Anforderungsdefinition

Dynamische En- und Decodierung von binären Nachrichten



Neutorstraße 13  
5020 Salzburg

AUSTRIA

Tel: +43 (662) 276198-11

Fax: +43 (662) 276198-98

Mail: [office@breanos.com](mailto:office@breanos.com)

|  |  |
| --- | --- |
| Datei | Document1 |
| Datum | 04.09.2018 |

Inhaltsverzeichnis

[1 Änderungsverzeichnis 1](#_Toc508801401)

[2 Anforderungsbeschreibung 1](#_Toc508801402)

[2.1 Einführung 1](#_Toc508801403)

[2.1.1 Inkludierter Umfang 1](#_Toc508801404)

[2.1.2 Exkludierter Umfang 1](#_Toc508801405)

[2.2 Geplantes Vorgehen 1](#_Toc508801406)

[2.3 Aufwand 1](#_Toc508801407)

[2.3.1 Anmerkungen 1](#_Toc508801408)

[2.4 Risiken 1](#_Toc508801409)

[2.4.1 Risiko 1 1](#_Toc508801410)

[2.4.1.1 Beschreibung 1](#_Toc508801411)

[2.4.1.2 Folgen 1](#_Toc508801412)

[2.4.1.3 Gegenmaßnahmen 1](#_Toc508801413)

[2.4.1.4 Wahrscheinlichkeit 1](#_Toc508801414)

[2.5 Erfordernisse 1](#_Toc508801415)

[2.5.1 Requirement 1 1](#_Toc508801416)

# Änderungsverzeichnis

|  |  |  |  |
| --- | --- | --- | --- |
| Datum | Version | Verfasser | Beschreibung |
| 03.09.2018 | 1 | Dominik Hutterer | Initialversion zur internen Verwendung |

# Anforderungsbeschreibung

## Referenzen

Als Referenz kann die Dokumentation der Kommunikation zwischen PLC und BREANOS Software herangezogen werden, sofern dies via binär codierten Nachriten geschieht. (bspw. Wuhan, Alfing, Sifang Geely …)

## Einführung

Für die bestehende Kommunikation müssen jedes Mal die jeweiligen Nachrichtenpakete im Code definiert werden, dies sollte in Zukunft dynamisch via XML vonstattengehen.

Dazu muss eine XML Struktur geschaffen werden, die weitestgehend dynamisch sowohl Header als auch Nutzdaten definieren kann. Im Allgemeinen besteht ein Nachrichtenpaket aus einem fix definiertem Header, mit elementaren Informationen zum Paket selbst, und den angehängten Nutzdaten, die je nach Pakettype variieren.

Nutzdaten wie Header bestehen jeweils aus einzelnen Datenfeldern, deren Eigenschaften im XML festgelegt werden müssen bzw. können. Diese umfassen:

* Datentyp: Datentyp des Feldes basierend auf den Basisdatentypen (Bool, Byte, I16, I32, I64, Float, Double, String, DateTime…)
* Bezeichung: Eindeutiger Feldname
* Adresse/Offset: Startposition im Bytestream
* Länge: Teilweise fix, muss diese bei gewissen Typen angegeben werden (String).

Die En- bzw. Decodierung der oben genannten Felder geschieht über die bereits bestehenden Converter aus den Vorprojekten bzw. aus dem Part Tracking System – Framework. Diese werden bei Bedarf erweitert. Konzipiert sind die Converter für die jeweiligen Zielsystem (S7, etc.).

Für die weiteren Projekte wird ein neuer Testclient, basierend auf WPF, geschaffen der die dynamische Paketverwaltung implementieren kann.

### Inkludierter Umfang

#### De- und Encodierung

Aus dem Team-Projekt PTS.Messages werden vorerst folgende Klassen bzw. Interfaces übernommen:

* *IMessageConverter*
* *MessageConverter*
* *S7Converter*

Aus diesen Klassen werden für das neue Framework folgende Projekte gebildet:

* .Contracts
* Daipan.Core.Messaging.General
* Daipan.Core.Messaging.Siemens

Die Definition von *BcdValue* in *S7Converter* muss als eine eigene Datei in die *Contracts* verschoben werden. Ebenso müssen in die Schnittstelle *IMessageConverter* zwei zusätzliche Funktionalitäten implementiert werden:

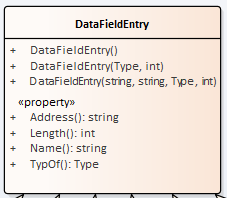
* public void SetBcd(byte[] data, BcdValue value, int index, int length)
* public BcdValue GetBcd(byte[] data, int index, int length = -1)

Die Implementierung der obigen Funktionen in *S7Converter* muss entsprechend in *MessageConverter* verschoben werden.

#### XML Codierter Nachrichtenaufbau

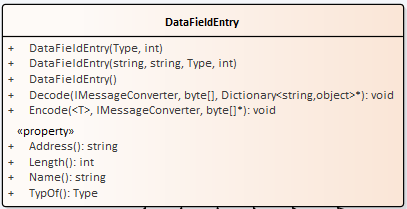
Wie oben beschrieben benötigt jedes Feld innerhalb eines Datenpakets eine Beschreibung im XML Format. Diese muss **Name, Type, Länge,** sowie **Adresse** beinhalten. Implementiert müssen alle C# Basisdatentypen werden, also: **Bool, Byte, Int16, UInt16, Int32, UInt32, Int64, UInt64, DateTime, Float, Double und String**. Zusätzlich muss das PLC-Format **BCD** implementiert werden. Daher empfiehlt es sich zur XML-Serialisierung für jeden möglichen Datentypen eine eigene Klasse anzulegen, die von einer gemeinsamen Basisklasse abstammt.

Die Basisklasse sollte dabei wie folgt aufgebaut sein:



Diese Vorgehensweise ist deshalb effizient, da Methoden zum De- bzw. Encodieren angelegt werden können, die später von den jeweiligen Childs überschrieben werden können und konstante Eigenschaften wie Type und in den meisten Fällen Länge direkt im jeweiligen Konstruktor hinterlegt sind.

Die direkte En- bzw. Decodierung erfolgt steuerungsspezifisch, daher muss der jeweiligen Funktion eine Implementierung der entsprechenden Schnittstelle übergeben werden. Weiters ist z beachten, dass der ursprüngliche Byte-Stream zum Auslesen, wie zur Modifikation als gesamte Einheit übergeben werden muss, da Adressen etc. entsprechend abgestimmt sind.



Die überschriebenen Encode und Decode Methode rufen die jeweils benötigte Methode im IMessageConverter auf.

#### Implementierung in PackageSenderClient

#### Schaffung einer eigenen WPF-Application

### Exkludierter Umfang

## Geplantes Vorgehen

Was haben wir vor?

## Aufwand

Der Aufwand wird wie folgt geschätzt.

|  |  |
| --- | --- |
| Aufgabe | Mannstunden |
| Aufgabe 1 | 3 |
| Aufgabe 2 | 2 |
| Aufgabe 3 | 14 |
| Gesamt | 19 |

### Anmerkungen

Was gehört zum Aufwand noch erwähnt?

## Risiken

### Risiko 1

#### Beschreibung

Worum geht es?

#### Folgen

Was kann passieren?

#### Gegenmaßnahmen

Wie kann das Risiko gelindert werden? Wie kann es vermieden werden? Wenn das Risiko eintritt, wie wird vorgegangen?

#### Wahrscheinlichkeit

Wie hoch schätzen wir die Wahrscheinlichkeit, dass das Risikoszenario auftritt?

## Erfordernisse

Wovon gehen wir als Grundlage der Erfüllung des Auftrags aus? Was muss uns zur Verfügung gestellt werden?

### Requirement 1

Beschreibung einer Voraussetzung