

**Dokumentation IPA**  
 EOL Kapazitiver Tankfüllstandsgeber

Auszubildender Informatiker Fachrichtung Applikationsentwicklung

Hallauer Joel

# Inhaltsverzeichnis

[1 Inhaltsverzeichnis 1](#_Toc37257338)

[2 Glossar 3](#_Toc37257339)

[3 Vorwort 4](#_Toc37257340)

[3.1 Zweck des Dokuments 4](#_Toc37257341)

[3.2 Allgemeines 4](#_Toc37257342)

[3.2.1 Verwendete Tools 4](#_Toc37257343)

[3.2.2 Vorgehensmodell 4](#_Toc37257344)

[4 Informieren 5](#_Toc37257345)

[4.1 Ausgangslage 5](#_Toc37257346)

[4.1.1 Ist-Stand 5](#_Toc37257347)

[4.1.2 Soll-Stand 5](#_Toc37257348)

[4.1.3 Anforderungen 6](#_Toc37257349)

[4.2 Ziele 6](#_Toc37257350)

[4.2.1 Muss-Ziele 6](#_Toc37257351)

[4.2.2 Kann-Ziele 6](#_Toc37257352)

[5 Planen 7](#_Toc37257353)

[5.1 Terminplan 7](#_Toc37257354)

[5.1.1 Terminplan Grob 7](#_Toc37257355)

[5.1.2 Terminplan Detailiert 7](#_Toc37257356)

[5.2 Meilensteine 9](#_Toc37257357)

[5.3 Ablauf EOL 10](#_Toc37257358)

[5.4 Vordefinierte Testfälle 11](#_Toc37257359)

[5.4.1 Kommunikation 11](#_Toc37257360)

[5.4.2 EOL 11](#_Toc37257361)

[5.5 Prüfplan 13](#_Toc37257362)

[5.5.1 Testing aller Tests 13](#_Toc37257363)

[5.5.2 Einzeltest 13](#_Toc37257364)

[6 Entscheiden 14](#_Toc37257365)

[7 Realisieren 14](#_Toc37257366)

[7.1 Kommunikation DLL 14](#_Toc37257367)

[7.1.1 Implementierung 14](#_Toc37257368)

[7.2 Tankfüllstandsgeber EOL 18](#_Toc37257369)

[7.2.1 Enviroment 18](#_Toc37257370)

[7.2.2 Userinterface 20](#_Toc37257371)

[7.2.3 EA-Modul 25](#_Toc37257372)

[7.2.4 State Maschine 29](#_Toc37257373)

[7.2.5 Lesen des Wasser Standes an DUT 30](#_Toc37257374)

[7.2.6 Alarm Test 31](#_Toc37257375)

[8 Kontrollieren 33](#_Toc37257376)

[8.1 Prüffälle Kommunikation 33](#_Toc37257377)

[8.1.1 TK01 - Lesen der Analogen Inputs 33](#_Toc37257378)

[8.1.2 TK02 - Lesen der Digitalen Inputs 33](#_Toc37257379)

[8.1.3 TK03 - Schreiben der Digitalen Outputs 34](#_Toc37257380)

[8.1.4 TK04 - Lesen Software Version 35](#_Toc37257381)

[8.1.5 TK05 - Schreiben/Lesen der B-Nummer 35](#_Toc37257382)

[8.1.6 TK05 - Schreiben/Lesen deS NMEA-Codes 35](#_Toc37257383)

[8.2 Prüfprotokoll Kommunikation 36](#_Toc37257384)

[8.3 Prüfffälle Tankfüllstandsgeber EOL 37](#_Toc37257385)

[8.4 Soll-Ist Zeitvergleich 38](#_Toc37257386)

[9 Auswerten 40](#_Toc37257387)

[9.1 Reflexion 40](#_Toc37257388)

[9.1.1 Allgemein 40](#_Toc37257389)

[9.1.2 Gelerntes 40](#_Toc37257390)

[10 Abbildungsverzeichnis 41](#_Toc37257391)

# Glossar

|  |  |
| --- | --- |
| Fachwort | Erklärung |
| BCD | Binary Coded Decimal: dualcodierte Dezimalziffer |
| DLL | Dynamic Link Library:  Dynamische Programmbibliothek die Code und Daten enthält. |
| DUT | Device Under Test: Prüfling |
| EOL | „End of Line“ Endkontroll Software, welche ein Gerät auf seine Korrektheit in jeder Hinsicht prüft. |
| \*.gui Datei | Userinterface Datei welche von der verwendeten Entwicklungsumgebung CVI verwendet wird. |
| NMEA2000 | NMEA2000 (National Marine Electronics Association) ist ein Netzwerkprotokoll und ein einheitlicher Standard zur Datenübertragung zwischen unterschiedlichen Geräte. |
| PTS | Datenblatt |
| State-maschine | Eine State Maschine hat eine bestimmte Anzahl von definierten Zuständen. Durch das Setzen eines Zustandes führ die Applikation den für diesen Zustand vorgesehen Code aus. |

# Vorwort

## Zweck des Dokuments

Dieses Dokument dient dazu, die Schritte in der praktischen Arbeit von Joel Hallauer nachvollziehbar darzulegen. Es werden alle Arbeitsschritte sowie Gedankengänge zur Arbeit «EOL für Kapazitiver Tankfüllstandsgeber» dokumentiert und beschrieben.

## Allgemeines

### Verwendete Tools

|  |  |
| --- | --- |
| Tool | Verwendung |
| SlickEdit Pro (v 19.0.11) | Codeing, Codegestaltung |
| CVI Labwindows (v 15.0.1) | Kompilieren des Codes, GUI Gestaltung |
| Notepad++ (v 7.5.9) | Temporäre Notizen |
| GanttProject (v 2.8.5) | Terminplan |
| Microsoft Word | Dokumentation und Aufgabenstellung |
| Github |  |

### Vorgehensmodell

Für die Arbeit zur EOL für einen Kapazitiven Tankfüllstandsgeber wird das Vorgehensmodell IPERKA verwendet.

**Informieren:**Was für Ziele wurden definiert?  
Wie sieht die Ausgangslage aus?  
Welche Mittel stehen zur Verfügung?

**Planen:**Wann sollen die Arbeiten erledigt werden (Terminplan/Meilensteine)?  
Was muss getestet werden?

**Entscheiden:**Ablauf der EOL bestimmen

**Realisieren:**Programmierung der KommunikationProgrammierung der EOL

**Kontrollieren:**Testen der Kommunikation  
Testen der EOL

**Auswerten:**Reflexion

# Informieren

## Ausgangslage

### Ist-Stand

Aktuell besteht bereits eine EOL-Vorrichtung für analoge Tankfüllstandgeber. Jedoch ist diese Vorrichtung Steuerungs- und Softwaremässig veraltet. Für die Steuerung sind bereits keine Ersatzteile mehr erhältlich. Auf der EOL-Vorrichtung können lediglich zwei analoge Tankfüllstandsgeber getestet werden:

* **Tankfüllstandsgeber Source-Mode:**  
  liefert je nach Füllstand des Tankes, eine Spannung, welche vom Anzeigeinstrument ausgewertet wird und dann entsprechend anzeigt
* A close up of a blue wall

  Description automatically generatedA picture containing indoor, table, desk, sitting

  Description automatically generated**Tankfüllstandsgeber Sink-Mode:**  
  verbraucht je nach Füllstand des Tankes, einen Strom. Dieser Stromverbrauch wird vom Anzeigeinstrument ausgewertet und entsprechend angezeigt

Abbildung 1 kapazitiver Tankfüllstandsgeber

Abbildung 2 Vorrichtung EOL kapazitiver Tankfüllstandsgeber

### Soll-Stand

Die bestehende EOL-Vorrichtung soll Hard- und Softwaremässig auf einen aktuellen Stand gebracht werden. Neu soll neben den analogen Tankfüllstandgebern auch ein digitaler Tankfüllstandgeber getestet werden können. Dieser Kommuniziert neu über NMEA2000 wodurch diese neu implementiert werden muss. Die Überarbeitung der Hardware ist nicht Bestandteil dieser IPA.

### Anforderungen

* Die Steuersoftware für die Endkontrolle „kapazitiver Tankfüllstandsgeber“ muss neu mit aktuellerer Technik (Windows, C, Ethernet, ...) erstellt werden.
* Der Ablauf der neuen Software muss gemäss der Prüfspezifikation (Dokument 10371354.pdf) erstellt werden.
* Die NMEA2000 Kommunikation muss für das Produkt angepasst werden (TPL\_PTS\_Z10006660\_AA.pdf).

## Ziele

### Muss-Ziele

|  |  |
| --- | --- |
| Ziel | Kriterium |
| Vorrichtung über GUI steuerbar | Ablauf muss über GUI gestartet werden können.  Programmzustand muss über GUI ersichtlich sein. |
| Die Artikeldefinitionen aus Ini-Datei laden | Alle Artikel aus bestehender Ini-Datei fehlerfrei einlesen |
| Zu prüfender Artikel aus Liste auswählbar | Alle Artikel aus Ini-Datei werden zur Auswahl aufgelistet |
| Analoge kapazitive Tankfüllstandsgeber prüfbar | Analoge kapazitive Tankfüllstandsgeber müssen geprüft werden können, gemäss Pruefvorschr\_98601240\_.pdf (Ablauf) Kapitel 4.3 |

### Kann-Ziele

|  |  |
| --- | --- |
| Ziel | Kriterium |
| Diagnosepanel Inputs und Outputs der Vorrichtung | Alle Ein-/Ausgänge beobachtbar/steuerbar |
| Digitale kapazitive Tankfüllstandsgeber prüfbar | Digitale kapazitive Tankfüllstandsgeber müssen geprüft werden können, gemäss Pruefvorschr\_98601240\_.pdf (Ablauf) Kapitel 4.3 und TPL\_PTS\_Z10006660\_AA.pdf (Kommunikation) |

# Planen

## Terminplan

### Terminplan Grob

Im Groben Terminplan ist zu erkennen, an welchen Tagen vorgesehen wird, welche Arbeit zu verrichten.



Abbildung 3 Terminplan Text

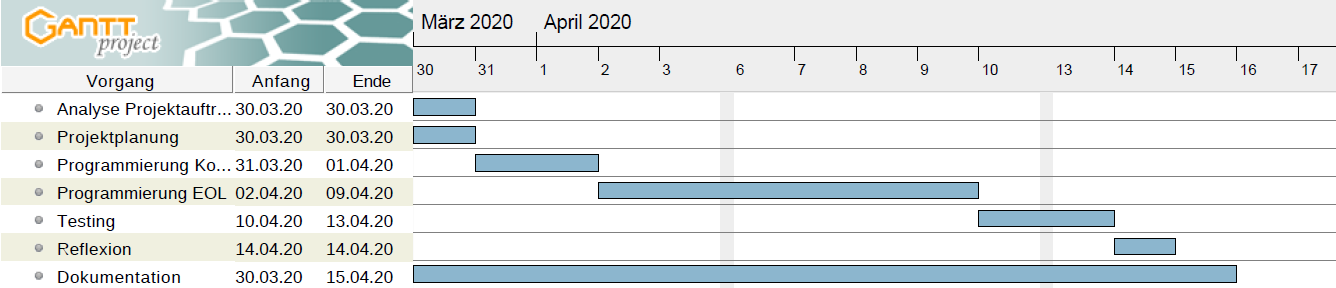


Abbildung 4 Terminplan Grafisch

### Terminplan Detailiert

Der Detaillierte Terminplan dient dazu, die Stunden, welche für eine Arbeit benötigt werden zu planen.

|  |  |
| --- | --- |
| Arbeit | Geplante Zeit [h] |
| Informieren |  |
| * Analyse des Projektauftrages | 0.5 |
| * Ist Stand Beschreiben | 0.5 |
| * Soll Stand beschreiben | 0.25 |
| * Muss- und Kann-Ziele beschreiben und definieren | 0.25 |
| Planen | 1.5 |
| * Terminplan Grob | 0.25 |
| * Terminplan detailliert | 1 |
| * Meilensteinplan | 0.5 |
| * Ablauf der EOL | 1.5 |
| * Definition de Testfälle | 1 |
| * Definition Prüfplan | 0.5 |
| Entscheiden |  |
| * Allfällige Entscheidungen | 1 |
| Realisieren |  |
| * Implementation der Kommunikation   + Analoge Inputs   + Digitale Inputs   + Digitale Outputs | 10 |
| * Implementation der EOL   + Kommunikation mit EA-Modul   + Kommunikation mit DUT   + Prüfungsablauf | 38 |
| Kontrollieren |  |
| * Prüfen + Massnahmen Kommunikation | 4 |
| * Prüfen + Massnahmen EOL | 4 |
| * Soll- Ist Stand überprüfen (Zeit nach Terminplan) | 2 |
| Auswerten |  |
| * Reflexion der Arbeit | 4 |
| Dokumentation | 20 |
| Total: | 89.25 |

## Meilensteine

|  |  |  |
| --- | --- | --- |
| Datum | Meilenstein | Zuweisung |
| 30.03.2013 | Projektstart | HaJ |
| 30.03.2013 | Projektplan | HaJ |
| 02.04.2013 | Kommunikation implementiert | HaJ |
| 10.04.2013 | EOL Implementiert | HaJ |
| 14.04.2013 | Getestet | HaJ |
| 15.04.2013 | Reflektiert und kontrolliert | HaJ |
| 15.04.2013 | Projektende | HaJ |

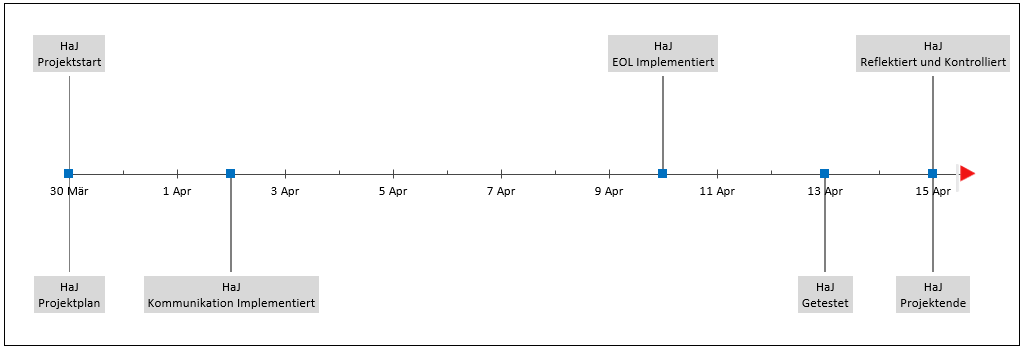


Abbildung 5 Meilenstein Planung

## Ablauf EOL

Die Prüfung eines Tankfüllstandgebers durch die EOL soll wie folgt ablaufen:

1. Nachdem der DUT an die Vorrichtung angeschlossen wurde, sendet die EOL einen Login Command an den Prüfling, um mit ihm kommunizieren zu können.
2. Nun wird der Wasserstand in den Tanks so lange erhöht, bis er laut dem Referenzgerät 100 % erreicht hat.
3. Somit kann nun der Wert vom Prüfling abgelesen werden.
4. Die Ausgelesenen Werte werden nun grafisch dargestellt. Die Prüfende Person justiert den Tankgeber so, dass der Wert mit dem Referenzgeber übereinstimmt.
5. Wenn es sich beim DUT um ein Waste-Water Tankfüllstandsgeber handelt, wird geprüft, ob ein Alarmsignal vorhanden ist. Ist dies der Fall, kann die Prüfung fortgesetzt werden.
6. Nun senkt man den Wasserstand wieder, bis das Referenzgerät 50 % anzeigt.
7. Ist der Wasserstand nun bei 50 % kann der Wert beim Prüfling ausgelesen werden.
8. Wenn dir Wert des DUT’s und des Referenzgerätes übereinstimmen, so wird der Wasserstand ein letztes Mal auf 0% gesenkt.
9. Die Werte aus dem Referenzgerät und dem Prüfling werden erneut verglichen.
10. Wess es sich beim DUT um ein Fresh-Water Tankfüllstandsgeber handelt, wird geprüft. Ob ein Alarmsignal vorhanden ist. Ist dies der Fall, so wird der geprüfte Tankfüllstandsgeber als «in Ordnung» angesehen und die Prüfung wird abgeschlossen. Stimmen die Werte jedoch nicht überein, wird eine Ausgabe erfolgen, welche über den Fehler berichtet und die Prüfung wird abgebrochen.

Auf der Vorrichtung ist genug Platz, um bis zu acht Prüflinge zur selben Zeit anzuschliessen. Jedoch kann nur das Signal eines Prüflings zur gleichen Zeit ausgelesen werden.

Mit einem Potentiometer an der Vorrichtung ist es jedoch möglich, den DUT auszuwählen, von welchem man aktuell die Daten auslesen möchte.

Um Zeit zu sparen, wird nun nachdem man den Wasserspiegel auf den richtigen Wert erhöht oder gesenkt hat, alle Zehn Prüflinge prüfen indem man mit dem Potentiometer das Signal umschaltet.

Erst wenn von allen Prüflingen der Wert eingelesen und überprüft wurde, fährt die EOL im Test weiter.

## Vordefinierte Testfälle

Die verschiedenen Testfälle werden im Vorhinein definiert. Die einzelnen Tests werden nach dem ebenfalls im Vorhinein ausgearbeiteten Prüfplan durchgeführt. Sobald alle Tests durchlaufen wurden, werden wenn nötig Massnahmen definiert und ergriffen.

### Kommunikation

|  |  |  |
| --- | --- | --- |
| ID | Test | Erwartetes Resultat |
| TK01 | Lesen der Analogen Inputs | Die Analogen Inputs können korrekt vom Tankfüllstandsgeber eingelesen werden.  Die Analogen Inputs können sowohl einzeln wie auch alle zusammen eingelesen werden |
| TK02 | Lesen der Digitalen Inputs | Die Digitalen Inputs können korrekt vom Tankfüllstandsgeber eingelesen werden.  Die Digitalen Inputs können sowohl einzeln wie auch alle zusammen ausgelesen werden |
| TK03 | Schreibend er Digitalen Outpus | Die Digitalen Outputs können korrekt am Tankfüllstandsgeber gesetzt werden. |
| TK04 | Lesen Software Version | Die Software Version soll korrekt eingelesen werden können. |
| TK05 | Schreiben/Lesen der B-Nummer | Die B-Nummer soll korrekt geschrieben und gelesen werden können.  Nach einem Neustart soll der Tankfüllstandsgeber dieselbe B-Nummer haben |
| TK06 | Schreiben/Lesen des NMEA-Codes | Der NMEA-Code soll korrekt geschrieben und gelesen werden können.  Nach einem Neustart soll der derselbe NMEA-Code gelesen werden |

### EOL

|  |  |  |
| --- | --- | --- |
| ID | Test | Erwartetes Resultat |
| TE01 | Wasser füllen | Durch das Schalten eines Ventiles über da EA-Modul wird Luft in den Wassertank gepumpt. Durch den Steigenden Druck im Tank steigt das Wasser die dafür vorgesehenen Rohre hinauf. |
| TE02 | Wasser halten | Durch das Schalten eines Ventils über das EA-Modul wird das Wasser in den Rohren gehalten, selbst wenn der Luftdruck im Wassertank nachlässt. |
| TE03 | Zu wenig / Zu viel Wasser | Ist zu wenig oder zu viel Wasser in die Rohre gepumpt worden, so soll dies vom Referenzgeber erkannt werden. Entsprechen soll auch eine Benachrichtigung an den Benutzer stattfinden. |
| TE04 | Entfernen des DUT | Entfernt man einen DUT bevor die Prüfung abgeschlossen ist, so soll dies von der EOL erkannt und die Prüfung gestoppt werden. |
| TE05 | Korrektes Darstellen der ausgelesenen Werte | Die von den Tankfüllstandsgebern ausgelesenen Werte sollen korrekt dargestellt werden. Es soll auf einen Blick erkennbar sein, ob sich der gelesene Wert im Toleranzbereich befindet. |

## Prüfplan

### Testing aller Tests

Um sicherzustellen, dass alle Tests erfolgreich sind, wurde ein Testkonzept entwickelt, damit am Ende die komplette Applikation funktioniert. Die Folgende Grafik soll das Testkonzept grafisch darstellen:



Abbildung 6 Prüfplan gesamt

##### Beschreibung

Begonnen wird mit dem ersten Testfall. Falls dieser ohne Probleme und Bugs erfüllt ist, geht man zum nächsten Testfall und testet diesen. Dies macht man so lange, bis man einen Fehler findet. In diesem Fall löst man das Problem sofort und fokussiert sich darauf, den Bug zu fixen. Sobald auch dieser Testfall erfolgreich ist, fängt man wieder mit dem ersten Testfall an und geht nochmal alle genau durch. Dies hat den Vorteil, dass falls die Änderung im Programm einen Einfluss auf früheren Test hat, man diese findet und ebenfalls beheben kann. Man macht dieses Spiel nun so lange, bis alle Bugs behoben wurden und man sagen kann das alle Tests erfolgreich erfüllt wurden.

### Einzeltest



Abbildung 7 Prüfplan einzeln

Die einzelnen Anforderungen werden nach dem Blackbox Konzept getestet. Ein Black Box Verfahren ist ein spezifikationsorientiertes Testentwurfsverfahren. Dabei werden die Testfälle ausschließlich aus der Spezifikation des zu testenden Objekts abgeleitet, ohne dabei dessen innere Struktur, den Code, zu berücksichtigen. Es wird also nur das von aussen sichtbaren Verhalten des Testobjektes beobachtet.

Bei jedem Test werden die erwarteten Resultate mit dem tatsächlichen Resultat vergleichen, Unterschiede festgehalten und gegebenenfalls Massnahmen definiert.

# Entscheiden

Alle geplanten Schritte werden entsprechend umgesetzt. Dazu gehörten:

* Soll-Zustand
* Muss-Ziele
* Kann-Ziele
* Anforderungen
* Terminplan
* Prüffälle
* Prüfplan

Die Aufgelisteten Punkte sollen so erfüllt werden, wie in der Planung beschrieben.

# Realisieren

## Kommunikation DLL

Für die Implementierung der Kommunikation wurde die PTS «TPL\_PTS\_Z10006660\_AA.pdf» herangezogen. In diesem Datenblatt sind alle Telegramme beschrieben, welche zum Anfordern eines Wertes benötigt werden. Darin enthalten sind alle digitalen und analogen Ein- und Ausgänge, sowie Geräteinformationen wie die Software Version.

Die Kommunikation zum Tankfüllstandsgeber wird nach erfolgreicher Implementierung als dll kompiliert, welche dann für die EOL verwendet werden kann.

### Implementierung

Zur Implementierung der Kommunikation über NMEA2000 mit dem Kapazitivem Tankgeber, wurde eine Vorlage herangezogen.

#### Analoge Eingänge

Aus dem Kapazitivem Tankfüllstandsgeber lassen sich drei Analoge Inputs auslesen. Diese Inputs können alle aus einem Telegramm ausgelesen werden, wobei jedoch nur zwei davon für die EOL relevant sind:

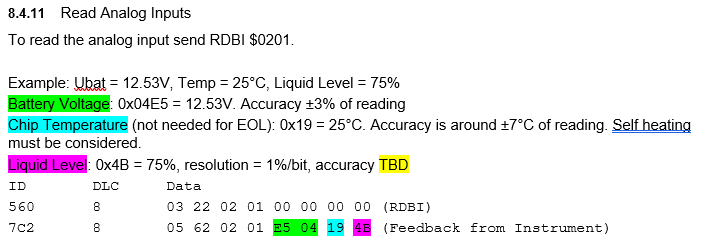


Abbildung 8 PTS Analoge Inputs

Um die Gewünschten Inputs lesen zu können, muss zuerst ein Request Telegramm über den CAN-Bus an den Tankgeber gesendet werden. Das Request Telegramm besteht aus 8 Bytes und muss wie folgt aufgebaut sein:

|  |  |  |
| --- | --- | --- |
| Byte | Wert | Beschreibung |
| [0]  [1] | [0x03]  [0x22] | Identifikationsnummer, dass ein Wert vom Tankfüllstandsgeber angefordert wird. |
| [2]  [3] | [0x02]  [0x01] | Identifikationsnummer, dass die analogen Inputs des Tankfüllstandgebers anfordert werden. |
| [4] – [7] | [0x00] | Die restlichen vier Bytes können auf dem Wert 0x00 belassen werden, da sie für das Anfordern der Daten nicht relevant sind. |

Nach einem erfolgreich gesendeten Request-Telegramm wird der Tankfüllstandsgeber mit einem Telegramm antworten, welche die angeforderten Daten enthält:

|  |  |  |
| --- | --- | --- |
| Byte | Wert | Beschreibung |
| [0]  [1] | [0x05]  [0x62] | Identifikationsnummer, dass es sich um ein Telegramm vom Tankfüllstandsgeber handelt. |
| [2]  [3] | [0x02]  [0x01] | Identifikationsnummer, dass es sich bei den folgenden Daten um die Werte der Analogen Inputs handelt. |
| [4]  [5] | Bsp: [0xE5]  Bsp: [0x04] | Wert Analoger Input: Battery Voltage |
| [6] | Bsp: [0x19] | Wert Analoger Input: Chip Temperature |
| [7] | Bsp: [0x4B] | Wert Analoger Input: Liquid Level |

#### Digitale Eingänge

Für die Digitalen Inputs eines Tankfüllstandgebers wird nur ein Telegramm benötigt. Darin enthalten sind die Werte drei verschiedener digitalen Inputs wobei für die EOL jedoch nur die DIP Switches benötigt werden.

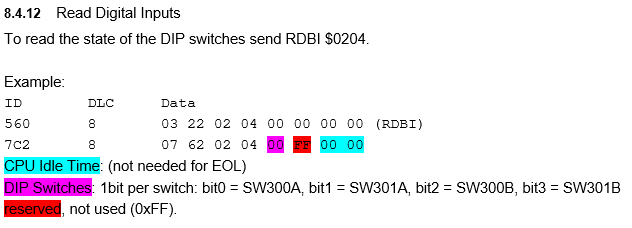


Abbildung 9 PTS Digitale Inputs

Wie auch bei den Analogen Inputs, muss, um die digitalen Inputs lesen zu können, zuerst ein Request-Telegramm an den Tankgeber gesendet werden:

|  |  |  |
| --- | --- | --- |
| Byte | Wert | Beschreibung |
| [0]  [1] | [0x03]  [0x22] | Identifikationsnummer, dass ein Wert vom Tankfüllstandsgeber angefordert wird. |
| [2]  [3] | [0x02]  [0x01] | Identifikationsnummer, dass die digitalen Inputs des Tankfüllstandgebers anfordert werden. |
| [4] – [7] | [0x00] | Die restlichen vier Bytes können auf dem Wert 0x00 belassen werden, da sie für das Anfordern der Daten nicht relevant sind. |

Nach einem erfolgreich gesendeten Request-Telegramm wird der Tankfüllstandsgeber mit einem Telegramm antworten, welche die angeforderten Daten enthält:

|  |  |  |
| --- | --- | --- |
| Byte | Wert | Beschreibung |
| [0]  [1] | [0x05]  [0x62] | Identifikationsnummer, dass es sich um ein Telegramm vom Tankfüllstandsgeber handelt. |
| [2]  [3] | [0x02]  [0x01] | Identifikationsnummer, dass es sich bei den folgenden Daten um die Werte der digitalen Inputs handelt. |
| [4] | Bsp: [0xE5] | Wert digitaler Input: DIP Switches |
| [6] | Bsp: [0x19] | Wert digitaler Input: reserved |
| [7] | Bsp: [0x4B] | Wert digitaler Input: CPU Idle Time |

#### Digitale Ausgänge

Da der kapazitive Tankgeber nur einen digitalen Output hat benötigt man lediglich ein Telegramm. Beim Digitalem Output handelt es sich um eine LED, welches sich auf der Leiterplatte eines Tankfüllstandgebers befindet. Diese LED kann mit folgendem Telegramm ein oder ausgeschalten werden:

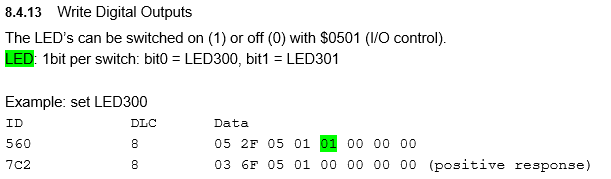


Abbildung 10 PTS Digitale Outputs

Um einen digitalen Output zu setzen, muss man lediglich ein Telegramm an den Tankgeber senden, welches dii korrekten Daten über den Zustand der Outputs enthält:

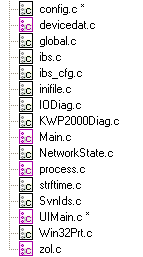
|  |  |  |
| --- | --- | --- |
| Byte | Wert | Beschreibung |
| [0]  [1] | [0x05]  [0x2F] | Identifikationsnummer, dass man ein Wert im Tankfüllstandsgeber schreiben möchte. |
| [2]  [3] | [0x05]  [0x01] | Identifikationsnummer, dass es sich bei den zu schreibenden Werten, um digitale Outputs handelt. |
| [4] | [0x00]  oder  [0x01] | Das vierte Byte wird dazu verwendet, die LED auf der Leiterplatte des Tankfüllstandgebers entweder ein- oder auszuschalten.  0x00 🡪 Ausschalten  0x01 🡪 Einschalten |

Nachdem das Telegramm mit den Werten der Zustände der digitalen Outputs an den Tankgeber gesendet wurde, erhält man eine Antwort, welche Bescheid gibt, dass das Setzen der Outputs erfolgreich war:

|  |  |  |
| --- | --- | --- |
| Byte | Wert | Beschreibung |
| [0]  [1] | [0x03]  [0x6F] | Identifikationsnummer, dass es sich um ein Telegramm vom Tankfüllstandsgeber handelt. |
| [2]  [3] | [0x05]  [0x01] | Identifikationsnummer, dass es sich beim Telegramm um die Antwort auf das Setzen der digitalen Outputs handelt. |
| [4] – [7] | [0x00] | Wenn die digitalen Outputs ohne Probleme gesetzt werden konnten, haben die Bytes 4 – 7 den Wert 0x00. |

## Tankfüllstandsgeber EOL

### Enviroment

Um die Applikation umzusetzen wird die ANSI-C-Entwicklungsumgebung CVI-Labwindows (C for Virtual Instrumentation) verwendet. Für die Programmierung der Software wird die CVI Version 7.1 auf einem Virtuellen Rechner mit dem Betriebssystem Windows XP verwendet.

Als Ursprung für die Tankfüllstandsgeber EOL Software wird die „LinkUp-EOL“ verwendet. Die grobe Dateistruktur ist aufgrund dessen bereits vorhanden und muss so nur angepasst werden.

Hier nun eine Auflistung der angepassten Dateistruktur für die Tankfüllstandsgeber EOL:

Abbildung 11 Dateistruktur

|  |  |
| --- | --- |
| Datei | Funktion |
| Config.c | In der config.c Datei werden Informationen für die Applikation gespeichert, welche während der Betriebszeit geändert werden können. So ist es möglich bestimmte Einstellung ohne neue Kompilation des Programmes zu ändern und anzupassen. |
| Devicedat.c | Die Devicedat.c ist dazu da, Informationen aus der Devicedat.ini Datei auszulesen und diese in eine Struktur zu speichern.  In der Devicedat.ini sind alle Typen von Tankfüllstandsgeber aufgelistet. Jeder Typ hatte eine Nummer und stellt eine Sektion in der Inidatei dar. Zudem sind für jeden Typen spezifische Informationen vorhanden, welche für die Prüfung wichtig sind, und nur für diesen Typen von Tankgeber zutreffen. |
| Global.c | Die gobal.c Datei ist für globale Deklarationen gedacht. In dieser Datei findet man globale Strukturen, welche über das gesamte Projekt hinweg benötigt werden. |
| Ibs.c | Mithilfe der Ibs.c Datei ist es möglich mit dem EA-Modul zu kommunizieren |
| Ibs\_cfg.c | In der Ibs\_cg.c Datei werden Konfigurationen für die Kommunikation mit dem EA-Modul vorgenommen. |
| Inifile.c | Die Inifile.c Datei bietet alle Funktionen, welche benötigt werden, um mit einer .ini Datei zu operieren. |
| IODiag.c | In der IODiag.c Datei sind alle Funktionen zur Interaktion mit dem Diagpanel deklariert. |
| KPW2000Diag.c | Mithilfe der KPW2000Diag.c Datei kann der Trace des CAN-Buses dargestellt und verfolgt werden. |
| Main.c | In der Main.c Datei befindet sich die main() Funktion womit die Applikation in dieser Datei startet. Hier werden zudem alle benötigten Initialisierungen vorgenommen. |
| NetworkState.c | Die NetworkState.c zeigt den Status der Netzwerkverbindung an. |
| Process.c | In der process.c Datei befindet sich der Ablauf der Applikation. Dies geschieht durch eine State-maschine. |
| Strftime.c | Die Strftime.c bietet eine Funktion, welche wie die Funktion sprintf() operiert. Sie gibt die Zeit und das Datum zurück. |
| SvnIds.c | Die SvnIds.c Datei wird vom SVN-Versionsverwaltungstool generiert und enthält Informationen über die Version und Autor der Software. |
| UiMain.c | In der UiMain.c Datei sind alle Funktionen, welche mit dem Userinterface interagieren deklariert. |
| Zol.c | In Zol.c Datei sind alle Ein und Ausgänge des EA-Moduls deklariert. |

### Userinterface

Das Userintferce zur EOL für die kapazitiven Tankgeber wurde nach einem Firmenstandard der veratron AG aufgebaut:

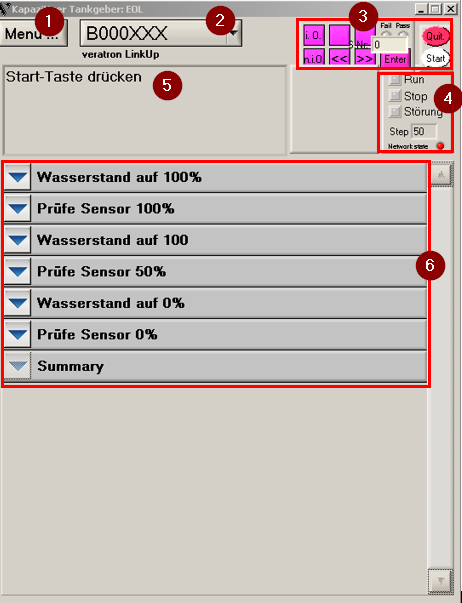


Abbildung 12 Userinterface EOL

Wie auf der Abbildung zu erkennen ist, sind sieben Prozessschritte vorhanden. Diese Schritte wurden mit einem von der veratron AG erstellen Code-Generator generiert. Dies gewährleistet, dass sich die Funktionen in verschiedenen Applikationen gleich verhalten. Die verschiedenen Prozessschritte können durch vordefinierte Dll’s eingebunden werden.

|  |  |
| --- | --- |
| Nummer | Beschreibung |
| 1 | Der Button «Menu…» öffnet bei einer Betätigung ein Menü, in welchem, von der Prüfung unabhängige Interaktionen ausgeführt werden können:  **Tests:** Beim Betätigen des «Tests …» Button wird ein neues Panel geöffnet, in welchem man vordefinierte Tests an einem DUT vornehmen kann.  **CAN:** Beim Betätigen des «CAN …» Button kann man den Trace des Can-Buses mitverfolgen.  **I/Os:** Beim Betätigen des «I/Os …» Button öffnet sich ein neues Panel, in welchem man alle Ein- und Ausgänge der Vorrichtung und des Prüflings Manuel schreiben und auslesen kann.  Abbildung 13 Menu GUI  **Programminfo:** Beim Betätigen des «Programminfo …» Button wird eine MessageBox geöffnet, welche die wichtigsten Informationen über das Programm enthält.  **Exit Program:** Beim Betätigen des «Exit Program» Button wird die Applikation geschlossen. |
| 2 | Über diesen Ring-Control lässt sich auswählen, was für ein Gerät man Prüfen möchte. Nachdem man ein Gerät ausgewählt hat, werden die entsprechenden Daten aus einer Konfigurationsdatei geladen. |
| 3 | Üblicherweise befinden sich an der Vorrichtung Druckschalter, welche für die Prüfung eines Gerätes benötigt werden. Um diese simulieren zu können, wurden diese als Buttons in dem Userinterface integriert. Sind die Druckschalter jedoch an der Vorrichtung vorhanden, so werden diese ausgeblendet. |
| 4 | In diesem Bereich gibt die Applikation Auskunft über den aktuellen Stand einer Prüfung. |
| 5 | In dem Dialogfeld werden Anweisungen, welche für die Prüfende Person wichtig sind, angezeigt. |
| 6 | In diesem Bereich sind alle Prozessschritte ersichtlich, welche für die Prüfung eines DUT’s benötigt werden. |

#### Prozessschritte

Für die Prüfung der Tankfüllstandsgeber sind sieben Prozessschritte eingebunden worden. Die Grafische Darstellung der verwendeten Prozessschritte wurde durch vordefinierte Dll’s eingebunden.

Für die Tankfüllstandsgeber EOL wurden lediglich zwei verschiedene Arten von Dll’s verwendet:

##### Workorder

Mit der Einbindung eines Workorder GUI’s durch eine Dll’s wird ein neuer Eintrag im Bereich für die Prozessschritte sichtbar.

Beim Workorder GUI ist ursprünglich dazu gedacht, eine Grafik im Userinterface Dynamisch einzubinden. Im Falle der Tankfüllstandsgeber EOL wird diese Funktion jedoch ignoriert und die Workorder Gui ist lediglich dazu da, den angezeigten Text und der Status der Prüfung verändern. Somit färbt sich der Prozessschritt orange, wenn dieser aktuell gerade ausgeführt wird, rot, wenn etwas bei der Prüfung in diesem Schritt fehlgeschlagen ist, oder grün, wenn der Prozessschritt erfolgreich durchlaufen wurde.

Für die Tankfüllstandsgeber EOL wurde der Prozessschritt «Workorder» drei Mal verwendet:

* Wasserstand auf 100%
* Wasserstand auf 50%
* Wasserstand auf 0%

Die Dll kommt neben der grafischen Darstellung auch mit den benötigten Funktionen für die Anpassung und Darstellung der Werte daher. Mit diesen Funktionen lassen sich alle Werte, welche im GUI sichtbar sind, anpassen:

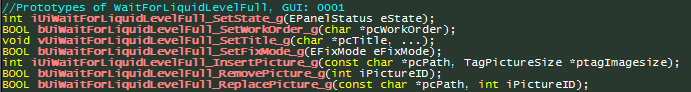


Abbildung 14 Prototypen LibDutCom Workorder

|  |  |
| --- | --- |
| Funktion | Beschreibung |
| SetState\_g() | Mit dieser Funktion lässt sich der Status des Prozessschrittes verändern. Dazu gibt man einen Wert aus dem Enumerator EPanelStatus mit:  eIdle: Nicht aktiv 🡪 grau eRunning: aktiv 🡪 orange ePass: Test erfolgreich 🡪 grün eFail: Test fehlgeschlagen 🡪 rot |

Die Restlichen Funktionen der Workorder Dll werden in dieser Dokumentation nicht beschrieben, da diese für die EOL nicht relevant sind.

##### Analog Value

Auch mit der Einbindung eines «analog value» GUI durch eine Dll’s wird ein neuer Eintrag im Bereich für die Prozessschritte sichtbar.

Bei dieser grafischen Darstellung eines analogen Wertes, lassen sich neben dem Text und der Farbe des Prozesses noch einige weitere Werte beeinflussen.

Sobald einen Prozessschritt erreicht wird, in welchem das «analog value» GUI verwendet wird, wird ein Bereich Sichtbar, welcher Elemente für die Darstellung des analogen Wertes enthält.

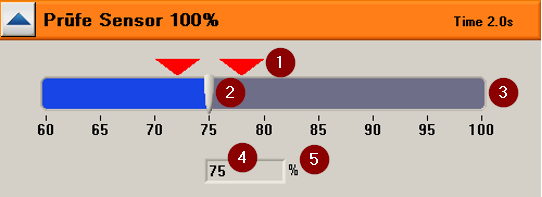


Abbildung 15 analog value LibTesterGui Dll GUI

|  |  |
| --- | --- |
| Nr. | Beschreibung |
| 1 | Die roten Dreiecke markieren die Toleranz auf der Werteskala. Damit ein Prüfling als korrekt angesehen werden kann, müssen die gemessenen Werte innerhalb dieser Toleranz sein. |
| 2 | Der Schieber auf der Werteskala markiert grafisch den aktuellen analogen Wert. Dieser sollte sich zwischen den Toleranzmarkern befinden. |
| 3 | Die Werteskala wird für die Grafische Darstellung der analogen Werte verwendet. Der minimale und maximale Wert lassen sich programmmässig verändern. |
| 4 | Hier wird der analoge Wert noch als Zahl, zur besseren Leserlichkeit dargestellt. |
| 5 | In diesem Bereich wird angegeben, um was für eine Einheit es sich beim Wert handelt. Auch dieser lässt sich programmmässig verändern. |

Die Dll kommt neben der grafischen Darstellung auch mit den benötigten Funktionen für die Anpassung und Darstellung der Werte daher. Mit diesen Funktionen lassen sich alle Werte, welche im GUI sichtbar sind, anpassen.

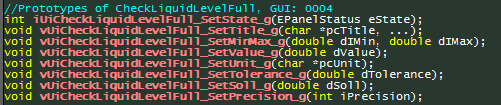


Abbildung 16 Prototypen LibDutCom analog value

|  |  |
| --- | --- |
| Funktion | Beschreibung |
| SetState\_g() | Mit dieser Funktion lässt sich der Status des Prozessschrittes verändern. Dazu gibt man einen Wert aus dem Enumerator EPanelStatus mit:  eIdle: Nicht aktiv 🡪 grau eRunning: aktiv 🡪 orange ePass: Test erfolgreich 🡪 grün eFail: Test fehlgeschlagen 🡪 rot |
| SetMinMax\_g() | Mittels der Funktion SetMinMax() lässt sich der Start- und Endpunkt der Werteskala definieren. |
| SetValue\_g() | Mit dieser Funktion lässt sich der aktuelle analoge Wert im GUI grafisch anzeigen. Dazu muss man lediglich beim Aufruf dieser Funktion den Wert als Parameter mitgeben. |
| SetUnit\_g() | Mit der SetUnit() Funktion lässt sich die angezeigte Einheit verändern. |
| SetSoll\_g() | Der Soll-Wert gibt den Optionalen Wert an, welcher der analoge Wert aufweisen sollte und lässt sich mit dieser Funktion setzen. |
| SetTolerance\_g() | Um die roten Toleranzmarker anzuzeigen wird diese Funktion verwendet. Die Toleranzmarker werden im Abstand des Mitgegebenen Wertes vom Soll-Wert platziert. |
| SetPrecision\_g() | Mit dieser Funktion kann man bestimmen, wie viele Stellen nach dem Komma angezeigt werden sollen. Dies beeinflusst sowohl für die Werteskala wie auch den im Textfeld angezeigt analogwert. |

Es wurden nur jene Funktionen beschrieben, welche schlussendlich auch in der EOL verwendet werden.

### EA-Modul

Für die Kommunikation mit der Vorrichtung ist ein Phoenix-Modul zuständig. Über dieses lassen sich alle digitalen sowie analogen Ein- und Ausgänge schreiben und auslesen. Die kommunikation mit dem Modul ist in der «ibs.c» implementiert. Die einzelnen Ein- und Ausgänge lassen sich jedoch in der «zol.c» Datei finden. Das Phoenix-Modul selbst wird über ein RJ-45 Kabel angeschlossen.

#### Digitale Eingäge

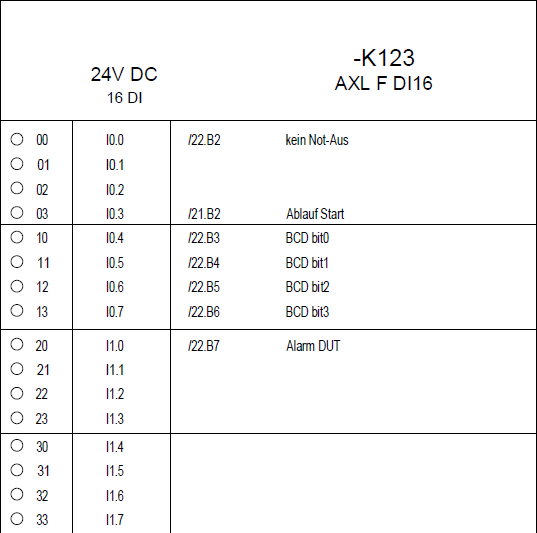


Abbildung 17 EA-Modul Schema DI

|  |  |  |
| --- | --- | --- |
| Input Nr | Name | Beschreibung |
| 0.0 | Kein Not-Aus | Notaus Taste ist quittiert. Der Ablauf kann |
| 0.3 | Ablauf Start | Wedel Schalter an der Vorrichtung, welcher von der Prüfenden Person benutzt wird, um den Ablauf zu starten und nach dem Justieren der Tankfüllstandsgeber im Ablauf weiterzufahren. |
| 0.4, 0.5, 0.6, 0.7 | BCD Bit 0 - 3 | Die digitalen Inputs 0.4 – 0.7 ergeben zusammen eine dualkodierte Dezimalziffer, welche Auskunft darüber gibt, von welchem Tankfüllstandsgeber aktuell die Werte ausgelesen werden. |
| 1.0 | Alarm DUT | Gibt Auskunft ob ein Alarmsignal vom DUT ausgegangen ist.  Ein Alarmsignal muss unter folgenden Kriterien erfolgen: Fresh-Water Tankgeber: Wasserstand < 25%  Waste-Water Tankgeber: Wasserstand > 75% |

Die digitalen Eingänge wurden wie folgt implementiert:

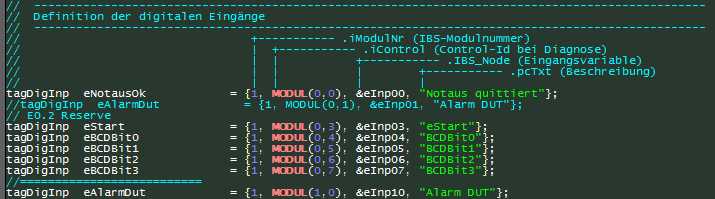


Abbildung 18 EA-Modul Implementierung DI

#### Digitale Ausgänge

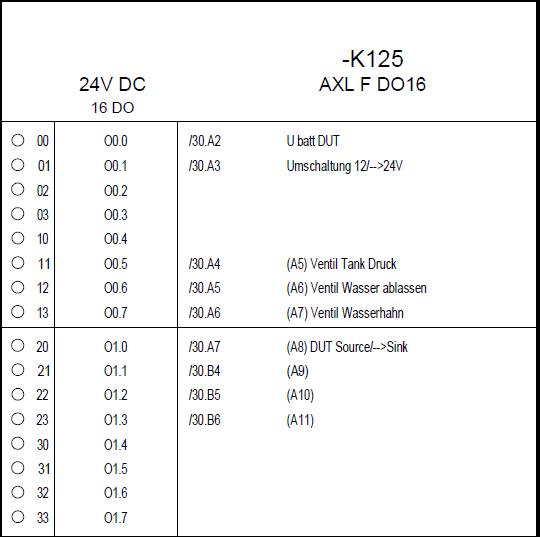


Abbildung 19 EA-Modul Schema DO

|  |  |  |
| --- | --- | --- |
| Input Nr | Name | Beschreibung |
| 0.0 | U bat DUT | Stromaufnahme des DUT’s |
| 0.1 | Umschaltung 12/24V | Stromaufnahme des DUT’s von 12 auf 24 Volt umschalten TRUE 🡪 12 V FALSE 🡪 24 V |
| 0.5 | Ventil Tank Druck | Druck im Wassertank aufbauen, damit Wasserspiegel steigt |
| 0.6 | Ventil Wasser ablassen | Wasserspiegel senken |
| 0.7 | Ventil Wasserhahn | Wasser in Position halten |
| 1.0 | DUT Source/Sink | Umschalten des einlesens der Werte von Source- zu Sinkmode TRUE 🡪 Sourcemode FALSE 🡪Sinkmode |

Die digitalen Ausgänge wurden wie folgt implementiert:

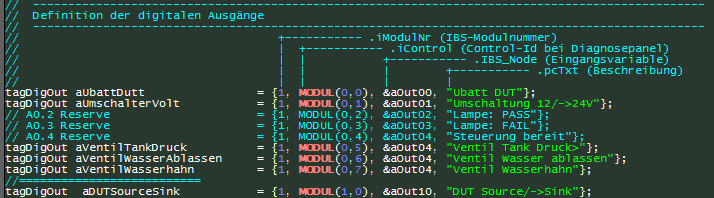


Abbildung 20 EA-Modul Implementierung DO

Um den Wasserspiegel zu erhöhen, senken oder ihn zu halten werden die Outputs 0.5 – 0.7 benötigt. Die Outputs müssen für die gewünschte Tätigkeit folgendermassen gesetzt werden:

|  |  |  |  |
| --- | --- | --- | --- |
| Aktion | 0.5 Ventil Tank Druck | 0.6 Ventil Wasser ablassen | 0.7 Ventil Wasserhahn |
| Wasserstand erhöhen | TRUE | FALSE | TRUE |
| Wasserstand senken | FALSE | TRUE | TRUE |
| Wasserstand halten | FALSE | FALSE | FALSE |

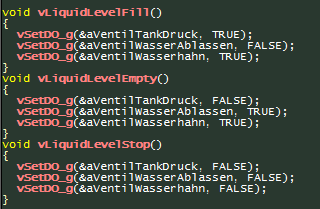
Für diese Aktionen wurden jeweils eigene Funktionen implementiert, welche diese Logik wiederspiegeln:

Abbildung 21 Wasserstand Funktionen

#### Analoge Eingänge

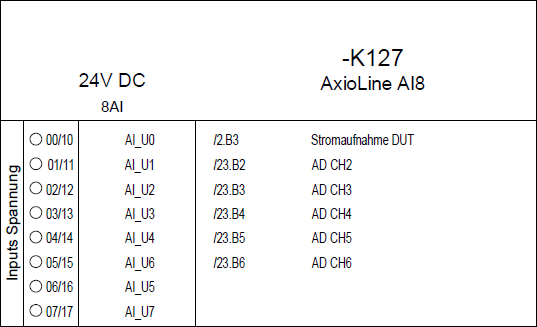


Abbildung 22 EA-Modul Schema AI

|  |  |  |
| --- | --- | --- |
| Input Nr | Name | Beschreibung |
| 0.0 | Stromaufnahme DUT | Stromaufnahme des Prüflings |
| 0.1 | AD CH2 | Die analogen Input chanels 2 – 6 werden dazu benötigt, den Wasserstand Wert der DUT’s oder des Referenzgerätes zu lesen.  Um ein Wasserstandswert zu lesen werden immer zwei Channels verwendet. Je nach Typ des Tankgebers werden jedoch Unterschiedliche Chanels genutzt:   |  |  | | --- | --- | | Zweipoligen Stecker | Channel 2 + 3 | | Vierpoligen Stecker | Sourcemode: Chanel 4 + 6  Sinkmode: Chanel 2 + 4 | | Referenz Gerät | Chanel 5 + 6 | |
| 0.2 | AD CH3 |
| 0.3 | AD CH4 |
| 0.4 | AD CH5 |
| 0.6 | AD CH6 |

Die analogen Eingänge wurden wie folgt implementiert:

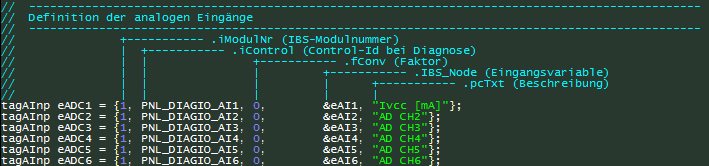


Abbildung 23 EA-Modul Implementierung AI

### State Maschine

Für die Implementierung des Ablaufs, wird die C-Datei «process.c» verwendet. Der Ablauf wird in einer State Maschine programmiert, wofür eine switch-Anweisung verwendet wird. Die einzelnen Fälle der Switch-Anweisung entsprechen jeweils einer Aufgabe im Ablauf der Prüfung.

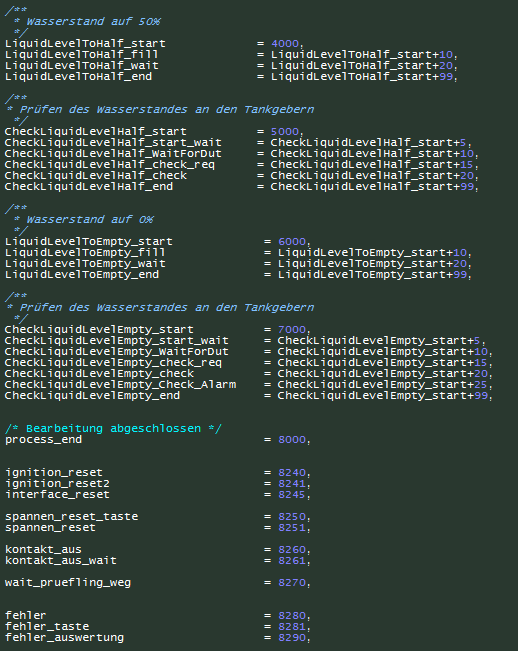
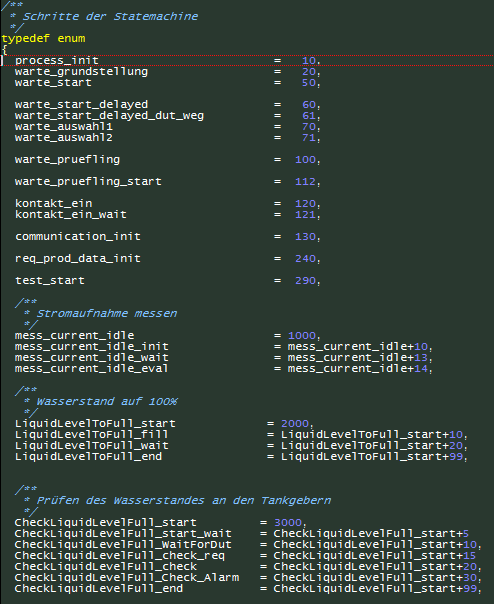
Die Fälle der Switch-Anweisungen werden vor der Implementation der Logik mithilfe eines Enumerators erstellt:

Abbildung 24 Statemaschine Fälle 1

Abbildung 25 Statemaschine Fälle 2



Die Switch Anweisung wird zyklisch durch einen Scheduler alle 100 Millisekunden aufgerufen.

Durch eine Statische Variable, welche zu Beginn mit dem State «process:\_init» initialisiert wird, ist es nun möglich von einem State zum nächsten zu gelangen. Dazu setzt man lediglich nach dem durchlauf der implementierten Logik eines Falles den Wert der Statusvariable auf den nächsten Fall, welcher durchlaufen werden soll. Beim nächsten Durchlaufen der Switch-Anweisung wird dann der Implantierte Code des neuen States ausgeführt.

### Lesen des Wasser Standes an DUT

Beim Lesen des Wasserstandes muss auf folgende Punkte Rücksicht genommen werden:

* Signaltyp des Tankfüllstandgebers
* Sink- oder Source-Mode
* Anzahl Pole am Stecker

Im ersten Schritt wird Unterschieden, ob es sich beim DUT um ein Analogen oder einen Digitalen Tankgeber handelt.

Handelt es sich um einen digitalen Tankfüllstandsgeber, so muss ein Request-Telegramm über den Can-Bus gesendet werden. Der Tankgeber antwortet nun ebenfalls über den Can-Bus und sendet den aktuellen Wasserstand.

Handelt es sich jedoch um einen analogen Tankfüllstandsgeber, so wird der Wasserstand nicht über den Can-Bus, sondern über die Analogen Inputs des EA-Moduls ausgelesen.

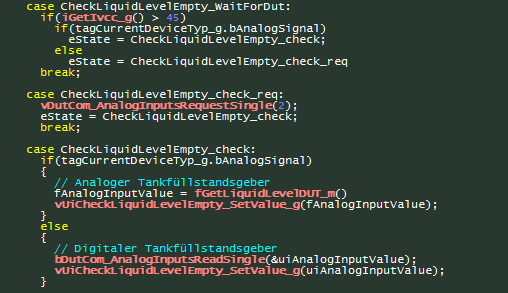


Abbildung 26 Read Liquid Level, Signaltype

Für die digitalen Tankfüllstandsgeber müssen keine weiteren Unterscheidungen gemacht werden. Das Signal, welches vom Can-Bus ausgelesen wurde, kann für die Prüfung ohne weiteres verwendet werden.

Bei den analogen Tankfüllstandsgeber muss nun im nächsten Schritt unterschieden werden, ob es sich beim Tankfüllstandsgeber um einen Zwei- oder Vierpoligen Stecker handelt. Besitzt der aktuell angeschlossene Prüfling ein Zweipoligen Stecker, so kann der Wasserstand mithilfe der Analogen Inputs des EA-Modules ausgelesen werden. Dies geschieht durch die Inputs «AD CH2» und «AD CH3».

Handelt es sich jedoch um einen Vierpoligen analogen Tankfüllstandsgeber muss auf ein weiteres Kriterium Rücksicht genommen werden. Es wird unterschieden, ob es sich um ein Sink-Mode-Tankgeber oder ein Source-Mode-Tankgeber handelt. Im Falle eines Sink-Mode-Tankgebers werden die analogen Inputs «AD CH 4» und «AD CH 6» benötigt, um den Wasserstand einzulesen. Ist der aktuelle DUT jedoch ein Source-Mode-Tankgeber müssen die analogen Inputs «AD CH2» und «AD CH4» verwendet werden.

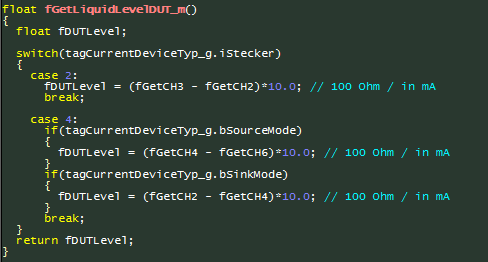


Abbildung 27 Read Liquid Level, Sink/Source

### Alarm Test

Ein Tankfüllstandsgeber muss ab einem bestimmten Wasserspiegel ein Alarmsignal ausgeben. Dieser bestimmte Wasserspiegel hängt von der Wasserart des Tankfüllstandgebers ab. Es gibt zwei verschiedene Wasserarten:

#### Waste-Water-Tankfüllstandsgeber:

Ein Waste-Water-Tankfüllstandsgeber wird in einem Tank eingesetzt, in welchem das verbrauchte Wasser landet. Dieser Tank darf nicht überlaufen, weshalb ein Alarm ausgegeben werden soll, sobald der Wasserstand höher als 75 % des Tankes ist.

Im Ablauf der EOL wird dies direkt nach dem Justieren der Prüflinge auf 100 % geprüft, wenn der Wasserspiegel über 75 % ist.

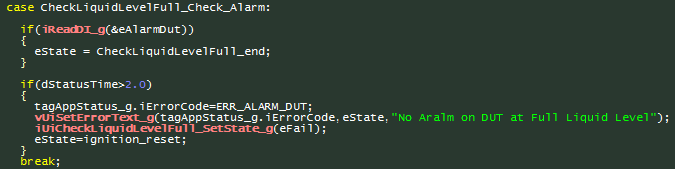


Abbildung 28 Waste-Water Alarm Prüfung

In der State-Maschine handelt der Fall «CheckLiquidLevelFull\_Check\_Alarm» die Prüfung des Alarms des DUT’s ab.

Sobald die State-Maschine in diesen Fall kommt, wird der digitale Input des Alarmsignals überprüft. Wenn nach zwei Sekunden kein Alarmsignal gefunden werden konnte, wird der Prüfling als Fehlerhaft angesehen und die Prüfung beendet.

#### Fresh-Water-Tankfüllstandsgeber:

Ein Fresh-Water-Tankfüllstandsgeber wird in einem Tank eingesetzt, in welchem das frische, zu verbrauchende Wasser ist. Dieser Tank darf nicht leer sein, weshalb ein Alarm ausgegeben werden soll, sobald der Wasserstand tiefer als 25 % es Tankes ist.

Im Ablauf der EOL wird dies direkt nach dem Prüfen der Prüflinge auf 0 % geprüft, wenn der Wasserspiegel unter 25 % ist.

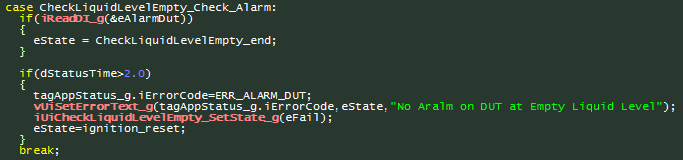


Abbildung 29 Fresh Water Alarm Prüfung

In der State-Maschine handelt der Fall «CheckLiquidLevelEmpty\_Check\_Alarm» die Prüfung des Alarms des DUT’s ab.

Sobald die State-Maschine in diesen Fall kommt, wird der digitale Input des Alarmsignals überprüft. Wenn nach zwei Sekunden kein Alarmsignal gefunden werden konnte, wird der Prüfling als Fehlerhaft angesehen und die Prüfung beendet.

# Kontrollieren

## Prüffälle Kommunikation

Beim Prüfen wird strikt nach dem definierten Prüfplan vorgegangen und es werden die im Voraus definierten Testfälle abgehandelt.

Für alle Tests wird, die von der veratron AG entwickelten Software «Test\_LibDutComm\_XXX» Software verwendet, da diese dafür entwickelt wurde, CAN-Kommunikations Dll’s zu testen.

### TK01 - Lesen der Analogen Inputs

**Beschreibung**

Aus dem Kapazitivem Tankfüllstandsgeber lassen sich drei Analoge Inputs auslesen. Diese sollen korrekt aus dem CAN-Telegramm gelesen werden können.

**Erwartetes Ergebnis**

Die Analogen Inputs können korrekt vom Tankfüllstandsgeber eingelesen werden.

Die Analogen Inputs können sowohl einzeln wie auch alle zusammen eingelesen werden

**Test**

|  |  |  |
| --- | --- | --- |
| Test | Ergebnis | Symbol |
| Lesen der Batterie Voltage | Der Analoge Wert der Batterie Voltage kann korrekt ausgelesen werden. | Häkchen |
| Lesen der Chip Temperatur | Der Analoge Wert der Chip Temperatur kann korrekt ausgelesen werden. | Häkchen |
| Lesen des Wasserstandes | Der Analoge Wert der Chip Temperatur kann korrekt ausgelesen werden. | Häkchen |
| Lesen aller analogen Inputs | Beim Anfordern aller analogen Inputs, werden alle Werte korrekt ausgelesen. | Häkchen |

**Auswertung / Massnahmen**

Da alle Tests Erfolgreich waren, sind keine Massnahmen nötig.

### TK02 - Lesen der Digitalen Inputs

**Beschreibung**

Aus dem Kapazitivem Tankfüllstandsgeber lassen sich drei Digitale Inputs auslesen. Da für die EOL jedoch nur ein Digitaler Inputs benötigt wird, werden nur die BCD Switches getestet.

Die BCD Switches könne vier verschiedene Stellungen einnehmen. Es werden alle vier Stellungen geprüft.

**Erwartetes Ergebnis**

Die Digitalen Inputs können korrekt vom Tankfüllstandsgeber eingelesen werden.

Die Digitalen Inputs können sowohl einzeln wie auch alle zusammen ausgelesen werden

**Test**

|  |  |  |
| --- | --- | --- |
| Test | Ergebnis | Symbol |
| Lesen des BCD Switch Stellung 1 | Der Digitale Wert der ersten BCD Stellung kann korrekt ausgelesen werden. | Häkchen |
| Lesen des BCD Switch Stellung 2 | Der Digitale Wert der zweiten BCD Stellung kann korrekt ausgelesen werden. | Häkchen |
| Lesen des BCD Switch Stellung 3 | Der Digitale Wert der dritten BCD Stellung kann korrekt ausgelesen werden. | Häkchen |
| Lesen des BCD Switch Stellung 4 | Der Digitale Wert der vierten BCD Stellung kann korrekt ausgelesen werden. | Häkchen |
| Lesen aller digitalen Inputs | Beim Anfordern aller digitalen Inputs, werden alle Werte korrekt ausgelesen. | Häkchen |

**Auswertung / Massnahmen**

Da alle Tests Erfolgreich waren, sind keine Massnahmen nötig.

### TK03 - Schreiben der Digitalen Outputs

**Beschreibung**

Der kapazitive Tankgeber besitzt lediglich einen digitalen Output. Es handelt sich um eine LED, welches sich auf der Leiterplatte eines Tankfüllstandgebers befindet

**Erwartetes Ergebnis**

Die Digitalen Inputs können korrekt vom Tankfüllstandsgeber eingelesen werden.

**Test**

|  |  |  |
| --- | --- | --- |
| Test | Ergebnis | Symbol |
| LED einschalten | Der Digitale Wert der ersten BCD Stellung kann korrekt ausgelesen werden. | Häkchen |
| LED auschlaten | Der Digitale Wert der zweiten BCD Stellung kann korrekt ausgelesen werden. | Häkchen |

**Auswertung / Massnahmen**

Da alle Tests Erfolgreich waren, sind keine Massnahmen nötig.

### TK04 - Lesen Software Version

**Beschreibung**

Der Kapazitive Tankfüllstandsgeber enthält Informationen über Software Version, welche durch ein CAN-Telegramm angefordert werden können.

**Erwartetes Ergebnis**

Die Software Version soll korrekt eingelesen werden können.

**Test**

|  |  |  |
| --- | --- | --- |
| Test | Ergebnis | Symbol |
| Lesen der Software Version | Die LED kann korrekt eingeschalten werden | Häkchen |

**Auswertung / Massnahmen**

Da alle Tests Erfolgreich waren, sind keine Massnahmen nötig.

### TK05 - Schreiben/Lesen der B-Nummer

**Beschreibung**

Die B-Nummer dient dazu, den Tankfüllstandsgeber eindeutig zu identifizieren. Jeder Tankfüllstandsgeber besitzt eine eigene B-Nummer, welche sowohl geschrieben als auch ausgelesen werden kann.

**Erwartetes Ergebnis**

Die B-Nummer soll korrekt geschrieben und gelesen werden können.

Nach einem Neustart soll der Tankfüllstandsgeber dieselbe B-Nummer haben

**Test**

|  |  |  |
| --- | --- | --- |
| Test | Ergebnis | Symbol |
| B-Nummer lesen | Die B-Nummer kann korrekt gelesen werden | Häkchen |
| B-Nummer schreiben | Die B-Nummer kann korrekt geschrieben werden | Häkchen |
| B-Nummer nach Neustart lesen | Auch nach einem Neustart, bleibt die neu geschriebene B-Nummer bestehen | Häkchen |

**Auswertung / Massnahmen**

Da alle Tests Erfolgreich waren, sind keine Massnahmen nötig.

### TK05 - Schreiben/Lesen deS NMEA-Codes

**Beschreibung**

NMEA ist ein Standardprotokoll für die Kommunikation zwischen Geräten, welches im Marine Bereich eingesetzt wird. Man kann einen sogenannten NMEA-Code anfordern, womit jedes Unternehmen, welches mit NMEA2000 arbeitet, einen eigenen Code besitzt. Dieser Code muss eingelesen und geschrieben werden können.

**Erwartetes Ergebnis**

Der NMEA-Code soll korrekt geschrieben und gelesen werden können.

Nach einem Neustart soll der derselbe NMEA-Code gelesen werden

**Test**

|  |  |  |
| --- | --- | --- |
| Test | Ergebnis | Symbol |
| NMEA-Code lesen | Der NMEA-Code kann korrekt gelesen werden | Häkchen |
| NMEA-Code schreiben | Der NMEA-Code kann korrekt geschrieben werden | Häkchen |
| NMEA-Code nach Neustart lesen | Auch nach einem Neustart, bleibt der neu geschriebene NMEA-Code bestehen | Häkchen |

**Auswertung / Massnahmen**

Da alle Tests Erfolgreich waren, sind keine Massnahmen nötig.

## Prüfprotokoll Kommunikation

Im Prüfprotokoll sind alle Tätigkeiten, welche zum Testen der Kommunikation gemacht wurden, in der korrekten Reihenfolge beschrieben.

|  |  |  |
| --- | --- | --- |
| Nr. | Tätigkeit | Bemerkungen |
| 1 | Testen der Analogen Inputs | Alle Tests sind erfolgreich |
| 2 | Testen der Digitalen Inputs | Alle Tests sind erfolgreich |
| 3 | Testen der Digitalen Outputs | Alle Tests sind erfolgreich |
| 4 | Testen der Software Version | Alle Tests sind erfolgreich |
| 5 | Testen der B-Nummer | Alle Tests sind erfolgreich |
| 6 | Testen des NMEA-Codes | Alle Tests sind erfolgreich |

## Prüfffälle Tankfüllstandsgeber EOL

m

## Soll-Ist Zeitvergleich

|  |  |  |  |
| --- | --- | --- | --- |
| Arbeit | Soll Zeit [h] | Ist Zeit [h] | Differenz [h] |
| Informieren |  |  |  |
| * Analyse des Projektauftrages | 0.5 | 1 | 0.5 |
| * Ist Stand Beschreiben | 0.5 | 0.5 | 0 |
| * Soll Stand beschreiben | 0.25 | 0.25 | 0 |
| * Muss- und Kann-Ziele beschreiben und definieren | 0.25 | 0.25 | 0 |
| Planen | 1.5 |  |  |
| * Terminplan Grob | 0.25 | 0.25 | 0 |
| * Terminplan detailliert | 1 | 1.5 | 0.5 |
| * Meilensteinplan | 0.5 | 0.5 | 0 |
| * Ablauf der EOL | 1.5 | 1 | -0.5 |
| * Definition de Testfälle | 1 | 1 | 0 |
| * Definition Prüfplan | 0.5 | 0.5 | 0 |
| Entscheiden |  |  |  |
| * Allfällige Entscheidungen | 1 | 0.25 | -0.75 |
| Realisieren |  |  |  |
| * Implementation der Kommunikation   + Analoge Inputs   + Digitale Inputs   + Digitale Outputs | 10 | 8 | -2 |
| * Implementation der EOL   + Kommunikation mit EA-Modul   + Kommunikation mit DUT   + Prüfungsablauf | 38 |  |  |
| Kontrollieren |  |  |  |
| * Prüfen + Massnahmen Kommunikation | 4 | 2 | -2 |
| * Prüfen + Massnahmen EOL | 4 |  |  |
| * Soll- Ist Stand überprüfen (Zeit nach Terminplan) | 2 |  |  |
| Auswerten |  |  |  |
| * Reflexion der Arbeit | 4 |  |  |
| Dokumentation | 20 |  |  |
| Total: | 89.25 |  |  |

# Auswerten

## Reflexion

### Allgemein

### Gelerntes

#### Dokumentation

#### Funktionen

#### Reibungspunkte

# Abbildungsverzeichnis

[Abbildung 1 kapazitiver Tankfüllstandsgeber 5](file:///E:\02_Arbeit\Veratron\IPA\05_Doku\DokumentationIPA.docx#_Toc37247857)

[Abbildung 2 Vorrichtung EOL kapazitiver Tankfüllstandsgeber 5](file:///E:\02_Arbeit\Veratron\IPA\05_Doku\DokumentationIPA.docx#_Toc37247858)

[Abbildung 3 Terminplan Text 7](#_Toc37247859)

[Abbildung 4 Terminplan Grafisch 7](#_Toc37247860)

[Abbildung 5 Meilenstein Planung 9](#_Toc37247861)

[Abbildung 6 Prüfplan gesamt 13](#_Toc37247862)

[Abbildung 7 Prüfplan einzeln 13](#_Toc37247863)

[Abbildung 8 PTS Analoge Inputs 14](#_Toc37247864)

[Abbildung 9 PTS Digitale Inputs 15](#_Toc37247865)

[Abbildung 10 PTS Digitale Outputs 16](#_Toc37247866)

[Abbildung 11 Dateistruktur 18](file:///E:\02_Arbeit\Veratron\IPA\05_Doku\DokumentationIPA.docx#_Toc37247867)

[Abbildung 12 Userinterface EOL 20](#_Toc37247868)

[Abbildung 13 Menu GUI 21](file:///E:\02_Arbeit\Veratron\IPA\05_Doku\DokumentationIPA.docx#_Toc37247869)

[Abbildung 14 EA-Modul Schema DI 23](#_Toc37247870)

[Abbildung 15 EA-Modul Implementierung DI 24](#_Toc37247871)

[Abbildung 16 EA-Modul Schema DO 24](#_Toc37247872)

[Abbildung 17 EA-Modul Implementierung DO 25](#_Toc37247873)

[Abbildung 18 Wasserstand Funktionen 25](file:///E:\02_Arbeit\Veratron\IPA\05_Doku\DokumentationIPA.docx#_Toc37247874)

[Abbildung 19 EA-Modul Schema AI 26](#_Toc37247875)

[Abbildung 20 EA-Modul Implementierung AI 26](#_Toc37247876)

[Abbildung 21 Statemaschine Fälle 1 27](file:///E:\02_Arbeit\Veratron\IPA\05_Doku\DokumentationIPA.docx#_Toc37247877)

[Abbildung 22 Statemaschine Fälle 2 27](file:///E:\02_Arbeit\Veratron\IPA\05_Doku\DokumentationIPA.docx#_Toc37247878)

[Abbildung 23 Read Liquid Level, Signaltype 28](#_Toc37247879)

[Abbildung 24 Read Liquid Level, Sink/Source 29](#_Toc37247880)

[Abbildung 25 Waste-Water Alarm Prüfung 29](#_Toc37247881)

[Abbildung 26 Fresh Water Alarm Prüfung 30](#_Toc37247882)