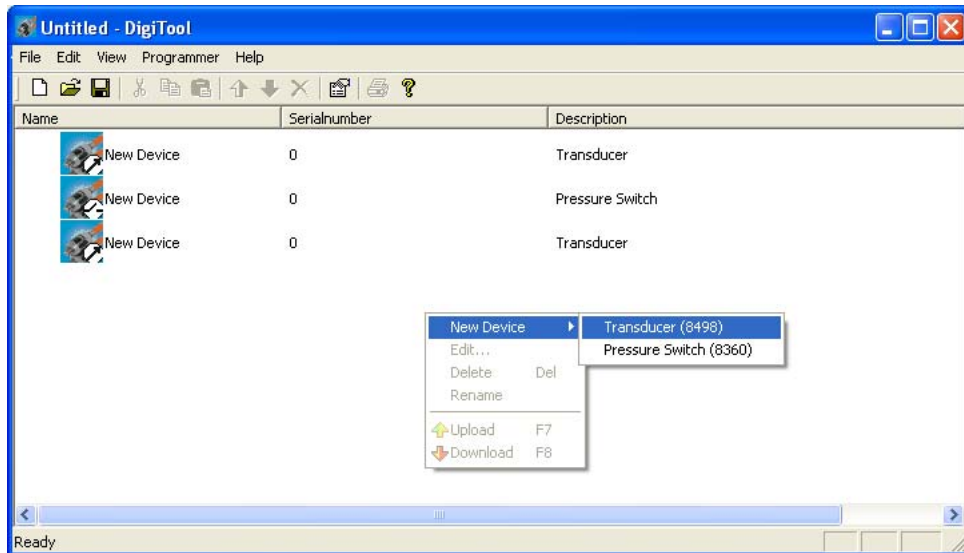


Abbildung 20

4.2.2 Menu

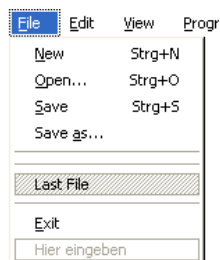


Abbildung 21



Abbildung 22

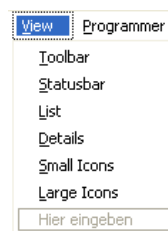


Abbildung 23



Abbildung 24



Abbildung 25

4.2.3 Konfigurationsdialog

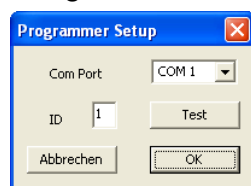


Abbildung 26 Konfigurations Dialog

4.2.4 Gerätedialoge

The 'Pressuretransducer / Device' dialog box contains the following fields and controls:

- Serialnumber: 0
- Calibrationdate: Week 0 / 2000
- Calibrationtemperature: 0 °C
- TC zero point correction: 0 ppm
- TC span correction: 0 ppm
- Span zoom: 0 %
- Settings section with two sliders:
 - Zeroset: A slider from 0 to 100 with a value of 50.10 %.
 - Spanset: A slider from 0 to 100 with a value of 50.10 %.
- Description: Transducer
- Buttons: Cancel, Upload, Download, OK

Abbildung 27 Drucktransmitter Dialog

The 'Pressureswitch / Device' dialog box contains the following fields and controls:

- Serialnumber: 0
- Calibrationdate: Week 0 / 2000
- Pressure Range: 0 ... 0 bar
- Switchpoint 1 section:
 - ON: 0 bar
 - OFF: 0 bar
 - Delay: 0 ms
- Switchpoint 2 section:
 - ON: 0 bar
 - OFF: 0 bar
 - Delay: 0 ms
- Description: Pressure Switch
- Buttons: Abbrechen, Upload, Download, OK

Abbildung 28 Druckschalter Dialog

4.3 Interaktionsdiagramme

4.3.1 erstelleDruckschalter

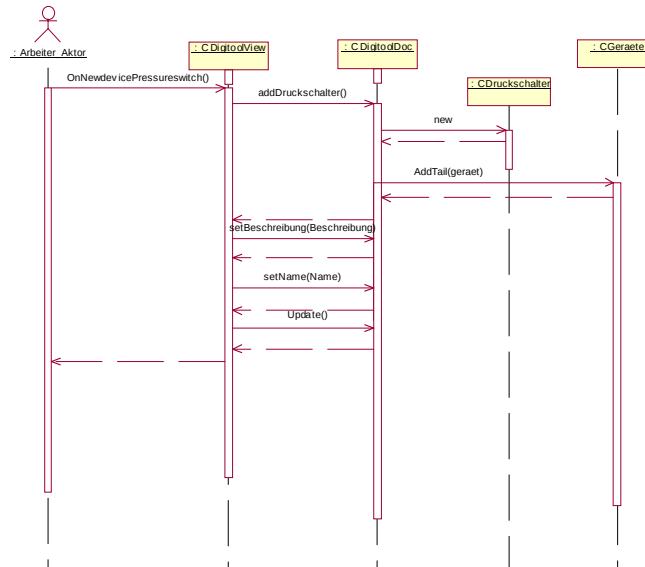


Abbildung 29

4.3.2 erstelleDrucktransmitter

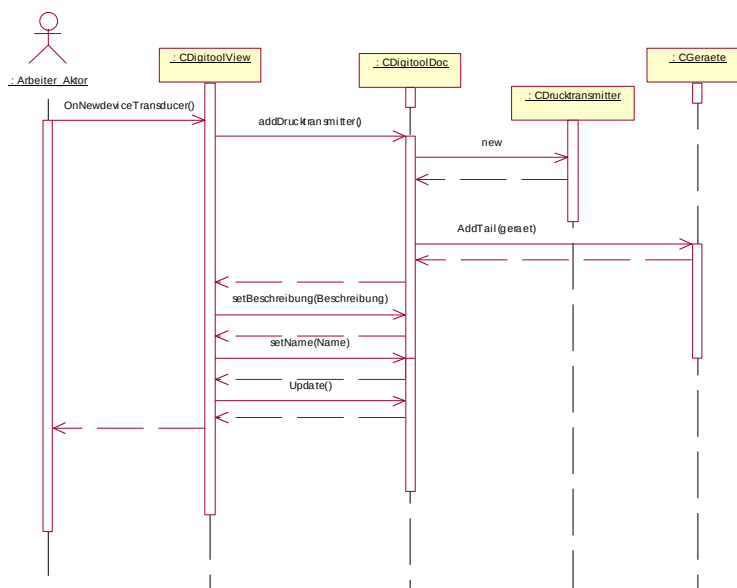


Abbildung 30

4.3.3 aendereGeraet

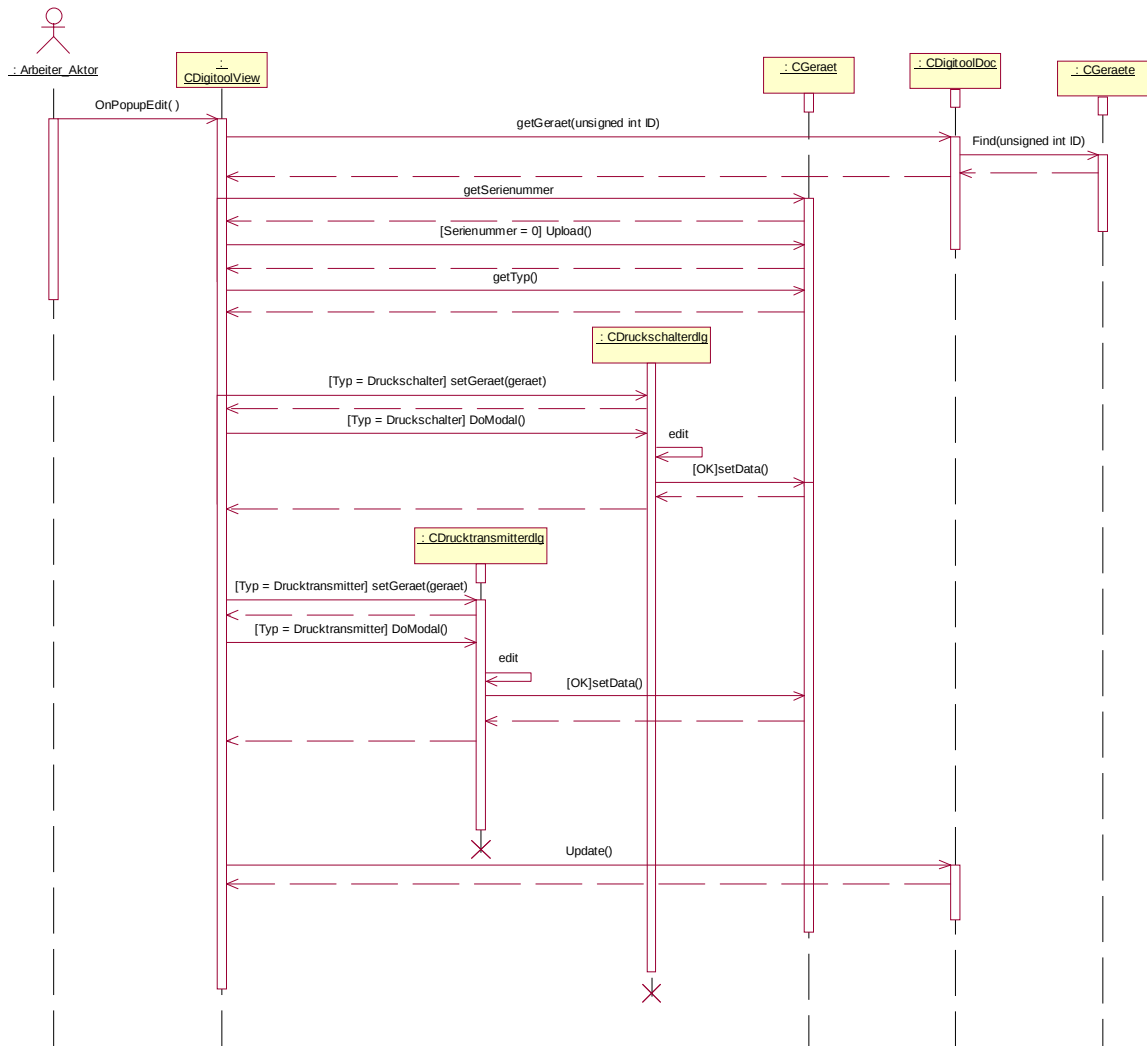


Abbildung 31

4.3.4 downloadeGeraetedaten

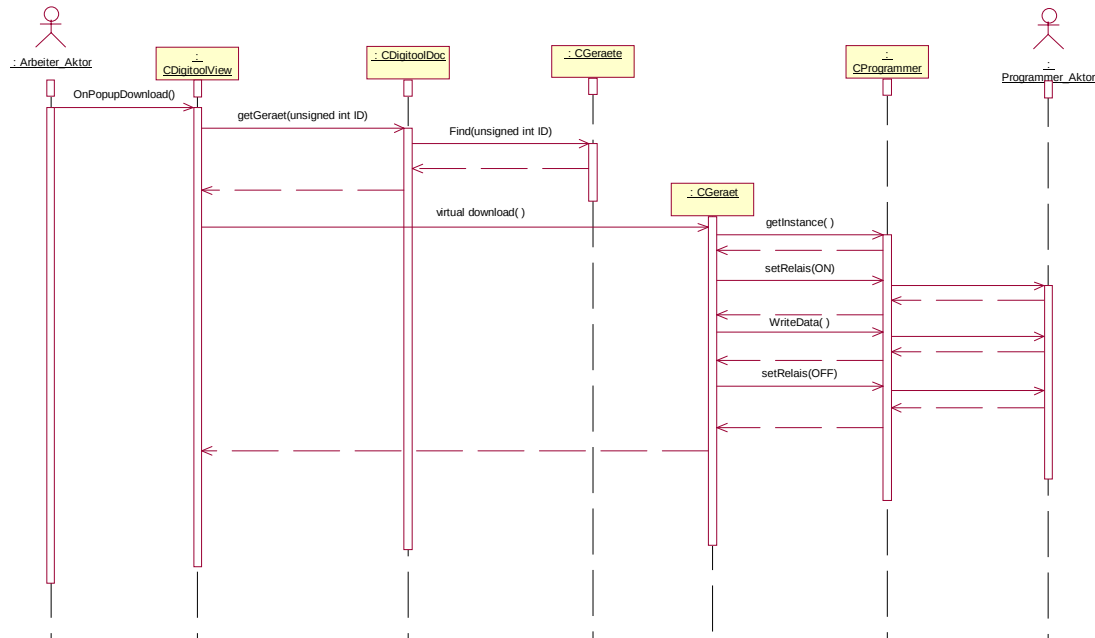


Abbildung 32

4.3.5 uploadeGeraetedaten

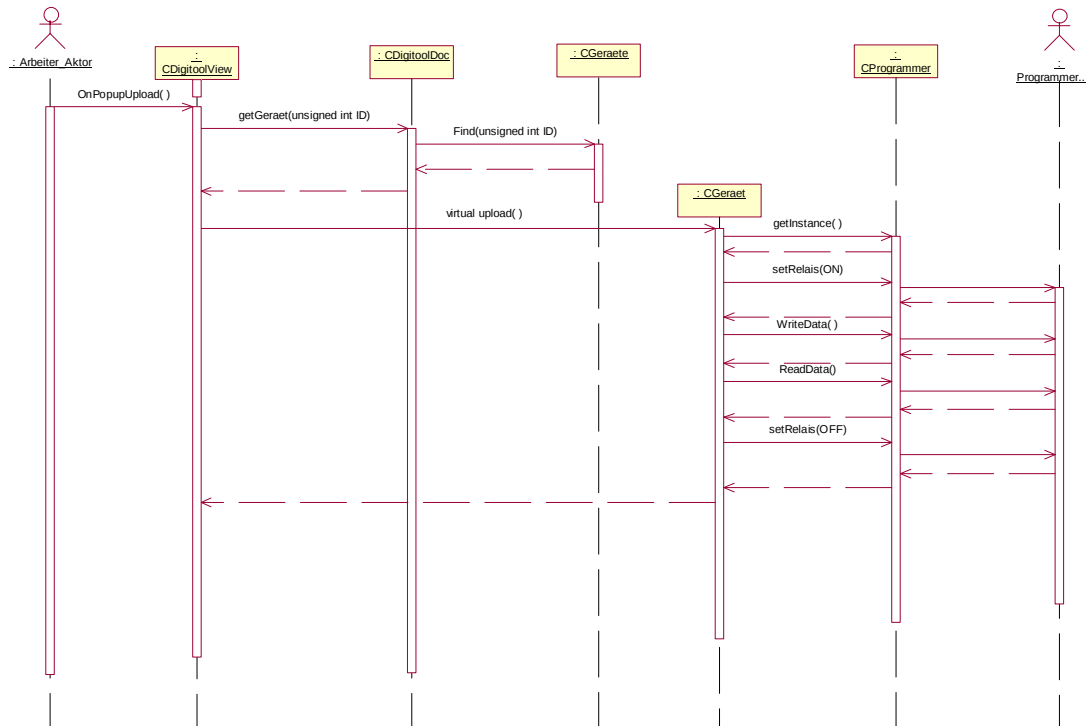


Abbildung 33

4.3.6 loescheGeraet

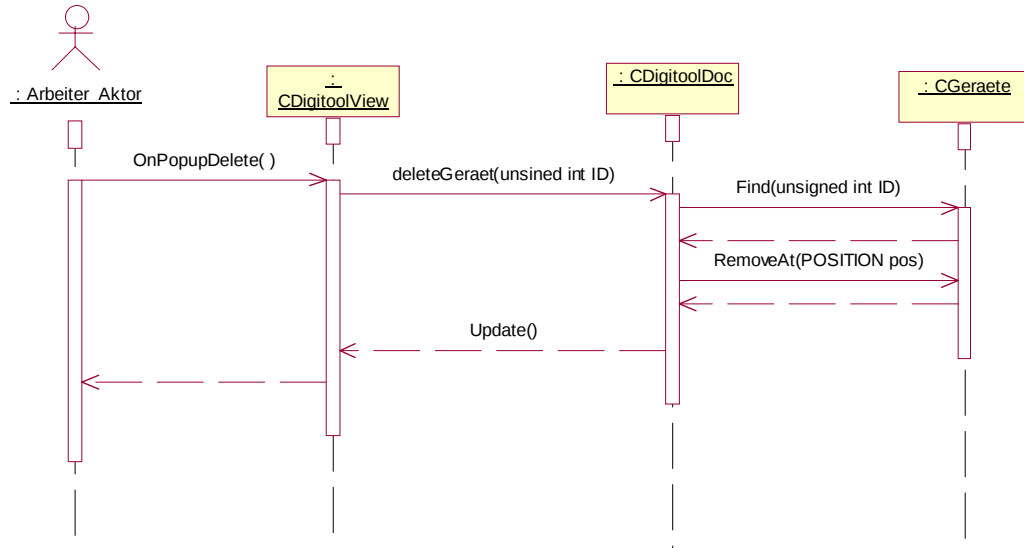


Abbildung 34

4.3.7 eroeffneProjekt

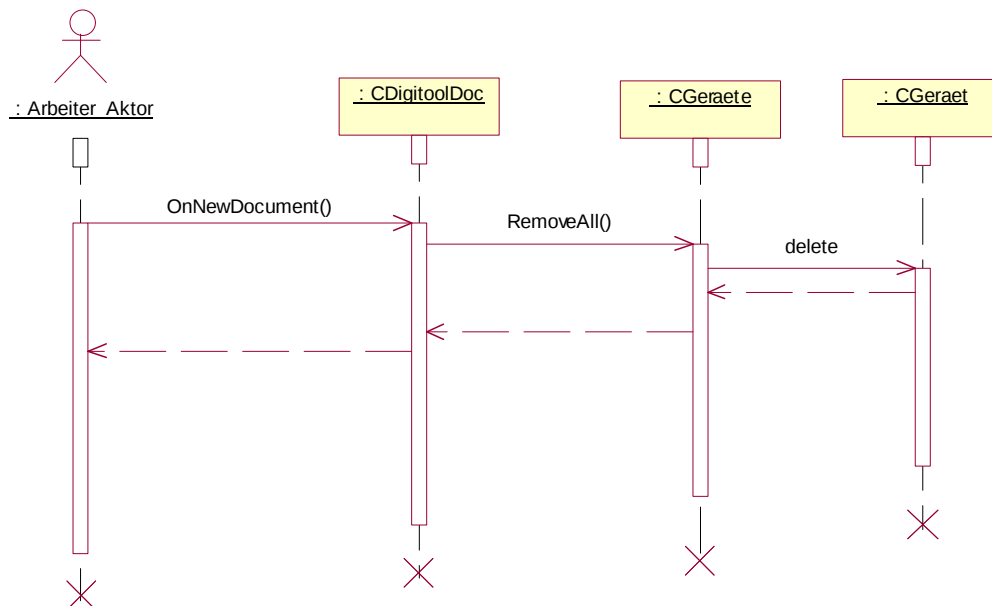


Abbildung 35

4.3.8 oeffneProjekt

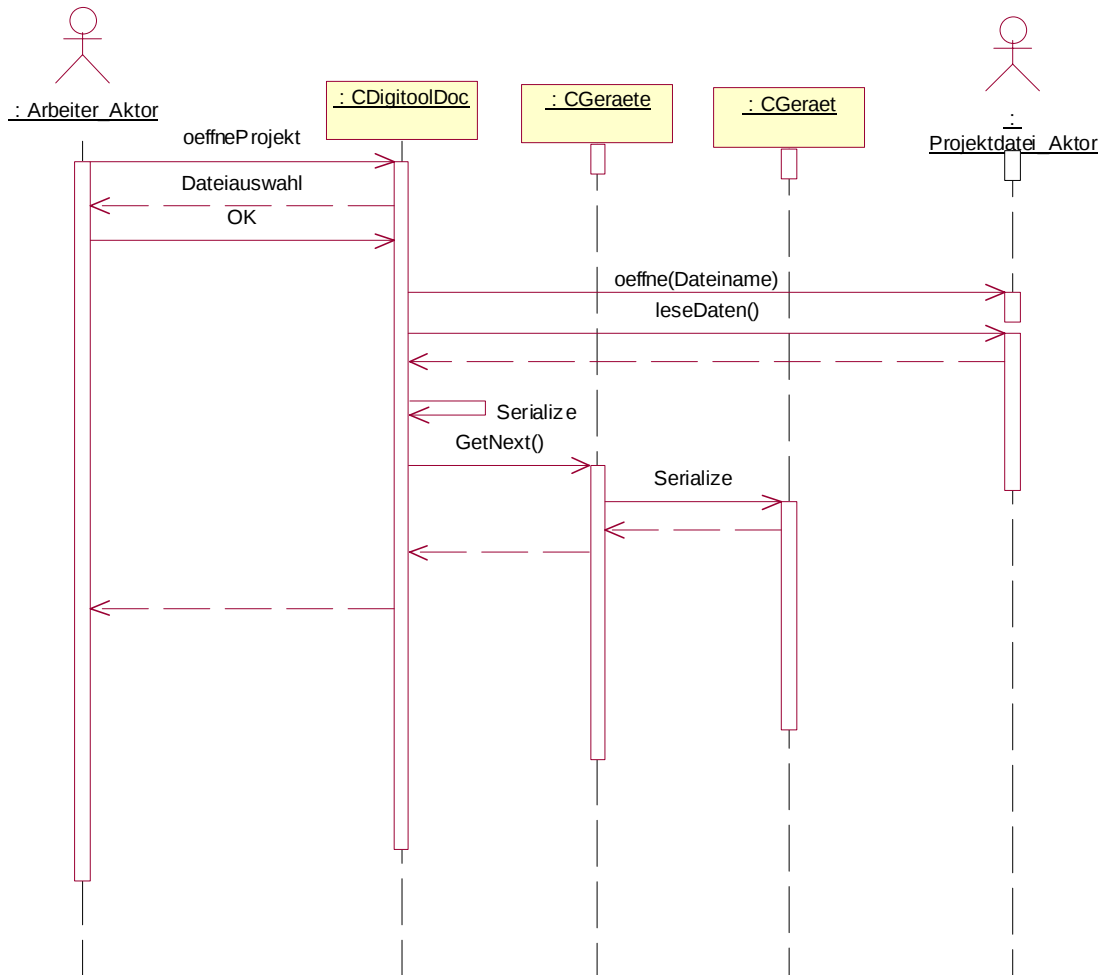


Abbildung 36

4.3.9 speichereProjekt

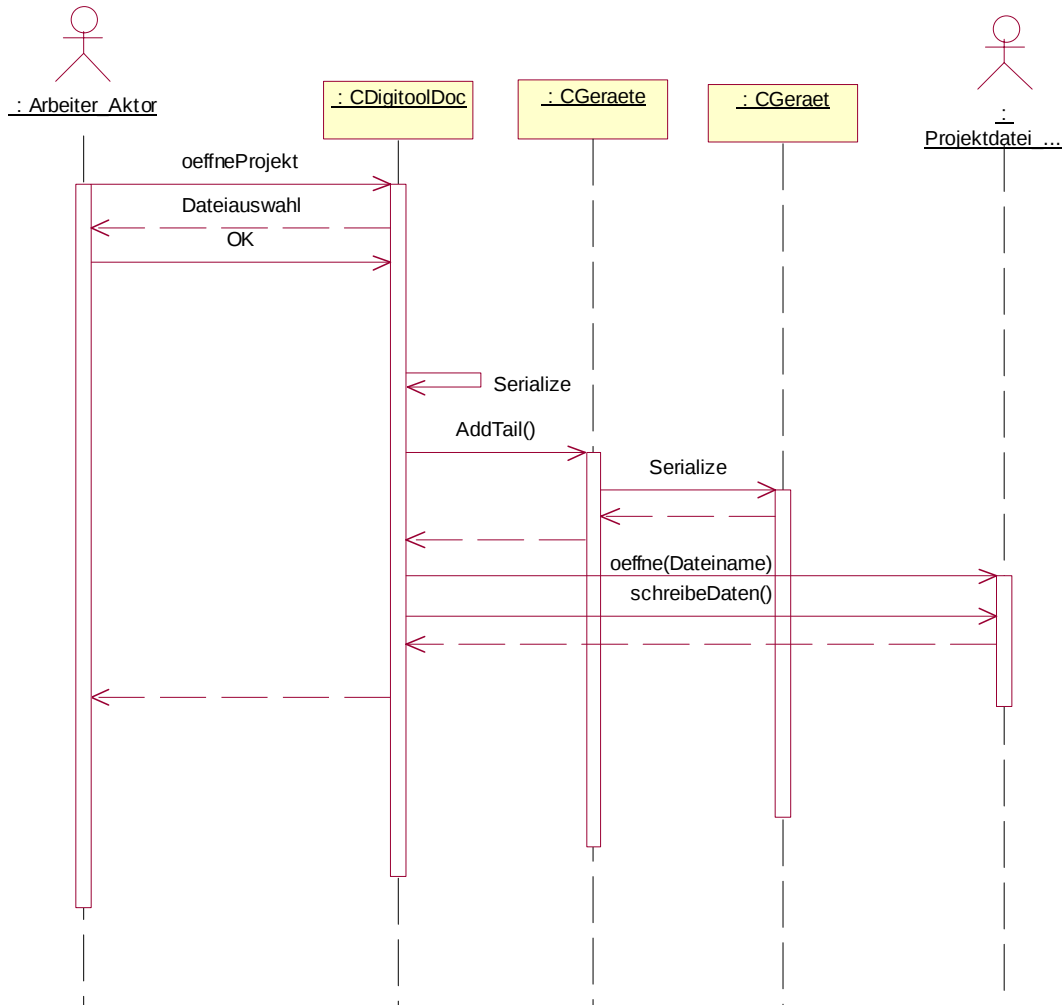


Abbildung 37

4.3.10 konfiguriereHW

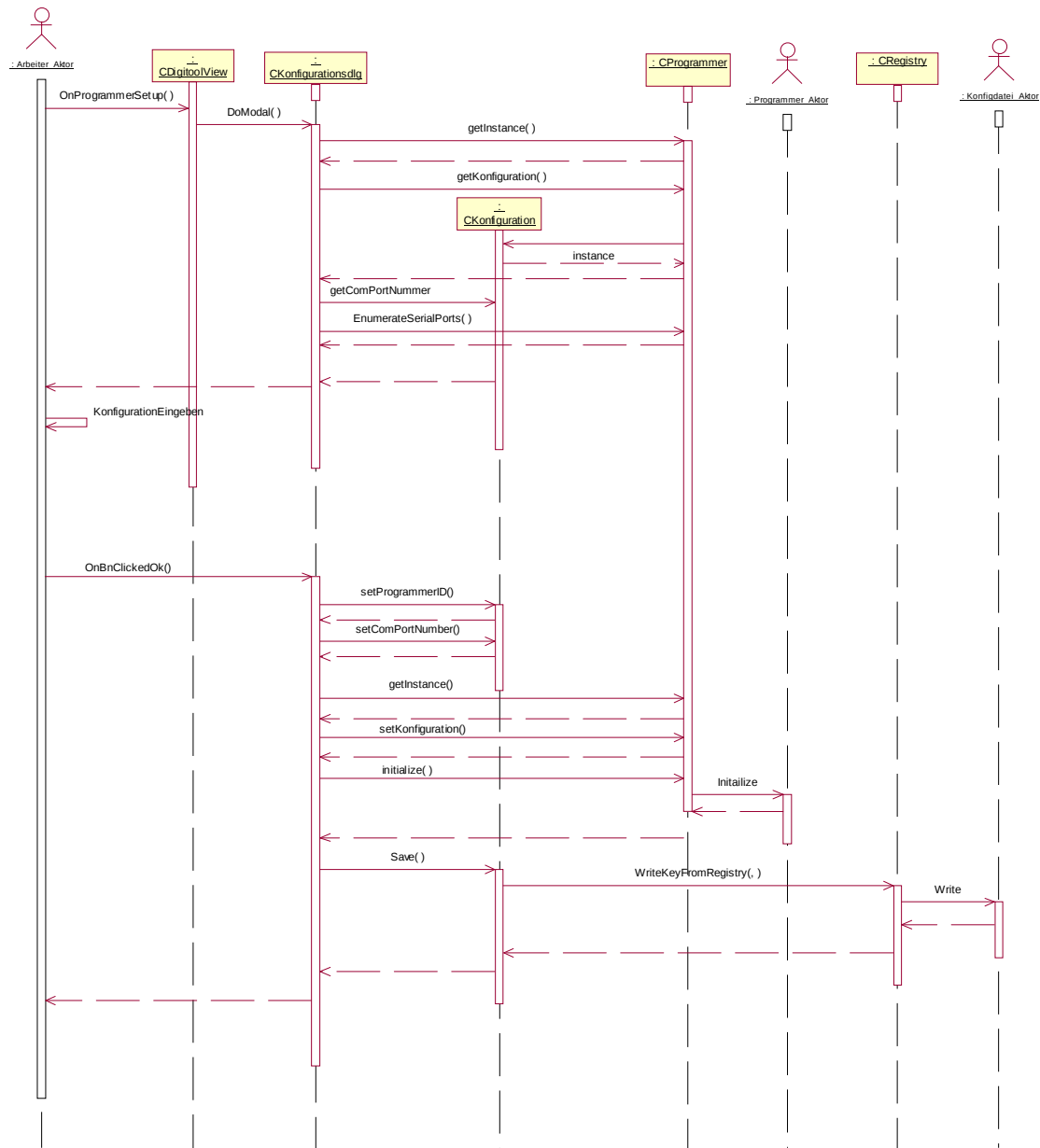


Abbildung 38

4.3.11 initialisiereHW

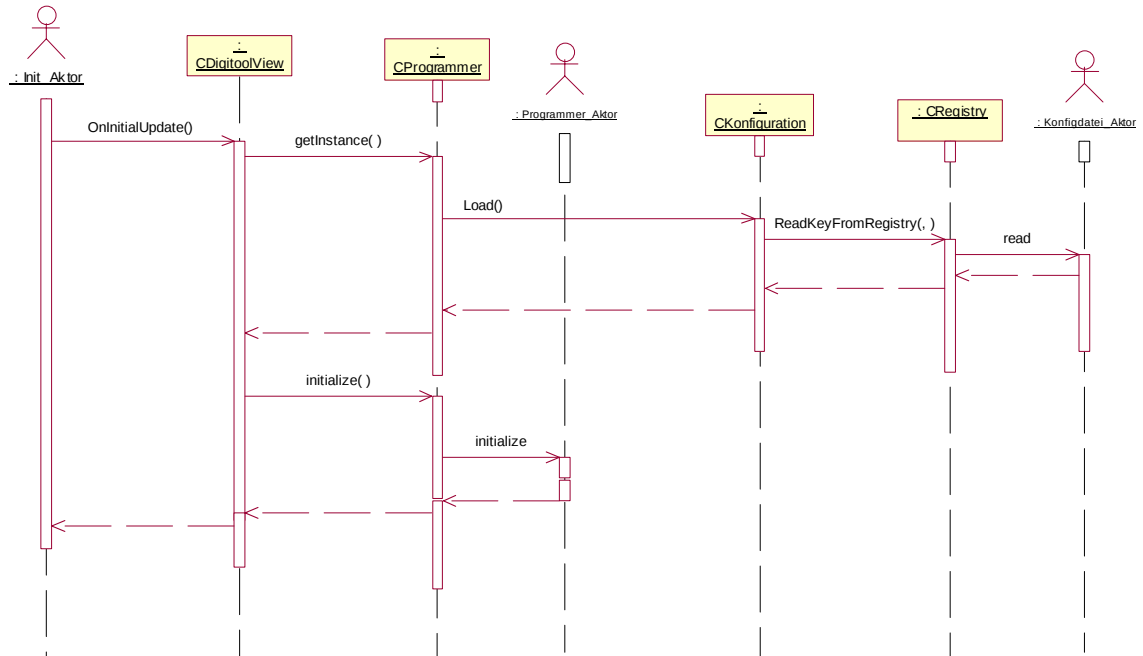


Abbildung 39

4.3.12 Package Diagramm

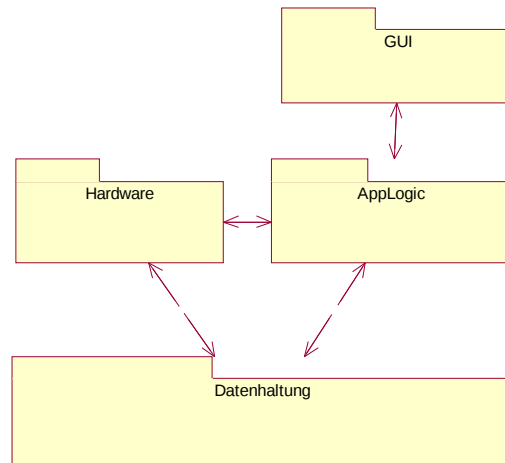


Abbildung 40

4.4 Design Klassendiagramm

4.4.1 GUI

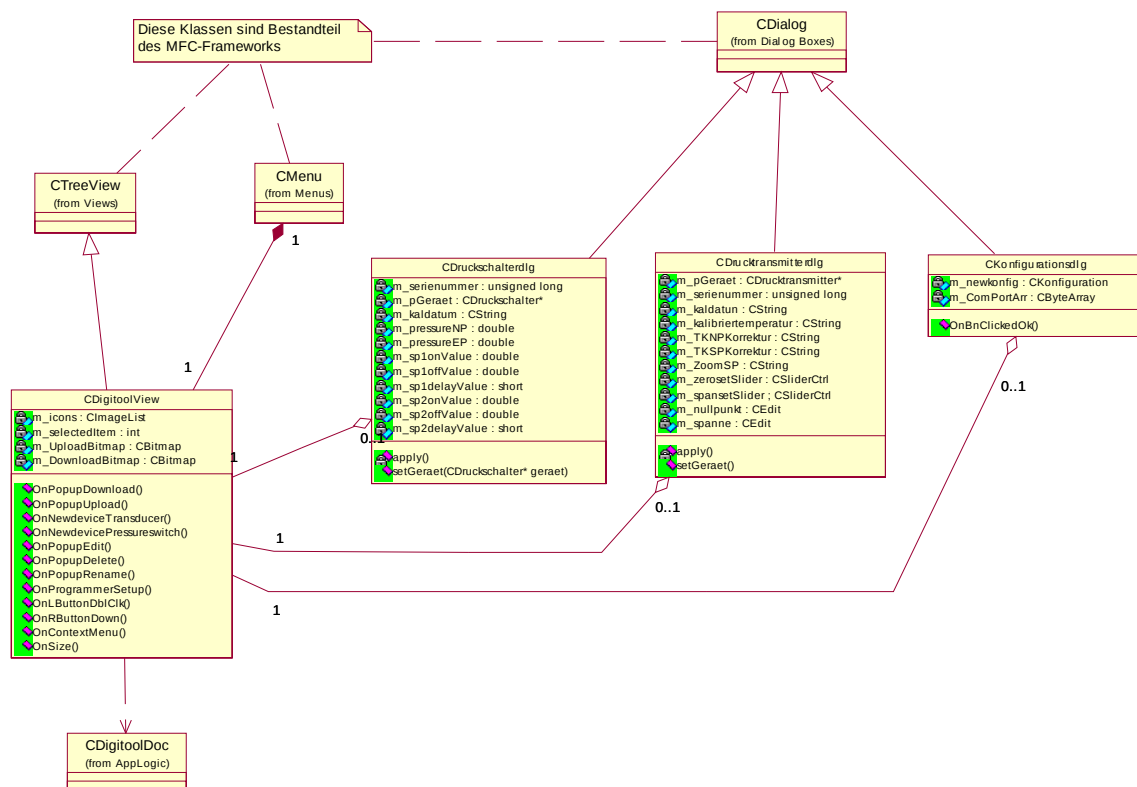


Abbildung 41

4.4.3 Hardware

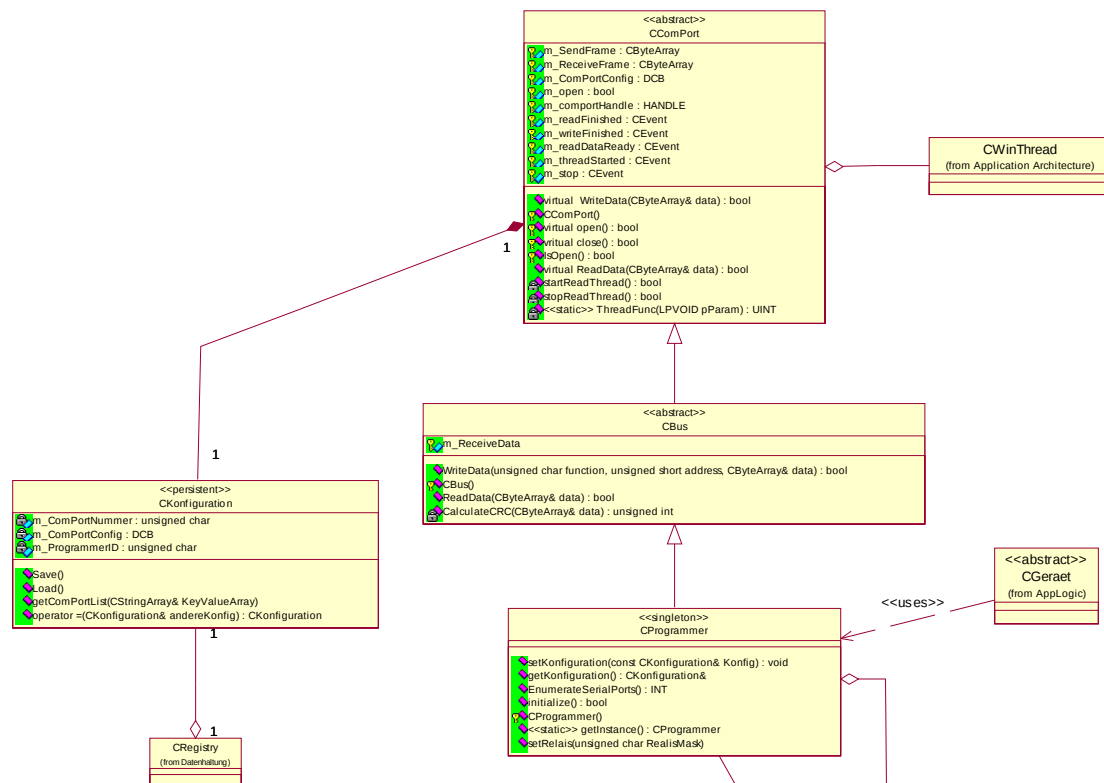


Abbildung 43

4.4.4 Datenhaltung

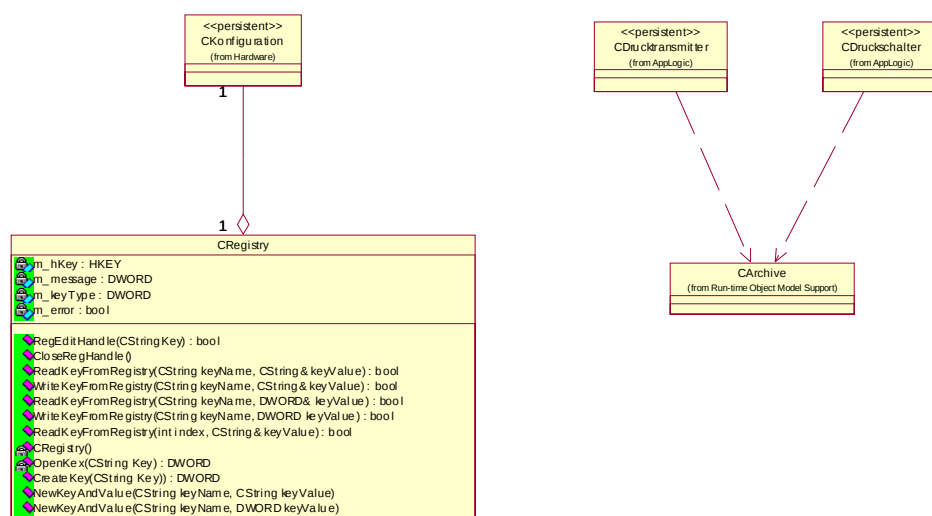


Abbildung 44

5 Implementation

5.1 Serielle Schnittstelle

Die serielle Schnittstelle ist eine der grössten Herausforderungen dieser Arbeit. Da bei der Verwendung der seriellen Schnittstelle etliche Komplikationen beim Buffer-Handling auftreten können, verdient dies besondere Beachtung.

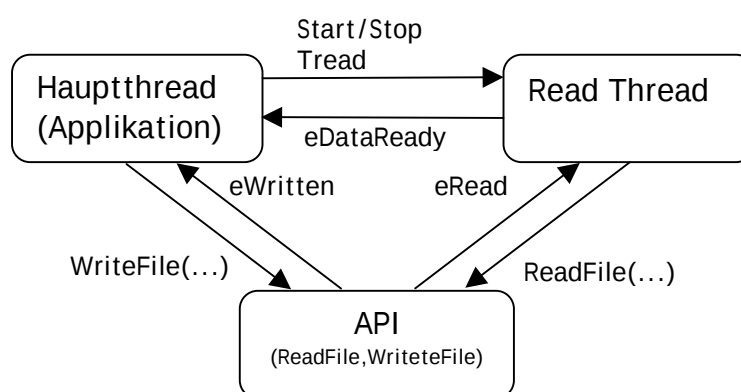
5.1.1 Threads

Um sicherzustellen das beim Lesen von der seriellen Schnittstelle auch Daten vorhanden sind, habe ich diese im Overlapped Mode geöffnet und hohle die Daten mittels eines Read-Threads ab.

Die Entscheidung ein Read-Thread zu realisieren stützt sich auf die bessere Prozessorbenutzung, welche bei einem wartenden Thread im Gegensatz zum „Pollingbetrieb“, der Prozessor für andere Applikationen oder Threads zu Verfügung steht.

5.1.2 Thread-Synchronisierung

Die Thread-Synchronisierung wurde über Events realisiert, da die API-Funktionen (WriteFile, ReadFile) in der Overlapped-Struktur auch solche verwenden.



6 Test

6.1 Testmethode

Für dieses Projekt zeigt sich die Blackbox Testmethode als die am besten geeignete Methode ab.

6.2 Erreichung der Ziele

Kategorie	Kriterien	✓/✗	Bemerkung
2.2.2.1 Allgemeine Beschreibung	Das zu entwickelnde Softwareprodukt ermöglicht seinen Benutzern Druckmessgeräte der Firma Trafag zu konfigurieren oder nachzukalibrieren .	✓	
	Das Programm kann mit Hilfe eines Programmieradapters („Digipack“) die Konfiguration der Druckmessgeräte lesen oder beschreiben.	✓	
2.2.2.2 Muss-Kriterien	Erstellung einer grafischen Benutzeroberfläche	✓	
	Ansteuerung des Programmieradapters in einem Modul	✓	
	Implementierung verschiedener Geräteprofile (Druckschalter)	✓	
	Implementierung verschiedener Geräteprofile (Analoge Drucktransmitter)	✓	
	Auslesen von Gerätedaten wie Seriennummer, Produktionsdatum usw.	✓	
	Speichern/Laden der Geräteparameter in eine /von einer Datei	✓	
2.2.2.3 Kann-Kriterien	Anzeigen von digitalen Kalibrierzertifikaten (Mittels Seriennummer)	✗	Ist im aktuellen Entwicklungsstand nicht implementiert. Gemäss Entscheid vom 17.06.2004.
	Herunterladen neuer, aktueller, digitaler Kalibrierzertifikate über das Web (eine Art Live Update)	✗	Ist im aktuellen Entwicklungsstand nicht implementiert. Gemäss Entscheid vom 17.06.2004.

6.3 Einsatz

Kategorie	Kriterien	✓/✗	Bemerkung
2.2.3.3 Software	Kommt ohne zusätzliche SW aus	✓	
	GUI Elemente in Englisch	✓	
2.2.3.4 Hardware	Läuft auf Windows 2000/XP	✓	
	Mindestanforderung : Desktop-PC oder Notebook mit Intelarchitektur(300MHz)	✓	

6.4 Funktionen

Kategorie	Kriterien	✓/✗	Bemerkung
2.2.4.1 Konfigurieren der Kommunikations- schnittstelle	Die Kommunikationsschnittstelle (Com-Port) muss ausgewählt werden können.	✓	
	Diese Einstellungen müssen abgespeichert werden.	✓	In Registry
2.2.4.2 Initialisieren des Programmier- adapters („Digipack“)	Nach der Auswahl des Com-Ports muss getestet werden, ob der Programmieradapter am aus- gewählten Port angeschlossen ist.	✓	
2.2.4.3 Erzeugen eines neuen Gerätes	Ein Gerät der Arbeitsfläche hinzugefügt.	✓	
	Der Benutzer muss beim Erzeugen eines neuen Gerätes den Gerätetyp auswählen können.	✓	
	Mit dieser Auswahl wird die richtige Programmiermethode und das GUI mit dem Parametern ausgewählt.	✓	
2.2.4.4 Löschen von Geräten	Die erzeugten Geräte müssen wieder von der Arbeitsfläche entfernt werden können.	✓	
2.2.4.5 Upload von Gerätedaten	Es werden über den Programmier- adapter die Geräteparameter aus dem Gerät gelesen.	✓	

2.2.4.7 Download von Gerätedaten	Es werden die Geräteparameter über den Programmieradapter in das Gerät geladen.	☒	
2.2.4.8 Speichern von Gerätedaten	Die Gerätedaten müssen in eine Datei gespeichert werden können.	☒	In Projektdatei
2.2.4.9 Laden von Gerätedaten	Die abgespeicherten Gerätedaten müssen wieder aus einer Datei geladen werden können.	☒	Aus Projektdatei

6.5 Schnittstellen

Kategorie	Kriterien	☒ / ☐	Bemerkung
2.2.5.1 Benutzeroberfläche	Es muss eine vollgrafische Windowsapplikation erstellt werden.	☒	
	Das User Interface muss den gängigen Standards für Windows- Applikationen genügen.	☒	
2.2.5.2 Software	Keine Schnittstelle zu anderen Softwareprodukten	☒	
2.2.5.3 Hardware	Programmieradapter verwendet	☒	
	Seriellen Schnittstelle (RS232) angesteuert	☒	
	Ansteuerung eventuell auch über Bluetooth oder USB (über Windows ComPort -Treiber).	☒	Zusätzlich mit USB getestet.
2.2.5.4 Kommunikations- schnittstellen	Der Programmieradapter wird mit einem Modbus Protokoll gemäss Spezifikationen der Trafag AG angesteuert.	☒	

6.6 Daten/Datenbank

Kategorie	Kriterien	✓/✗	Bemerkung
	Gerätedaten oder Gerätekonfigurationen müssen in eine Datei abgespeichert werden können.	✓	In Projektdatei

6.7 Weitere Merkmale

Kategorie	Kriterien	✓/✗	Bemerkung
2.2.8.2 Randbedingungen im Betrieb	Es ist eine MFC ¹ (Microsoft Foundation Classes) Anwendung in C++ zu entwickeln.	✓	
	Der Programmieradapter muss für den Betrieb immer angeschlossen sein.	✓	

¹ MFC = Microsoft Foundation Classes ein objektorientiertes Framework auf C++ Basis der Windows API.

7 Schlusswort

Nun ist es soweit, das Ende der Diplomarbeit ist erreicht.

Die Aufgabe war für mich eine Herausforderung, da es einige schwierige Punkte zu bewältigen gab. Die Ansteuerung der seriellen Schnittstelle und die Grafische Benutzeroberfläche erforderten mehr Beachtung als ich eingerechnet hatte.

Ebenfalls hatte ich unterschätzt wie viel Zeit bei der Implementation mit lesen von Onlinehilfen und durchforsten der MSDN¹ benötigt, ich konnte dieses Zeitdefizit jedoch gut mit den nicht benötigten Reserven von Analyse und Design kompensieren.

Ich konnte bei dieser Arbeit die in den letzten Semestern erhaltenen Theorie anwenden und in die Praxis umsetzen. Ebenfalls konnte ich von den gemachten Erfahrungen profitieren und habe mir weiteres Wissen über die Entwicklungsumgebung und die Windows-API aneignen können.

Trotz einigen Hindernissen, bin ich der Meinung mit meiner Arbeit ein gutes Projekt erarbeitet zu haben, dass sich in der Industrie verwenden lässt.

Eine erste Version ist bei der Firma Trafag bereits im Test- und Piloteinsatz, es sind auch schon weitere Wünsche für Weiterentwicklungen vorhanden.

Ich werde diese Software bei der Firma Trafag weiterentwickeln und betreuen.

Ich möchte an dieser Stelle meinem Betreuer Dominik Venosta und meinen Fachlehrern im 7.Semester für die tolle Unterstützung danken.

¹ MSDN = Microsoft Developer Network (Online Hilfe der Microsoft für Entwickler im Web)

8 Literatur- und Linkverzeichnis

8.1 Softwareengineering

Autor : Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides
Buchtitel : Entwurfsmuster
Verlag : Addison-Wesley
ISBN-Nr. : 3-8273-1862-9

Autor : Heide Balzert
Buchtitel : Lehrbuch der Objektmodellierung
Verlag : Spektrum Akademischer Verlag Heidelberg-Berlin
ISBN-Nr. : 3-8274-0285-9

Autor : Fredy Ulmer
Titel : Software Engineering (TSU Kursstoff)
Fach : Software Engineering

8.2 Programmierung C++/MFC

Autor : Davis Chapman
Buchtitel : Visual C++ .NET
Verlag : Markt+Technik
ISBN-Nr. : 3-8272-6320-4

Autor : David Kruglinski, George Sheperd, Scot Wingo
Buchtitel : Inside Visual C++
Verlag : Microsoft Press Deutschland
ISBN-Nr. : 3-86063-461-5

Titel : Windows Programmierung mit Visual C++ (TSU Kursstoff)
Fach : Windows Programmieren

Autor : Markus Burri
Titel : C++ (TSU Kursstoff)
Fach : C++

8.3 Internetlinks

MSDN
Microsoft Developer Network <http://msdn.microsoft.com/visualc/>

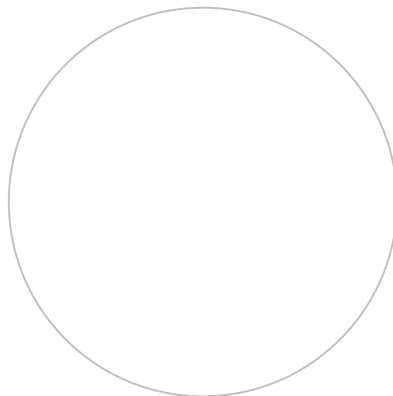
CodeGuru Forum <http://www.codeguru.com/forum/>

9 Anhang A

9.1 CD

Inhalt:

- Source
- Setuppaket
- Dokumentation Diplomarbeit
- Dokumentation Programmieradapter „Digipack“



9.2 Arbeitspakete

9.2.1 Planung

Definition Anforderungsspezifikationen

Ziel: Anforderungsspezifikationen unter Berücksichtigung der Aufgabenstellung und Wünsche der Firma Trafag erstellen.

Beginn geplant: 06.04.2004 Dauer geplant: 10.04.2004
Beginn effektiv: 05.04.2004 Dauer Effektiv: 10.04.2004
Begründung der Abweichungen: -

Bestehende Dokumentation Studieren

Ziel: Bestehende Dokumentationen von der Firma Trafag über Programmieradapter und Geräte studieren. Klären unklarer Punkte.

Beginn geplant: 06.04.2004 Dauer geplant: 10.04.2004
Beginn effektiv: 05.04.2004 Dauer Effektiv: 10.04.2004
Begründung der Abweichungen: -

Erstellung erster Entwürfe

Ziel: Erstellung erster Skizzen (Wichtigste Highlevel Use Cases) des Systems. Definition der Systemgrenzen.

Beginn geplant: 11.04.2004 Dauer geplant: 13.04.2004
Beginn effektiv: 05.04.2004 Dauer Effektiv: 10.04.2004
Begründung der Abweichungen: -

Vorbereitung Erste Betreuersitzung

Ziel: Aufarbeitung der Dokumentation für erste Betreuersitzung

Beginn geplant: 11.04.2004 Dauer geplant: 14.04.2004
Beginn effektiv: 05.04.2004 Dauer Effektiv: 05.04.2004
Begründung der Abweichungen: -

9.2.2 Analyse

Definition Highlevel Use Cases

Ziel: Definition der Highlevel Use Cases in Form einer Kurzbeschreibung und finden der Aktoren.

Beginn geplant: 15.04.2004 Dauer geplant: 21.04.2004
Beginn effektiv: 12.04.2004 Dauer Effektiv: 12.04.2004
Begründung der Abweichungen: -

Definition Extended Use Cases

Ziel: Detaillierte Dokumentation jedes Use Cases inklusive Aktoren.

Beginn geplant: 22.04.2004 Dauer geplant: 28.04.2004
Beginn effektiv: 12.04.2004 Dauer Effektiv: 18.04.2004
Begründung der Abweichungen: -

Erstellung Use Case Diagramme

Ziel: Erstellung der Use Case Diagramme

Beginn geplant: 15.04.2004 Dauer geplant: 28.04.2004
Beginn effektiv: 12.04.2004 Dauer Effektiv: 18.04.2004
Begründung der Abweichungen: -

Erstellung Systemsequenzdiagramme

Ziel: Erstellung der Systemsequenzdiagramme zu jedem Use Case.
Beginn geplant: 15.04.2004 Dauer geplant: 05.05.2004
Beginn effektiv: 12.04.2004 Dauer Effektiv: 18.04.2004
Begründung der Abweichungen: -

Definition Konzeptionelles Modell

Ziel: Erstellung des konzeptionellen Modells.
Beginn geplant: 29.04.2004 Dauer geplant: 05.05.2004
Beginn effektiv: 18.04.2004 Dauer Effektiv: 09.05.2004
Begründung der Abweichungen: -

Definition Kontrakte

Ziel: Erstellung der Kontrakte
Beginn geplant: 29.04.2004 Dauer geplant: 05.05.2004
Beginn effektiv: 18.04.2004 Dauer Effektiv: 09.05.2004
Begründung der Abweichungen: -

Review / Betreuersitzung

Ziel: Abschluss der Analysephase / Betreuersitzung
Beginn geplant: 11.05.2004 Dauer geplant: 11.05.2004
Beginn effektiv: 11.05.2004 Dauer Effektiv: 11.05.2004
Begründung der Abweichungen: -

9.2.3 Design

Definition Real Use Cases

Ziel: Definition und Beschreibung der Real Use Cases
Beginn geplant: 14.05.2004 Dauer geplant: 20.05.2004
Beginn effektiv: 09.05.2004 Dauer Effektiv: 29.05.2004
Begründung der Abweichungen: -

Definition Systemarchitektur

Ziel: Definition der Systemarchitektur
Beginn geplant: 14.05.2004 Dauer geplant: 20.05.2004
Beginn effektiv: 09.05.2004 Dauer Effektiv: 29.05.2004
Begründung der Abweichungen: -

Definition User Interface

Ziel: Definition und Design des GUI
Beginn geplant: 21.05.2004 Dauer geplant: 27.05.2004
Beginn effektiv: 09.05.2004 Dauer Effektiv: 18.06.2004
Begründung der Abweichungen:
Auf Grund nicht geplante Zeitbeanspruchung im Fach Software Projekt und eine intensive Zeit in meiner Firma hat sich die Diplomarbeit etwas verzögert.

Definition Interaktionsdiagramme

Ziel: Definition der Interaktionsdiagramme
Beginn geplant: 21.05.2004 Dauer geplant: 27.05.2004
Beginn effektiv: 09.05.2004 Dauer Effektiv: 18.06.2004
Begründung der Abweichungen: -

Definition Designklassendiagramm

Ziel: Definition des Designklassendiagramm. Überarbeitung durch Anwenden von Designpatterns.
Beginn geplant: 21.05.2004 Dauer geplant: 27.05.2004
Beginn effektiv: 09.05.2004 Dauer Effektiv: 18.06.2004
Begründung der Abweichungen: -

Review / Bereuersitzung

Ziel: Abschluss der Designphasephase / Betreuersitzung
Beginn geplant: 28.05.2004 Dauer geplant: 04.05.2004
Beginn effektiv: 17.06.2004 Dauer Effektiv: 17.06.2004
Begründung der Abweichungen: -

9.2.4 Implementation

Erstellung der Klassendefinition

Ziel: Erstellung der Klassendefinition und Schnittstellen in MFC gemäss Designklassendiagramm
Beginn geplant: 05.06.2004 Dauer geplant: 11.06.2004
Beginn effektiv: 17.06.2004 Dauer Effektiv: 20.06.2004
Begründung der Abweichungen: -

Implementierung der Methoden

Ziel: Implementierung der Methoden der Applikationslogik.
Beginn geplant: 12.06.2004 Dauer geplant: 25.06.2004
Beginn effektiv: 17.06.2004 Dauer Effektiv: 29.06.2004
Begründung der Abweichungen: -

Erstellung des User Interfaces

Ziel: Erstellung und Design des GUI. Implementierung der GUI-Klassen und Methoden.
Beginn geplant: 12.06.2004 Dauer geplant: 25.06.2004
Beginn effektiv: 09.05.2004 Dauer Effektiv: 18.06.2004
Begründung der Abweichungen: -

Review / Bereuersitzung

Ziel: Abschluss der Implementation / Betreuersitzung
Beginn geplant: 26.06.2004 Dauer geplant: 29.06.2004
Beginn effektiv: 06.07.2004 Dauer Effektiv: 06.07.2004
Begründung der Abweichungen: -

9.2.5 Tests

Testplanung

Ziel: Erstellung der Testcheckliste
Beginn geplant: 30.06.2004 Dauer geplant: 06.07.2004
Beginn effektiv: 05.07.2004 Dauer Effektiv: 12.07.2004
Begründung der Abweichungen: -

Testausführung

Ziel: Testausführung gemäss Testcheckliste
Beginn geplant: 07.07.2004 Dauer geplant: 20.07.2004
Beginn effektiv: 12.07.2004 Dauer Effektiv: 02.08.2004
Begründung der Abweichungen: -

Korrekturen der Fehler

Ziel: Korrekturen allfälliger Fehler
Beginn geplant: 07.07.2004 Dauer geplant: 20.07.2004
Beginn effektiv: 12.08.2004 Dauer Effektiv: 12.08.2004
Begründung der Abweichungen: -

9.2.6 Dokumentation

Dokumentationsvorlage erstellen

Ziel: Dokumentationsvorlage im Microsoft Word erstellen
Beginn geplant: 01.04.2004 Dauer geplant: 05.04.2004
Beginn effektiv: 01.04.2004 Dauer Effektiv: 05.04.2004
Begründung der Abweichungen: -

Fortwährende Dokumentation

Ziel: Laufendes Dokumentieren jedes Entwicklungsschrittes.
Beginn geplant: 05.04.2004 Dauer geplant: 18.07.2004
Beginn effektiv: 05.04.2004 Dauer Effektiv: 12.08.2004
Begründung der Abweichungen: -

Zusammenstellen der Dokumentation

Ziel: Zusammenstellen der gesamten Dokumentation. Ergänzung fehlender Teile.
Beginn geplant: 21.07.2004 Dauer geplant: 03.08.2004
Beginn effektiv: 12.08.2004 Dauer Effektiv: 14.08.2004
Begründung der Abweichungen: -

Reinlesen der Dokumentation / Betreuersitzung

Ziel: Reinlesen der gesamten Dokumentation. Abschluss der Dokumentation.
Beginn geplant: 21.07.2004 Dauer geplant: 11.08.2004
Beginn effektiv: 14.08.2004 Dauer Effektiv: 14.08.2004
Begründung der Abweichungen: -

Vorbereitungen zur Abgabe

Ziel: Drucken der Dokumentation und erstellen der Projekt CD mit allen Projektdateien. Abschluss der Projektarbeit.
Beginn geplant: 14.08.2004 Dauer geplant: 16.08.2004
Beginn effektiv: 14.08.2004 Dauer Effektiv: 16.08.2004
Begründung der Abweichungen: -

Abgabe der Dokumentation

Ziel: Dokumentation abgeben
Beginn geplant: 17.08.2004 Dauer geplant: 17.08.2004
Beginn effektiv: 17.08.2004 Dauer Effektiv: 17.08.2004

9.3 Bestimmungen für die Diplomarbeit

Bestimmungen für die Diplomarbeit

1. Zweck der Bestimmungen

Diese allgemeinen Bestimmungen gehören mit zur Aufgabenstellung der Diplomarbeit. Sie haben den Zweck den Ablauf der Diplomarbeit im Detail zu regeln.

2. Zielsetzungen der Diplomarbeit

- Durch eine Vertiefung in einem Spezialgebiet, soll das Studieren und selbständige Erarbeiten von Grundlagen praktiziert werden.
- Der Diplomand soll die erworbenen Kenntnisse in der Projektführung praktisch erproben, indem er das Projekt "Diplomarbeit" selbständig durchführt.
- Die Diplomarbeit soll von praktischem Nutzen sein. Dadurch kann die Motivation und Freude gefördert werden.

3. Themenvorschläge

Am Ende des 6. Semester werden alle Diplomanwärter eingeladen, 2 Vorschläge für die Diplomarbeit zu machen. Diese Vorschläge können aus dem persönlichen Interessengebiet sein, von der Arbeitgeberfirma stammen, oder aus der Themenliste entnommen werden. Die Themenliste wird im Sekretariat geführt und ist ein Pool der Vorschläge von Fachlehrern, Institutionen und Firmen. Sie wird den Diplomanwärttern zur Auswahl zugestellt.

Die Arbeit sollte möglichst ein in sich abgeschlossenes Thema darstellen. Insbesondere muss die Arbeit bei der Vorführung im Schulhaus aufgebaut und funktionell vorgeführt werden können.

Bei Arbeiten aus dem persönlichen Interessengebiet muss folgendes berücksichtigt werden: Das Produkt der Arbeit soll in dieser Form nicht bereits käuflich sein. Das Thema soll praxisnah sein und realisierbaren, konkret spezifizierten Vorstellungen entsprechen.

Bei Arbeiten von der Arbeitgeberfirma gilt es folgendes zu berücksichtigen: Der Inhalt der Arbeit soll vom alltäglichen Know-how der Firma wesentlich abweichen. Nicht möglich ist z.B. eine Arbeit über die Weiterentwicklung eigener Serieprodukte. Die Arbeit soll vielmehr einen technischen Neuigkeitsgrad, Grundlagen für ein neues Konzept, Einsatz neuerer Technologien usw. im Sinne einer Studien- oder Grundlagenarbeit beinhalten. Eine reine Studienarbeit ist aber nicht möglich. Es muss in jedem Fall eine Realisierung gemacht werden (Muster, Prototyp, erste Software-Version).

4. Themenfestlegung

Die Schulleitung (Schulleiter und Abteilungsleiter) prüft die Vorschläge, stellt Rückfragen und macht eventuell weitere Anregungen im Gespräch mit den Studenten. Die Schulleitung entscheidet dann über das Thema und ordnet einen Betreuer zu.

5. Betreuer

Der Betreuer kann ein Fachlehrer der TSU oder ein Vorgesetzter der Arbeitgeberfirma mit einer höheren Ausbildung sein. Der Studierende kann mit den Themenvorschlägen auch einen Betreuer vorschlagen.

Der zugewiesene Betreuer fasst die Aufgabenstellung. Diese schickt er zur Begutachtung an den zugewiesenen Experten. Nach der Reaktion des Experten wird die reingeschriebene Aufgabenstellung vom Betreuer unterschrieben und an das Sekretariat geschickt. Die Austeilung der Aufgabenstellung erfolgt gemäss Terminplan durch das Sekretariat.

Während der Arbeitsausführung müssen in regelmässigen Abständen mindestens 4 Besprechungen durchgeführt werden. Der Betreuer macht sich Besprechungsnotizen. Beschlüsse werden gegenseitig unterzeichnet. Der Betreuer visiert die Besprechung unter Punkt 15 dieser Bestimmungen. Der Arbeitsverlauf kann kurzfristig zusätzliche Besprechungen erfordern. Telefongespräche gelten nicht als Besprechungen.

Der Betreuer ist kein Fachberater, kennt sich aber in der Materie so gut aus, dass er den Diplomanden durch das Projekt begleiten kann. Er fördert das selbständige Arbeiten des Diplomanden, überwacht den zeitlichen Ablauf und fällt allenfalls notwendige Abweichungsentscheide von der Aufgabenstellung.

Nach der Abgabe korrigiert und bewertet der Betreuer die Arbeit eingehend anhand des Formulars "Beurteilungshilfsblatt Diplomarbeit". Drei Tage vor der Präsentation faxt der Betreuer sein Bewertungsformular an den Experten. Auf Verlangen schickt er auch seine Besprechungsnotizen.

Der Betreuer nimmt zusammen mit dem Experten an der Präsentation der Arbeit durch den Diplomanden teil und bespricht die Notengebung mit dem Experten.

6. Zulassung, Randbedingungen

Zur Diplomarbeit zugelassen wird, wer gemäss den Promotionsbestimmungen im 7. Semester studiert.

Es ist nicht gestattet, für Diplomarbeiten irgendwelche Entschädigungen von Auftraggebern entgegenzunehmen. Die Abwicklung der Diplomarbeit erfolgt gemäss einem separaten, jährlich neuen Terminplan.

7. Experte

Anlässlich der Expertensitzung werden die Themen nach Fachkenntnissen eingeteilt, sodass jeder Experte zirka 3 Arbeiten hat. Der Experte macht eventuelle Anregungen oder Einschränkungen und begutachtet die Aufgabenstellung des Betreuers.

Anhand der abgegebenen Dokumentation macht der Experte eine unabhängige Bewertung der Arbeit. Die Experten treffen sich vor der Präsentation und vergleichen die Arbeiten untereinander, damit eine möglichst ausgeglichene Bewertung erzielt wird.

Während der Präsentation hat der Experte die Leitung. Anschliessend an die Präsentation wird die Note durch den Experten und Betreuer endgültig berechnet und festgelegt.

8. Ausführung der Arbeit

Die Erarbeitung der Diplomarbeit erfolgt schul- und berufsbegleitend während der zweiten Hälfte des Diplomsemesters und einer zusätzlichen Frist. Nachdem der Diplomand Aufgabenanalyse, Vorstudie, Voruntersuchungen und Terminplan erstellt hat, vereinbart er mit dem Betreuer die erste Besprechung. Diese muss spätestens 6 Wochen nach der Austeilung der Aufgabenstellung stattfinden.

9. Umfang der Arbeit

Der zeitliche Umfang der Arbeit soll 200 Stunden nicht überschreiten. Bei grösserem Umfang soll die Arbeit im Team gemacht werden. Der selber verfasste Text (ohne Anhang) darf bei reinen Softwareaufgaben maximal 60 Seiten, bei allen übrigen Arbeiten maximal 80 Seiten umfassen. Es wird erwartet, dass die Arbeit in der Freizeit gemacht wird. Falls eine Firma bezahlte Arbeitszeit zur Verfügung stellt, muss sie unter Punkt 15 ausgewiesen und vom Vorgesetzten visiert werden.

10. Abgabe der Arbeit

Die Abgabe der Diplomarbeit erfolgt in zweifacher Ausführung, mit allen Unterschriften unter Punkt 15 dieser Bestimmungen versehen, an das Sekretariat. Bei Arbeiten mit Software muss ein Datenträger mit kompilierbarer Source und ausführbarem Programm beiliegen.

Der Termin ist gemäss Angabe der Schulleitung. Als Datum gilt dasjenige der Überbringung, bzw. Datum des Poststempels. Zu spät abgegebene Arbeiten werden nicht angenommen, womit ein Diplomabschluss im gleichen Jahr nicht möglich ist. Eine Wiederholung der Diplomarbeit mit einem anderen Thema ist gemäss Promotionsordnung möglich.

11. Vorführung der Arbeit

Die Arbeit wird gemäss Diplom-Prüfungsplan der Schulleitung dem Experten und Betreuer vorgeführt. Der Diplomand kann während maximal 15 Minuten die Arbeit in der Funktion und Ausführung vorführen. Während der restlichen 15 Minuten stellt der Experte Fragen. Diese können auch globaler Art sein (Umfeld der Arbeit, andere Lösungsvarianten).

12. Notenbekanntgabe

Dem Diplomanden wird die Bewertung der Hauptkriterien schriftlich bekanntgegeben. Falls der Diplomand detailliertere Auskünfte über die Bewertung will, kann er diese, innerhalb einer Woche nach dem Prüfungstag, beim Betreuer oder Experten nachfragen.

13. Eigentum der Arbeit

Eine abgegebene Dokumentation bleibt im Eigentum der Schule und kann weiteren Studierenden zugänglich gemacht werden. Die Weiterverwendung darf nur zu Studienzwecken und nicht zu gewerblich- industrieller Nutzung erfolgen. Bei Arbeiten für die Arbeitgeberfirma, oder bei wirtschaftlichen Interessen des Diplomanden verpflichtet sich die Schule zur Geheimhaltung, falls diese gewünscht wird.



Die Arbeit und das erworbene Wissen sind Eigentum des Diplomanden und je nach Beteiligung, eventuell auch der Arbeitgeberfirma.

14. Rekurs

Lassen sich Unstimmigkeiten oder Streitigkeiten in der Bewertung nicht direkt oder mit Hilfe der Schulleitung lösen, ist ein Rekurs möglich. Dieser muss schriftlich, mit Begründungen, innert 14 Tagen nach Bekanntgabe der Noten an den Präsidenten der Aufsichtskommission eingereicht werden.

15. Bestätigungen und Unterschriften

15.1 Besprechungen Betreuer mit Diplomand

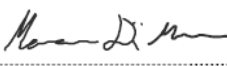
Besprechung	Datum	Visum	Bemerkungen
1			
2			
3			
4			
5			
6			

-> Siehe Protokolle auf folgenden Seiten

15.2 Betreuer

Ich bestätige, die Bestimmungen durchgelesen und den Diplomanden zu einer selbständigen Arbeit angeleitet und betreut zu haben.

Datum: 13.08.04

Unterschrift: 

15.3 Arbeitgeber oder Vorgesetzter

Vom Betrieb zur Verfügung gestellte, bezahlte Arbeitsstunden: 0
(Unterschrift nur falls Zeit zur Verfügung gestellt wurde)

Datum:

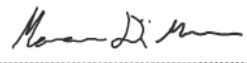
Unterschrift:

15.4 Diplomand

Totale Anzahl Arbeitsstunden für die Diplomarbeit: 200
(Inklusive allenfalls zur Verfügung gestellte, bezahlte Arbeitszeit)

Ich bestätige, nach obigen Bestimmungen gearbeitet und die Arbeit selbständig durchgeführt zu haben.

Datum: 13.08.04

Unterschrift: 

Ausgabe: 5.4.2000, HM

9.4 Protokolle der Betreuersitzungen

Technikerschule Uster

TSU

Protokoll Betreuungssitzung Nr: 1

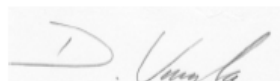
Art der Arbeit	Diplomarbeit
Student	Marco Di Menco
Betreuer	Dominik Venosta
Thema	Kalibriertool für Drucktransmitter
Ort, Datum, Zeit	Wald, 12.5.2004

Arbeitsstand	Pflichtenheft erstellt Analyse bereits teilweise erstellt
Probleme, Fragen	Umfang der Analyse (Rückfrage mit Marcel Müller)
Weiteres Vorgehen	Analyse vervollständigen Design erstellen
Beschlüsse	
Nächster Termin	Nach Absprache

Der Betreuer gibt dem Studenten jeweils eine Kopie. Er sammelt die ausgefüllten Protokolle und bewahrt sie auf bis 1 Monat nach der Präsentation (Ablauf Rekursfrist).

2.5.2001, HM

Unterschrift Betreuer



Unterschrift Student





Technikerschule Uster

TSU

Protokoll Betreuungssitzung Nr: 2

Art der Arbeit	Diplomarbeit
Student	Marco Di Menco
Betreuer	Dominik Venosta
Thema	Kalibriertool für Drucktransmitter
Ort, Datum, Zeit	Wald, 17.6.2004

Arbeitsstand	Analyse vollständig Design vollständig Implementierung begonnen
Probleme, Fragen	
Weiteres Vorgehen	Implementierung weiterfahren
Beschlüsse	
Nächster Termin	Nach Absprache

Der Betreuer gibt dem Studenten jeweils eine Kopie. Er sammelt die ausgefüllten Protokolle und bewahrt sie auf bis 1 Monat nach der Präsentation (Ablauf Rekursfrist). 2.5.2001, HM

Unterschrift Betreuer

Unterschrift Student



Technikerschule Uster

TSU

Protokoll Betreuungssitzung Nr: 3

Art der Arbeit	Diplomarbeit
Student	Marco Di Menco
Betreuer	Dominik Venosta
Thema	Kalibriertool für Drucktransmitter
Ort, Datum, Zeit	Wald, 6.7.2004

Arbeitsstand	Implementierung bis auf einige Details fertig Programm am testen
Probleme, Fragen	
Weiteres Vorgehen	Dokumentation fertigstellung und nochmals überarbeiten
Beschlüsse	
Nächster Termin	Nach Absprache

Der Betreuer gibt dem Studenten jeweils eine Kopie. Er sammelt die ausgefüllten Protokolle und bewahrt sie auf bis 1 Monat nach der Präsentation (Ablauf Rekursfrist). 2.5.2001, HM

Unterschrift Betreuer

Unterschrift Student



Technikerschule Uster

TSU

Protokoll Betreuungssitzung Nr: 4

Art der Arbeit	Diplomarbeit
Student	Marco Di Menco
Betreuer	Dominik Venosta
Thema	Kalibriertool für Drucktransmitter
Ort, Datum, Zeit	Wald, 12.8.2004

Arbeitsstand	Projekt fertig
Probleme, Fragen	Aufbau und Inhalt der Präsentation
Weiteres Vorgehen	
Beschlüsse	
Nächster Termin	keine

Der Betreuer gibt dem Studenten jeweils eine Kopie. Er sammelt die ausgefüllten Protokolle und bewahrt sie auf bis 1 Monat nach der Präsentation (Ablauf Rekursfrist).

2.5.2001, HM

Unterschrift Betreuer

Unterschrift Student

10 Anhang B

10.1 Dokumentation Programmieradapter „Digipack“

Kommunikationsprotokoll für Protokollkonverter

DIGI-PACK

(DGP_V1-1)

Normal Protocol:

Read – Cycle:

	Byte	MSB = „0“	LOW	HIGH	LOW	HIGH	LOW	HIGH
Request	ID.No.	FUNC.	Address		Numb. of Bytes		CRC 16	

	Byte	MSB = „0“	LOW	HIGH	LOW	HIGH		LOW	HIGH
Replay	ID.No.	FUNC.	Address		Numb. of Bytes		n - Bytes	CRC 16	

Read liest *Numb. Of Bytes* Bytes beginnend bei *Address* aus dem entsprechenden Speicher.
 ID.NO ist die Identifikationsnummer des Messkanals.
 CRC16: Berechnete Checksumme nach dem gleichnamigen Verfahren.

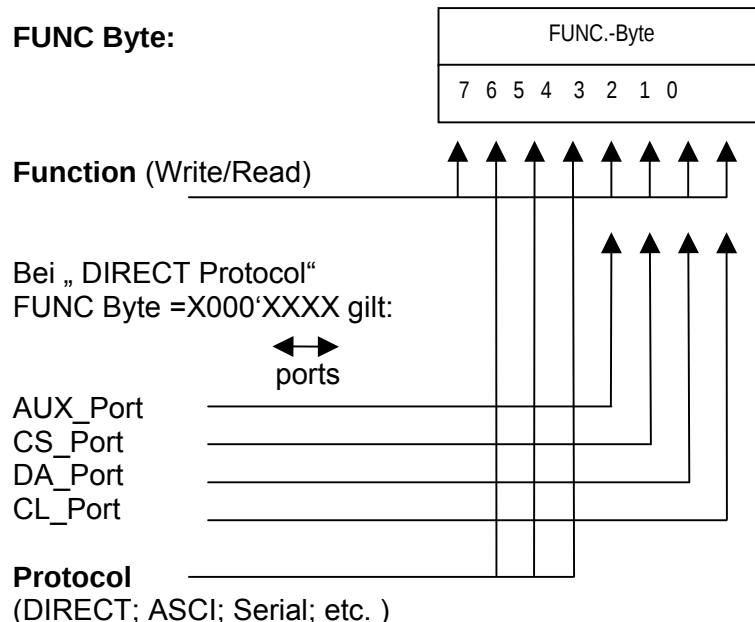
Write – Cycle:

	Byte	MSB = „1“	LOW	HIGH	LOW	HIGH		LOW	HIGH
Request	ID.No.	FUNC.	Address		Numb. of Bytes		n - Bytes	CRC 16	

	Byte	MSB = „1“	LOW	HIGH	LOW	HIGH		LOW	HIGH
Replay	ID.No.	FUNC.	Address		Numb. of Bytes		CRC 16		

Write schreibt *Numb. Of Bytes*, welche in *n-Bytes* stehen beginnend bei *Address* in den entsprechenden Speicher.

FUNC Byte:



000: DIRECT	100: not used
001: ASIC	101: not used
010: UART/SPI/I2C	110: not used
011: CAN	111: HARDWARE

Definition:

Kommunikation:

Alle Daten werden über einen Buffer von 1024 Byte geschrieben und gelesen. Deshalb können Datenblöcke, welche grösser als 1024 Byte sind, nicht in einer Sequenz geschrieben werden!

Das Protokollende wird durch den Empfang einer Lücke von mehr als 3 Bytelängen definiert!

Identitäts-Nummer:

Den Zugriff auf die entsprechenden Plätze wird über die Id.Nr. gehandelt.

Id.Nr. = 0 adressiert alle Plätze gleichzeitig. Jedoch antwortet nur das Pack mit der Id.Nr. 1.

Id.Nr. = Pack-ID adressiert alle Plätze auf dem jeweiligen Pack.

Die Id.Nr. sind dem Platz entsprechend verteilt und werden durch die auf dem Pack vordefinierten ID-Switch (Lötbrücken) oder durch die im EEPROM gespeicherten Werte wie folgt bestimmt.

Id.Nr. = (ID-Switch * 6) + 1

Die Pack-ID werden ebenfalls durch die auf dem Pack vordefinierten ID-Switch (Lötbrücken) bestimmt:

Pack-ID = (ID-Switch +201)

Der ID-Switch besteht aus den Adressen **D7...D3 (0...31)**.

D2 definiert beim Starten ob das **EEProm (D2=1)** oder die **Lötbrücken (D2=0)** gelten.

Alternativ kann die Id.Nr. auch über die Software oder durch die Daisy-Chain vergeben werden. Dabei kann über das **HARDWARE-Protokoll** die **Start-ID** (Id.Nr. des ersten Platzes) und die **Pack-ID** unabhängig vergeben werden.

Beim Daisy-Chain ist jeweils das Pack, bei welchem der **Daisy-In auf Low** und bei welchem der **Daisy-Out auf High** ist, über die Daisy-ID ansprechbar. Die Daisy Ein/Ausgänge können im Hardware-Protokoll manipuliert werden.

Daisy-ID = (255)

Baudrate:

Die Baudrate wird durch die auf dem Pack vordefinierten BAUD-Switch (Lötbrücken) bestimmt:

Die Baudrate wird durch die Adressen **D1...D0** wie folgt definiert:

<u>D1 D0</u>		
0	0	9600 Baud
0	1	19200 Baud
1	0	57600 Baud
1	1	115200 Baud

D2 definiert beim Starten ob das EEPROM (D2=0) oder die Lötbrücken (D2=1) gelten.

Parity und CRC:

Die Parität muss beim Master auf **ODD** = ungerade eingestellt werden.

Die CRC-Checksumme wird nach CRC16 mit dem Generator-Polynom

1010 0000 0000 0001 = A001H berechnet.

Errorhandling:

Entdeckt die Kommunikationsroutine irgendwo im Protokoll einen Fehler, so bricht sie die Kommunikation ab und setzt einen entsprechenden ErrorCode, welcher mittels ErrorRead ausgelesen werden kann. Es wird also im Fehlerfall kein REPLAY gesendet.
Die Anwenderroutine wird auch nur nach korrekter Übertragung (no CRC Error) ausgeführt.

Das **Status**-Byte kann wie folgt gesetzt sein:

00 h	No error	(Kein Fehler bei vorgängiger Übertragung)
01 h	Timeout error	(letzte Verarbeitung abgebrochen)
02 h	Busy error	(letzte Verarbeitung nicht fertig)
03 h	Buffer overflow error	(DatenBuffer überlaufen)
04 h	Parity error	(Parität falsch)
05 h	CRC error	(Checksumme falsch)
06 h	Protocol low data error	(zu wenig Byte empfangen)
07 h	Protocol overrun error	(zu viele Byte empfangen)

Bei ungültigen Funktionscode oder sonstigen Fehlern in den Anwenderroutinen wird im Replay immer im **FUNC.-Byte [FFh]** (Indikation Applikationerror) und in den **Address-Bytes** der empfangene **Funktionscode** und den applikationsspezifischen **Errortyp** (Applikationerror) zurückgesendet.

Die **Address**-Bytes werden bei ungültigem Funktionscode, ungültiger Adresse oder ungültigen Daten mit folgendem Inhalt gesendet:

Address-Low:	FUNC	received Function-Code
Address-High:	00 h	unknow Function error
	01 h	incorrect Address
	02 h	incorrect Data

Application Error Frame

Replay	Byte	Byte	Byte	Byte	LOW	HIGH	LOW	HIGH
	ID.No.	FFh for Error	FUNC.	Error Code	Numb. of Bytes (0000h)		CRC 16	

SERIAL-Protokoll

Read SERIAL-INFO

[20h]

Request	Byte	MSB = „1“	LOW	HIGH	LOW	HIGH		LOW	HIGH
	ID.No.	FUNC. (Baud)	Address 0000		Numb. of Bytes 0002		n – Bytes	CRC 16	
Replay	Byte	MSB = „1“	LOW	HIGH	LOW	HIGH	LOW	HIGH	
	ID.No.	FUNC.	Address		Numb. of Bytes		CRC 16		

Read SERIAL-INFO gibt die Anzahl empfangener Bytes als **Int16** an.

Direkt nach einem Senden (z.B durch Write-UART) wird der Buffer mit dem entsprechenden Counter-Wert gefüllt.

Dieser Wert kann mit dem Befehl *Read SERIAL-INFO* aus dem Buffer ausgelesen werden.

Read SERIAL

[21h]

Request

Byte	MSB = „0“	LOW	HIGH	LOW	HIGH	LOW	HIGH
ID.No.	FUNC.	Address 0000		Numb. of Bytes		CRC 16	

Replay

Byte	MSB = „0“	LOW	HIGH	LOW	HIGH		LOW	HIGH
ID.No.	FUNC.	Address		Numb. of Bytes		n – Bytes	CRC 16	

Read-SERIAL liest die vom letzten Write-Befehl gelesenen Daten aus dem Buffer.

Numb. of Bytes müssen mit den zu erwartenden Daten übereinstimmen, oder kleiner als diese sein, sonst wird ein Applikationserror zurückgegeben.

Send UART [A0h]

Request	Byte	MSB = „1“	LOW	HIGH	LOW	HIGH		LOW	HIGH
	ID.No.	FUNC.	Address BAUD & P & S & Delay		Numb. of Bytes		n – Bytes	CRC 16	

Replay	Byte	MSB = „1“	LOW	HIGH	LOW	HIGH	LOW	HIGH
	ID.No.	FUNC.	Address		Numb. of Bytes		CRC 16	

Send UART sendet die *n*-Bytes direkt an den UART-Ausgang (**CL-PORT**) des Packs. Das Startbit wird LOW (0V) ausgegeben.

Über „Address“ wird der **UART-Mode** bestimmt. Dabei kann die Baudrate, Parität, Stopbit und der Interframe-Space bestimmt werden.

Interframe Space definiert die Pause (in **0.1ms**) zwischen den zusendenden Senderahmen (10/11-Bit Paketen).

Direkt nach dem Senden wird der Buffer mit den empfangnen UART-Daten gefüllt. Dabei wartet der Empfangsport (**DA_PORT**) auf ein Startbit. Es wird jeweils 4 Bytelängen jedoch maximal 35.6ms **und** auf eine Aktivität der DIGI-PACK-Schnittstelle gewartet. Diese empfangnen Werte können danach mit dem Befehl *Read SERIAL* aus dem Buffer ausgelesen werden.

UART-Mode

Baudrate	Address HIGH-Byte (Bit 7...4)	
150 Baud	0000	0xh
300 Baud	0001	1xh
600 Baud	0010	2xh
1200 Baud	0011	3xh
2400 Baud	0100	4xh
4800 Baud	0101	5xh
9600 Baud	0110	6xh
19200 Baud	0111	7xh
not used	1000...1111	-

Parity	Stopbit	Address HIGH-Byte (Bit 3...2)
None	1 Stop-Bit	00
None	2 Stop-Bit	01
Even	1 Stop-Bit	10
Odd	1 Stop-Bit	11

Interframe-Space	Address HIGH & LOW-Byte (Bit 1...0 & 7...0)	
1 Unit = 0.1 ms	xx xxxx'xxxx	0...1023

Send RS232 [A1h]

Request	Byte	MSB = „1“	LOW	HIGH	LOW	HIGH		LOW	HIGH
	ID.No.	FUNC.	Address BAUD & P & S & Delay		Numb. of Bytes		n – Bytes	CRC 16	

Replay	Byte	MSB = „1“	LOW	HIGH	LOW	HIGH	LOW	HIGH
	ID.No.	FUNC.	Address		Numb. of Bytes		CRC 16	

Send RS232 sendet die *n*-Bytes direkt an den RS232-Ausgang (**CL-PORT**) des Packs. Das Startbit wird HIGH (5V) ausgegeben.

Über „Address“ wird der **UART-Mode** bestimmt. Dabei kann die Baudrate, Parität, Stopbit und der Interframe-Space bestimmt werden.

Interframe Space definiert die Pause (in **0.1ms**) zwischen den zusendenden Senderahmen (10/11-Bit Paketen).

Direkt nach dem Senden wird der Buffer mit den empfangnen UART-Daten gefüllt.

Dabei wartet der Empfangsport (**DA_PORT**) auf ein Startbit. Es wird jeweils 4 Bytelängen jedoch maximal 35.6ms **und** auf eine Aktivität der DIGI-PACK-Schnittstelle gewartet.

Diese empfangnen Werte können danach mit dem Befehl *Read SERIAL* aus dem Buffer ausgelesen werden.

Send RS485 [A2h]

Request	Byte	MSB = „1“	LOW	HIGH	LOW	HIGH		LOW	HIGH
	ID.No.	FUNC.	Address BAUD & P & S & Delay		Numb. of Bytes		n – Bytes	CRC 16	

Replay	Byte	MSB = „1“	LOW	HIGH	LOW	HIGH	LOW	HIGH
	ID.No.	FUNC.	Address		Numb. of Bytes		CRC 16	

Send RS485 sendet die *n*-Bytes direkt an den RS485-Ausgang (**CL-PORT & DA-PORT**) des Packs. Das Startbit wird auf CL-Port LOW (0V) und auf DA-PORT HIGH (5V) ausgegeben.

Über „Address“ wird der **UART-Mode** bestimmt. Dabei kann die Baudrate, Parität, Stopbit und der Interframe-Space bestimmt werden.

Interframe Space definiert die Pause (in **0.1ms**) zwischen den zusendenden Senderahmen (10/11-Bit Paketen).

Direkt nach dem Senden wird der Buffer mit den empfangnen UART-Daten gefüllt.

Dabei wartet der Empfangsport (**CL-PORT & DA-PORT**) auf ein Startbit. Ein Startbit wird erkannt, wenn entweder CL-PORT=LOW oder DA-PORT=HIGH wird. Es wird jeweils 4 Bytelängen jedoch maximal 35.6ms **und** auf eine Aktivität der DIGI-PACK-Schnittstelle gewartet.

Diese empfangnen Werte können danach mit dem Befehl *Read SERIAL* aus dem Buffer ausgelesen werden.

Send UART with PULL-HIGH at HALFDUPLEX

[A4h]

Request	Byte	MSB = „1“	LOW	HIGH	LOW	HIGH		LOW	HIGH
	ID.No.	FUNC.	Address BAUD & P & S & Delay		Numb. of Bytes		n – Bytes	CRC 16	

Replay	Byte	MSB = „1“	LOW	HIGH	LOW	HIGH	LOW	HIGH
	ID.No.	FUNC.	Address		Numb. of Bytes		CRC 16	

Send UART_PH sendet die *n*-Bytes direkt an den Seriellen-Ausgang (**DA-PORT**) des Packs. Das Startbit wird LOW (0V) ausgegeben.

Über „Address“ wird der **UART-Mode** bestimmt. Dabei kann die Baudrate, Parität, Stopbit und der Interframe-Space bestimmt werden.

Interframe Space definiert die Pause (in **0.1ms**) zwischen den zusendenden Senderahmen (10/11-Bit Paketen).

Direkt nach dem Senden wird der Buffer mit den empfangnen UART-Daten gefüllt.

Dabei wartet der Empfangsport (**DA_PORT**) auf ein Startbit. Es wird jeweils 4 Bytelängen jedoch maximal 35.6ms **und** auf eine Aktivität der DIGI-PACK-Schnittstelle gewartet.

Diese empfangenen Werte können danach mit dem Befehl Read SERIAL aus dem Buffer ausgelesen werden.

Ein weiterer Ausgang (**CL_PORT**) wird während der Empfangssequenz auf High geschalten und kann somit als Pull-High verwendet werden.

SPI Read Status

[28h]

Request	Byte	MSB = „0“	LOW	HIGH	LOW	HIGH	LOW	HIGH
	ID.No.	FUNC.	Address (0000H)		Numb. of Bytes (0001H)		CRC 16	

Replay	Byte	MSB = „0“	LOW	HIGH	LOW	HIGH		LOW	HIGH
	ID.No.	FUNC.	Address (0000H)		Numb. of Bytes (0001H)		Status	CRC 16	

SPI Read Status liest das Status-Byte eines SPI-kompatiblen Bausteins, wie z.B. ein SPI- EEPROM .

Numb. of Bytes muss 0001 sein, sonst wird ein Applikationserror zurückgegeben.

Der Clock-Ausgang ist CL-PORT

Der Daten-Ein/Ausgang ist DA-PORT (SDI und SDO sind verbunden)

Der Chip-Enable-Ausgang ist CS-PORT

SPI Write Status

[A8h]

Request	Byte	MSB = „1“	LOW	HIGH	LOW	HIGH		LOW	HIGH
	ID.No.	FUNC.	Address (0000H)		Numb. of Bytes (0001H)		Status	CRC 16	

Replay	Byte	MSB = „1“	LOW	HIGH	LOW	HIGH	LOW	HIGH
	ID.No.	FUNC.	Address (0000H)		Numb. of Bytes (0001h)		CRC 16	

SPI Write Status schreibt das Status-Byte eines SPI-kompatiblen Bausteins, wie z.B. ein SPI-EEProm.

Numb. of Bytes muss 0001 sein, sonst wird ein Applikationserror zurückgegeben.

Der Clock-Ausgang ist CL-PORT

Der Daten-Ein/Ausgang ist DA-PORT (SDI und SDO sind verbunden)

Der Chip-Enable-Ausgang ist CS-PORT

Read large SPI [2Ah]

	Byte	MSB = „0“	LOW	HIGH	LOW	HIGH	LOW	HIGH
Request	ID.No.	FUNC.	Address		Numb. of Bytes		CRC 16	

	Byte	MSB = „0“	LOW	HIGH	LOW	HIGH		LOW	HIGH
Replay	ID.No.	FUNC.	Address		Numb. of Bytes		n – Bytes	CRC 16	

Read large SPI liest den Inhalt eines SPI-kompatiblen Bausteins, wie z.B. ein SPI- EEPROM. Die Adresse des EEPROMs muss als 16Bit (2Byte) – Adressformat definiert sein. Die Startadresse wird durch [Address], die Anzahl der ab Startadresse zu lesenden Bytes wird durch [Numb. of Bytes] bestimmt.

Der Clock-Ausgang ist CL-PORT
Der Daten-Ein/Ausgang ist DA-PORT (SDI und SDO sind verbunden)
Der Chip-Enable-Ausgang ist CS-PORT

Write large SPI [AAh]

	Byte	MSB = „1“	LOW	HIGH	LOW	HIGH		LOW	HIGH
Request	ID.No.	FUNC.	Address		Numb. of Bytes		n – Bytes	CRC 16	

	Byte	MSB = „1“	LOW	HIGH	LOW	HIGH	LOW	HIGH
Replay	ID.No.	FUNC.	Address		Numb. of Bytes		CRC 16	

Write large SPI schreibt einen SPI-kompatiblen Bausteins, wie z.B. ein SPI- EEPROM. Die Adresse des EEPROMs muss als 16Bit (2Byte) – Adressformat definiert sein. Die Startadresse wird durch [Address], die Anzahl der ab Startadresse zu lesenden Bytes wird durch [Numb. of Bytes] bestimmt. Ein Write-Enable wird automatisch gesendet. Das Page-Ende ist auf 16 Byte definiert und wird automatisch erkannt. Nach jeder Page wird eine Internal-Write-Cycle-Pause von 5ms abgewartet.

Der Clock-Ausgang ist CL-PORT
Der Daten-Ein/Ausgang ist DA-PORT (SDI und SDO sind verbunden)
Der Chip-Enable-Ausgang ist CS-PORT

Read smal I2C [2Ch]

Request	Byte	MSB = „0“	LOW	HIGH	LOW	HIGH	LOW	HIGH
	ID.No.	FUNC.	Address		Numb. of Bytes		CRC 16	

Replay	Byte	MSB = „0“	LOW	HIGH	LOW	HIGH		LOW	HIGH
	ID.No.	FUNC.	Address		Numb. of Bytes		n – Bytes	CRC 16	

Read smal I2C liest den Inhalt eines I²C-kompatiblen Bausteins, wie z.B. ein I²C- EEPROM. Die Darstellung der EEPROM-Adresse muss im Format: [Controll-Byte + Adress-Byte] (2Byte) vorliegen.

Das Control-Byte wird durch [Address-High], die Startadresse durch [Address-Low], und die Anzahl der ab Startadresse zu lesenden Bytes durch [Numb. of Bytes] bestimmt.

Der Device-Typ „E²Potentiometer (5xh)“ wird erkannt.

Als Folge wird ein verkürztes I²C-Protokoll ohne zweimaliges Senden des Control-Bytes gesendet.

Der Clock-Ausgang ist CL-PORT

Der Daten-Ein/Ausgang ist DA-PORT (SDI und SDO sind verbunden)

Der Ausgang CS_PORT ist während der Aktivität High und kann somit für Pull-High der Datenleitung benutzt werden.

Write smal I2C [ACh]

Request	Byte	MSB = „1“	LOW	HIGH	LOW	HIGH		LOW	HIGH
	ID.No.	FUNC.	Address		Numb. of Bytes		n – Bytes	CRC 16	

Replay	Byte	MSB = „1“	LOW	HIGH	LOW	HIGH	LOW	HIGH
	ID.No.	FUNC.	Address		Numb. of Bytes		CRC 16	

Write smal I2C schreibt einen I²C -kompatiblen Bausteins, wie z.B. ein I²C - EEPROM. Die Darstellung der EEPROM-Adresse muss im Format: [Controll-Byte + Adress-Byte] (2Byte) vorliegen.

Das Control-Byte wird durch [Address-High], die Startadresse durch [Address-Low], und die Anzahl der ab Startadresse zu lesenden Bytes durch [Numb. of Bytes] bestimmt.

Das Page-Ende ist auf 16 Byte definiert und wird automatisch erkannt. Nach jeder Page wird eine Internal-Write-Cycle-Pause von 10ms abgewartet.

Der Clock-Ausgang ist CL-PORT

Der Daten-Ein/Ausgang ist DA-PORT (SDI und SDO sind verbunden)

Der Ausgang CS_PORT ist während der Aktivität High und kann somit für Pull-High der Datenleitung benutzt werden.

HARDWARE-Protokoll

Read Hardware 70h]

Request	Byte	MSB = „0“	LOW	HIGH	LOW	HIGH	LOW	HIGH
	ID.No.	FUNC.	Address (0000H)		Numb. of Bytes (0001H)		CRC 16	

Replay	Byte	MSB = „0“	LOW	HIGH	LOW	HIGH		LOW	HIGH
	ID.No.	FUNC.	Address (0000H)		Numb. of Bytes (0001H)		Port	CRC 16	

Read Hardware liest den durch [ID.No.] angewählten Port aus.
[Address] muss 0000, [Numb. of Bytes] muss 0001 sein, sonst wird ein Applikationserror zurückgegeben.
Das ausgelesene Byte hat folgende Darstellung:

Byte Adr. 0:

MSB				LSB			
REL	-	DA_OE	CL_OE	AUX	CS	DA	CL

Write Hardware [F0h]

Request	Byte	MSB = „1“	LOW	HIGH	LOW	HIGH		LOW	HIGH
	ID.No.	FUNC.	Address (0000H)		Numb. of Bytes (0001H)		Port	CRC 16	

Replay	Byte	MSB = „1“	LOW	HIGH	LOW	HIGH		LOW	HIGH
	ID.No.	FUNC.	Address (0000H)		Numb. of Bytes (0001H)		CRC 16		

Write Hardware beschreibt den durch [ID.No.] angewählten Port aus.
[Address] muss 0000, [Numb. of Bytes] muss 0001 sein, sonst wird ein Applikationserror zurückgegeben.
Das Port-Zustands-Byte hat folgende Darstellung:

Byte Adr. 0:

MSB				LSB			
REL	-	DA_OE	CL_OE	AUX	CS	DA	CL

Read Hardware “Read-Modify-Write” 71h]

	Byte	MSB = „0”	LOW	HIGH	LOW	HIGH	LOW	HIGH
Request	ID.No.	FUNC.	Address (0000H)		Numb. of Bytes (0002H)		CRC 16	

	Byte	MSB = „0”	LOW	HIGH	LOW	HIGH		LOW	HIGH
Replay	ID.No.	FUNC.	Address (0000H)		Numb. of Bytes (0002H)		Port&Rel.	CRC 16	

Read Hardware RMW liest den durch [ID.No.] angewählten Port in der RMW-Darstellung aus.

[Address] und [Numb. of Bytes] müssen gültig sein, sonst wird ein Applikationserror zurückgegeben.

Die ausgelesenen Bytes haben folgende Darstellung:

Byte Adr. 0:

Byte Adr. 1:

MSB				LSB				MSB				LSB			
0	0	0	0	AUX	CS	DA	CL	0	0	0	0	REL	0	DA_OE	CL_OE

Write Hardware “Read-Modify-Write” F1h]

	Byte	MSB = „1”	LOW	HIGH	LOW	HIGH		LOW	HIGH
Request	ID.No.	FUNC.	Address (0000H)		Numb. of Bytes (0002H)		Port&Rel.	CRC 16	

	Byte	MSB = „1”	LOW	HIGH	LOW	HIGH		LOW	HIGH
Replay	ID.No.	FUNC.	Address (0000H)		Numb. of Bytes (0002H)		CRC 16		

Write Hardware RMW beschreibt den durch [ID.No.] angewählten Port.

Dabei kann mit dem Setzen der Enable-Bits durch eine „1“ gewählt werden, welches Bits geändert werden oder durch Setzen einer „0“ welche Bits im alten Zustand belassen werden sollen.

[Address] und [Numb. of Bytes] müssen gültig sein, sonst wird ein Applikationserror zurückgegeben.

Die ausgelesenen Bytes haben folgende Darstellung:

Byte Adr. 0:

Byte Adr. 1:

MSB				LSB				MSB				LSB			
AUX EN	CS EN	DA EN	CL EN	AUX	CS	DA	CL	REL EN	.	DA_OE EN	CL_OE EN	REL	0	DA_OE	CL_OE

Beispiel:

Ein/Ausschalten des angewählten Relais

FUNC. = F1h

Address = 0000h

Numb. of Bytes = 0001h

Data = 88h = Relais einschalten / 80h = Relais ausschalten

Read entire Hardware

74h]

Request	Byte	MSB = 0*	LOW	HIGH	LOW	HIGH	LOW	HIGH
	ID.No.	FUNC.	Address		Numb. of Bytes		CRC 16	

Replay	Byte	MSB = 0*	LOW	HIGH	LOW	HIGH		LOW	HIGH
	ID.No.	FUNC.	Address		Numb. of Bytes		n – Bytes	CRC 16	

Read entire Hardware liest alle Zustände der durch [Address] und [Numb. of Bytes] angewählten Ports. Dabei ist dem 1.Port die Adresse 0000, dem 2. Port die Adresse 0001 etc. zugeordnet.

Der Wert der [ID.No.] ist unbedeutend.

[Address] und [Numb. of Bytes] müssen gültig sein, sonst wird ein Applikationserror zurückgegeben.

Die ausgelesenen Bytes haben folgende Darstellung:

Byte Adr. 0...5:

MSB				LSB			
REL	-	DA_OE	CL_OE	AUX	CS	DA	CL

Write entire Hardware

F4h]

Request	Byte	MSB = 1*	LOW	HIGH	LOW	HIGH		LOW	HIGH
	ID.No.	FUNC.	Address		Numb. of Bytes		n – Bytes	CRC 16	

Replay	Byte	MSB = 1*	LOW	HIGH	LOW	HIGH		LOW	HIGH
	ID.No.	FUNC.	Address		Numb. of Bytes		CRC 16		

Write entiere Hardware beschreibt alle Zustände der durch [Address] und [Numb. of Bytes] angewählten Ports. Dabei ist dem 1.Port die Adresse 0000, dem 2. Port die Adresse 0001 etc. zugeordnet.

Der Wert der [ID.No.] ist unbedeutend.

[Address] und [Numb. of Bytes] müssen gültig sein, sonst wird ein Applikationserror zurückgegeben.

Die gesendeten Bytes haben folgende Darstellung:

Byte Adr. 0...5:

MSB				LSB			
REL	-	DA_OE	CL_OE	AUX	CS	DA	CL

Read Cross Hardware

76h]

	Byte	MSB = „0“	LOW	HIGH	LOW	HIGH	LOW	HIGH
Request	ID.No.	FUNC.	Address		Numb. of Bytes		CRC 16	

	Byte	MSB = „0“	LOW	HIGH	LOW	HIGH	LOW	HIGH
Replay	ID.No.	FUNC.	Address		Numb. of Bytes		n – Bytes	CRC 16

Read Cross Hardware liest den Zustand der Karte geordnet nach Typ (CL, DA, CS...), adressiert durch [Address] und [Numb. of Bytes]. Dabei sind die CL-Ports 1...6 der Adresse 0000, die DA-Ports 1...6 der Adresse 0001 etc. zugeordnet.

Der Wert der [ID.No.] ist unbedeutend.

[Address] und [Numb. of Bytes] müssen gültig sein, sonst wird ein Applikationserror zurückgegeben.

Die Daten-Bytes haben folgende Darstellung:

Address:	MSB							LSB
0000H	-	-	CL6	CL5	CL4	CL3	CL2	CL1
0001H	-	-	DA6	DA5	DA4	DA3	DA2	DA1
0002H	-	-	CS6	CS5	CS4	CS3	CS2	CS1
0003H	-	-	AUX6	AUX5	AUX4	AUX3	AUX2	AUX1
0004H	-	-	CL_OE6	CL_OE5	CL_OE4	CL_OE3	CL_OE2	CL_OE1
0005H	-	-	DA_OE6	DA_OE5	DA_OE4	DA_OE3	DA_OE2	DA_OE1
0006H	-	-	REL6	REL5	REL4	REL3	REL2	REL1

Write Cross Hardware

F6h]

	Byte	MSB = „1“	LOW	HIGH	LOW	HIGH	LOW	HIGH
Request	ID.No.	FUNC.	Address		Numb. of Bytes		n – Bytes	CRC 16

	Byte	MSB = „1“	LOW	HIGH	LOW	HIGH	LOW	HIGH
Replay	ID.No.	FUNC.	Address		Numb. of Bytes		CRC 16	

Write Cross Hardware beschreibt den Zustand der Karte geordnet nach Typ (CL, DA, CS...), adressiert durch [Address] und [Numb. of Bytes]. Dabei sind die CL-Ports 1...6 der Adresse 0000, die DA-Ports 1...6 der Adresse 0001 etc. zugeordnet.

Der Wert der [ID.No.] ist unbedeutend.

[Address] und [Numb. of Bytes] müssen gültig sein, sonst wird ein Applikationserror zurückgegeben.

Read Hardware Configuration 78h]

Request	Byte	MSB = „0“	LOW	HIGH	LOW	HIGH	LOW	HIGH
	ID.No.	FUNC.	Address		Numb. of Bytes		CRC 16	

Replay	Byte	MSB = „0“	LOW	HIGH	LOW	HIGH	LOW	HIGH
	ID.No.	FUNC.	Address		Numb. of Bytes		n – Bytes	CRC 16

Read Hardware Configuration liest die aktuellen Einstellungen der Karte aus. Die Parameter sind adressiert durch [Address] und [Numb. of Bytes]. [Address] und [Numb. of Bytes] müssen gültig sein, sonst wird ein Applikationserror zurückgegeben.

Die Parameter sind wie folgt den Adressen zugeordnet:

Address:	MSB	LSB
0000H	* Baudrate (0=9600, 1= 19200, 2=57600, 3=115200)	
0001H	* Pack-Id. (0...255)	
0002H	* Start-Id. (0...255)	
0003H	-	-
0004H ... 0007H	Reseve (4 Byte)	
000FH ... 0008H	Version-Info (8 Char of ASCII)	

* Diese Werte werden ins EEPROM gespeichert (Adresse 3F0H ... 3FFH)

Write Hardware Configuration F8h]

Request	Byte	MSB = „1“	LOW	HIGH	LOW	HIGH	LOW	HIGH
	ID.No.	FUNC.	Address		Numb. of Bytes		n – Bytes	CRC 16

Replay	Byte	MSB = „1“	LOW	HIGH	LOW	HIGH	LOW	HIGH
	ID.No.	FUNC.	Address		Numb. of Bytes		CRC 16	

Write Hardware Configuration überschreibt die aktuellen Einstellungen der Karte. Die neue Konfiguration ist unmittelbar nach der Ausführung des Befehles gültig. Die Parameter sind adressiert durch [Address] und [Numb. of Bytes]. [Address] und [Numb. of Bytes] müssen gültig sein, sonst wird ein Applikationserror zurückgegeben.

Hardware EEPROM Read

[7Ah]

Request	Byte	MSB = „0“	LOW	HIGH	LOW	HIGH	LOW	HIGH
	ID.No.	FUNC.	Address		Numb. of Bytes		CRC 16	

Replay	Byte	MSB = „0“	LOW	HIGH	LOW	HIGH		LOW	HIGH
	ID.No.	FUNC.	Address		Numb. of Bytes		n – Bytes	CRC 16	

Hardware EEPROM Read liest den Inhalt des auf der Karte befindenden EEPROM.

Die Adresse des EEPROMs muss als 16Bit (2Byte) – Adressformat definiert sein.

Die Startadresse wird durch [Address], die Anzahl der ab Startadresse zu lesenden Bytes wird durch [Numb. of Bytes] bestimmt.

Hardware EEPROM Write

[FAh]

Request	Byte	MSB = „1“	LOW	HIGH	LOW	HIGH		LOW	HIGH
	ID.No.	FUNC.	Address		Numb. of Bytes		n – Bytes	CRC 16	

Replay	Byte	MSB = „1“	LOW	HIGH	LOW	HIGH	LOW	HIGH
	ID.No.	FUNC.	Address		Numb. of Bytes		CRC 16	

Hardware EEPROM Read beschreibt den Inhalt des auf der Karte befindenden EEPROM.

Die Adresse des EEPROMs muss als 16Bit (2Byte) – Adressformat definiert sein.

Die Startadresse wird durch [Address], die Anzahl der ab Startadresse zu schreibenden Bytes wird durch [Numb. of Bytes] bestimmt.

Die Grösse des Speichers ist abhängig vom eingesetzten EEPROM:

25LC80 = 1024 Byte

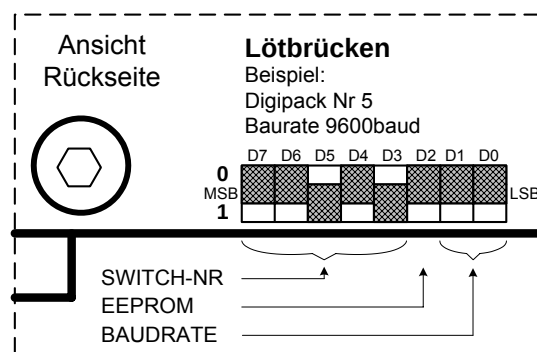
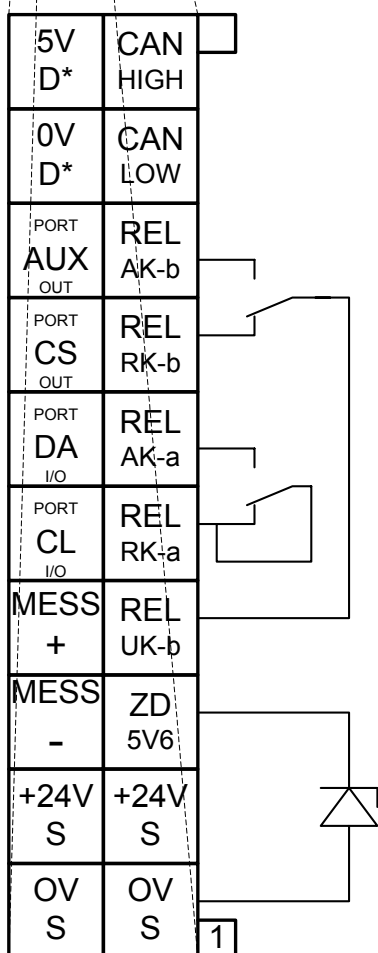
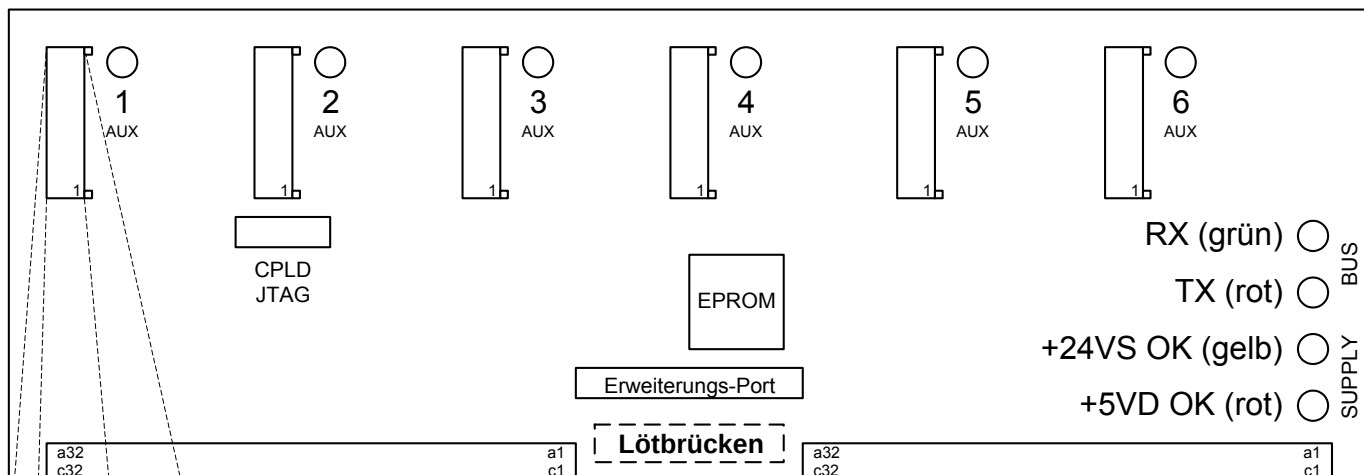
25LC320 = 4096 Byte

25LC640 = 8192 Byte

...

Die Adressen 3F0H ... 3FFH sind für die Digi-Pack-Parameter Baudrate, Pack-Id etc. vergeben und sollten vom Anwender nicht überschrieben werden!

Anschlussbelegung Stecker DIGI-PACK



BAUDRATE (RS485):

00 = 9600 baud
01 = 19200 baud
10 = 57600 baud
11 = 115200 baud

SWITCH-NR:

ID 1. Platz = (Switch-Nr * 6) + 1
Gültige Switch-Nr (0....31)

EEPROM/SWITCH:

0 = Lötbrücken
gelten für ID_Nr.
und Baudrate
1 = EEPROM
bestimmt ID_Nr.
und Baudrate

Beispiel für 9600 Baud:

Pack 1...6	—00000000—	
Pack 7...12	—00001000—	
Pack 13...18	—00010000—	
Pack 19...24	—00011000—	
Pack 25...30	—00100000—	
Pack 31...36	—00101000—	
Pack 37...42	—00110000—	
Pack 43...48	—00111000—	
Pack 49...54	—01000000—	
Pack 55...60	—01001000—	
...		
Pack 187 ... 192	—11111000—	

* Galvanisch getrennt