

Titel: **Individuelle praktische Arbeit**

Thema: **Datenaufbereitung und Bedienoberfläche für
Testautomationstool TsNet**

Dieses Dokument ist nach der „Dokumentations-Empfehlung für die IPA Informatik“ des Kantons Zug, Stand 06.01.2017, aufgebaut und enthält den schriftlichen Teil der „Individuellen Praktischen Arbeit“ von Wanda Lao. Es dient zur Orientierung der Experten und des Fach-Vorgesetzten.

Der erste Teil befasst sich mit der Aufgabenstellung, der Planung und dem Arbeitsjournal. Der zweite Teil beinhaltet die Charakteristik der eigentlichen Arbeit.

Key Words: IPA, TsNet, VBA

Dokument Kategorie:	ProjectRecord
Gültigkeitsdatum:	2017-04-10
Dokument Status:	Abgeschlossen
Autor:	Wanda Lao
Abteilung:	BT CPS R&D ZG CS SAP
Verantwortliche Stelle:	wanda.lao@siemens.com
Firma:	Siemens Schweiz AG, Building Technologies Division Control Products & Systems
Basierend auf Vorlage:	Workbook_Standard; 4; 2014-11-05; Donat Hutter,3531

Änderungsgeschichte

Rev.	Datum	Autor	Änderungen
0.1	24-März-2017	Wanda Lao	Status = In Bearbeitung <ul style="list-style-type: none"> - Journal Tag 1 - Strukturaufbau - Tätigkeiten und Meilensteine
0.2	27-März-2017	Wanda Lao	Status = In Bearbeitung <ul style="list-style-type: none"> - Journal Tag 2 - Einführung - Projektauftrag gemäss PkOrg
0.3	28-März-2017	Wanda Lao	Status = In Bearbeitung <ul style="list-style-type: none"> - Journal Tag 3 - Projektorganisation
0.4	30-März-2017	Wanda Lao	Status = In Bearbeitung <ul style="list-style-type: none"> - Journal Tag 4
0.5	31-März-2017	Wanda Lao	Status = In Bearbeitung <ul style="list-style-type: none"> - Journal Tag 5
0.6	03- April-2017	Wanda Lao	Status = In Bearbeitung <ul style="list-style-type: none"> - Journal Tag 6 - IPA Kurzfassung
0.7	04-April-2017	Wanda Lao	Status = In Bearbeitung <ul style="list-style-type: none"> - Journal Tag 7 - Realisierung
0.8	06- April-2017	Wanda Lao	Status = In Bearbeitung <ul style="list-style-type: none"> - Journal Tag 8 - Realisierung
0.9	07- April- 2017	Wanda Lao	Status = In Bearbeitung <ul style="list-style-type: none"> - Journal Tag 9 - Einfügen von <ul style="list-style-type: none"> - Soll-Zeitplan - White-Box-Test - Struktogrammen
1.0	10- April- 2017	Wanda Lao	Status = In Bearbeitung <ul style="list-style-type: none"> - Journal Tag 10 - Einfügen von <ul style="list-style-type: none"> - Ist-Zeitplan - Akzeptanztest - Glossar - Auswertung - Tabellen und Abbildungen bezeichnet

Inhaltsverzeichnis

1.	TEIL 1	5
1.1	EINFÜHRUNG	5
1.1.1	IPA	5
1.1.2	Zweck des Dokumentes	5
1.1.3	Ziel Publikum	5
1.2	PROJEKTAUFTRAG GEMÄSS PKORG	6
1.2.1	Ausgangslage	6
1.2.2	Detaillierte Aufgabenstellung	7
1.2.3	Mittel und Methoden	7
1.2.4	Vorkenntnisse	8
1.2.5	Vorarbeiten	8
1.2.6	Neue Lerninhalte	8
1.2.7	Arbeiten in den letzten 6 Monaten	8
1.3	PROJEKTORGANISATION	9
1.3.1	Datensicherung	9
1.3.2	Beteiligte Dienste, Fachabteilungen und Personen	9
1.3.3	Abteilung BT CPS R&D ZG CS SAP	10
1.3.4	Verwendete Projektmanagementmethode	10
1.3.5	Arbeitsplatz	11
1.3.6	Ordnerstruktur auf Server	11
1.3.7	Risikobeschreibung	12
1.4	PLANUNG	13
1.4.1	Zeitplan: Soll	14
1.4.2	Zeitplan: Soll-Ist-Vergleich	15
1.4.3	Tätigkeiten	16
1.4.4	Meilensteine	18
1.5	ARBEITSJOURNAL	19
1.5.1	Zweck des Arbeitsjournals	19
1.5.2	Anwendungsbereich, Abgrenzung	19
1.5.3	Aufbau	19
1.5.4	Arbeitsjournale vom 24.03.2017 bis 10.04.2017	20
2.	TEIL 2	30
2.1	IPA KURZFASSUNG	30
2.1.1	Ausgangssituation	30
2.1.2	Umsetzung	30
2.1.3	Ergebnis	30
2.2	REALISIERUNG	31
2.2.1	Bedienoberfläche	31
2.2.2	Informationen zur Bedienoberfläche	34
2.2.3	Entwicklungsumgebung VBA (Editor)	36
2.2.4	Struktogramme	37
2.2.5	Beschreibungen der Funktionen ohne Struktogramm	43
2.2.6	Implementierung	44
2.3	KONTROLLE	49
2.3.1	Testkonzept	49
2.3.2	White-Box-Test	52
2.3.3	Akzeptanz-Test	61
2.4	AUSWERTUNG	74
2.4.1	Schlusswort	74
2.5	GLOSSAR	75
2.6	QUELLEN	76
2.7	ANHANG	77
2.7.1	Code	78

Tabellenverzeichnis:

Tabelle 1 : Projektauftrag [1].....	6
Tabelle 2: Risikobeschreibung.....	12
Tabelle 3: Zeitplan – Teile	13
Tabelle 4: Zeitplan – Legende	13
Tabelle 5: Tätigkeiten	18
Tabelle 6: Meilensteine.....	18
Tabelle 7: Funktionen.....	34
Tabelle 8: Definierte Felder.....	35
Tabelle 9: Erklärung zum Struktogramm.....	38
Tabelle 10: Funktionsbeschreibung [ohne Struktogramm].....	43
Tabelle 11: Glossar	76
Tabelle 12: Quellenverzeichnis.....	76

Abbildungsverzeichnis:

Abbildung 1: Siemens Server	9
Abbildung 2: Lokaler Backup	9
Abbildung 3: IPERKA [2].....	10
Abbildung 4: Arbeitsumgebung.....	11
Abbildung 5: Ordnerstruktur.....	11
Abbildung 6: Zeitplan - Soll.....	14
Abbildung 7: Zeitplan – Ist	15
Abbildung 8: Bedienoberfläche	31
Abbildung 9: Inputfeld.....	31
Abbildung 10: Funktionsfeld Teil 1	32
Abbildung 11: Funktionsfeld Teil 2	32
Abbildung 12: Funktionsfeld Teil 3	32
Abbildung 13: Funktionsfeld Teil 4	32
Abbildung 14: Funktionsfeld Teil 5	32
Abbildung 15: Funktionsfeld Teil 6	33
Abbildung 16: Funktionsfeld Teil 6.1	33
Abbildung 17: Funktionsfeld Teil 6.2	33
Abbildung 18: Funktionsfeld Teil 6.3	33
Abbildung 19: VBA-Editor	36
Abbildung 20: Struktogramm AppendFromEDE	39
Abbildung 21: Struktogramm ObjToAlias	39
Abbildung 22: Struktogramm AliasToObj	40
Abbildung 23: Struktogramm BACnetDataEDE	40
Abbildung 24: Struktogramm CheckObjects.....	41
Abbildung 25: Struktogramm CheckAlias.....	42
Abbildung 26: AppendFromEDE Button.....	44
Abbildung 27: AppendFromEDE User Form.....	44
Abbildung 28: AppendFromEDE vorher	44
Abbildung 29: AppendFromEDE nachher	44
Abbildung 30: ObjToAlias Button	45
Abbildung 31: ObjToAlias User Form.....	45
Abbildung 32: ObjToAlias Ergebnis.....	45
Abbildung 33: AliasToObj Button	46
Abbildung 34: AliasToObj User Form.....	46
Abbildung 35: AliasToObj Ergebnis.....	46
Abbildung 36: BACnet Data Button.....	47
Abbildung 37: BACnet Data User Form.....	47
Abbildung 38: BACnet Data Löschen.....	47
Abbildung 39: BACnet Data Hinzufügen	47
Abbildung 40: Testumgebung.....	49
Abbildung 41: Beispiel – nicht Eigenanteil.....	77

1. Teil 1

1.1 Einführung

1.1.1 IPA

IPA steht für **i**ndividuelle **p**raktische **A**rbeit und wird von allen Informatik–Lernenden im letzten Semester der beruflichen Grundbildung durchgeführt. Die Plattform PkOrg ist zuständig für einen organisatorischen und einwandfreien Ablauf der IPA.

1.1.2 Zweck des Dokumentes

Der IPA-Bericht beinhaltet alle Arbeitsschritte, welche im Rahmen der IPA von Wanda Lao durchgeführt wurden. Das Dokument beinhaltet zwei Teile. Im ersten Teil befinden sich die Aufgabestellung, die Planung und das Arbeitsjournal der ganzen IPA.

Der zweite Teil umfasst die Beschreibung der eigentlichen Arbeit.

1.1.3 Ziel Publikum

Der Inhalt richtet sich in erster Linie an die Experten und den Fachvorgesetzten der IPA. Dadurch kann die Arbeit nachvollzogen und beurteilt werden.

Des Weiteren können Entwickler von TsNet diese Dokumentation verwenden, um sich zu informieren.

1.2 Projektauftrag gemäss PkOrg

Projekttitel	Datenaufbereitung und Bedienoberfläche für Testautomationstool TsNet
Prüfungskandidat	Wanda Lao
Fachvorgesetzter	Michael Speckien
Auftraggeber	Michael Speckien

Tabelle 1 : Projektauftrag [1]

1.2.1 Ausgangslage

TsNet wird seit geraumer Zeit zur Testspezifikation und Testdurchführung innerhalb der Siemens Gebäudetechnik genutzt. Der Bereich Applications hat ein Excel-Sheet zur Spezifikation von Testschritten erstellt.

Davon gibt es eine aktuelle Version, jedoch muss diese wegen

- funktionalen Erweiterung,
- schlechter Wartbarkeit,
- alten Excel- und Windows-Versionen komplett neu erstellt werden.

Das Projekt TsNet V2 besteht aus

1. einem Definitionsteil (Excel-Template), in dem die Testschritte und die erwarteten Ergebnisse definiert werden sowie
2. einem Run-Time-Teil, der mit dem Controller kommuniziert und Testschritte vorgibt und Ergebnisse abfragt.

Im Rahmen von Lehrlingsarbeiten und Praktika wird der Definitionsteil komplett neu erstellt. Dazu gibt es bereits ein angefangenes Excel-Projekt, das in dieser IPA weiterbearbeitet werden soll.

Das Ziel dieser IPA ist es, eine Tabelle mit allen zu testenden Variablen (das sind hier BACnet-Objekte) zu erstellen.

Dazu müssen Tabellen, Bedienmasken und VBA-Programme gemacht werden, die dem Benutzer eine effiziente Eingabe der Daten ermöglichen.

1.2.2 Detaillierte Aufgabenstellung

Als Basis für die Arbeiten gilt eine Requirement-Specification (siehe Aufgabenstellung)

Neben der Spezifikation werden diverse Beispieldateien zur Verfügung gestellt. Diese dienen zur Erläuterung der Spezifikation und zum Test der Funktionen

Aufgabenstellung: Spezifikationsliste der Testvariablen für TsNet V2 gemäss Spezifikation

- Terminplanung und Projektstatus
- Alle 2 Tage ist ein Statusmeeting mit dem Auftraggeber durchzuführen
- Erstellung der Software
- Erstellung der Softwaredokumentation. Zielgruppe: SW-Entwickler, die das TsNet-Tool warten, pflegen und weiterentwickeln
- Erstellen der Testfälle und Durchführen von Tests.
- Test der Eingabedialoge mit korrekten und falschen Werten
- Test der Useability der Eingabedialoge mit Fachperson ausserhalb vom TsNet-Projekt
- Test aller erstellten Funktionen gegenüber der Spezifikation
- Test aller erstellten Funktionen gegen Fehlbedienung und fehlerhafte Daten und Dateien
- Entsprechend den IPA-Regeln ist ein Arbeitsjournal zu führen.

Erwartete Lieferungen:

- Terminpläne und Projektstatus alle 2 Tage und am Ende der IPA
- Excel-File mit der Spezifikationsliste gemäss Spezifikation
- kommentierter Source-Code
- kommentierte Eingabedialoge
- Softwaredokumentation: Es kann jede grafische Darstellung des Programmablaufs und des Datenflusses verwendet werden, sofern sie dazu geeignet ist, einem anderen Entwickler die Weiterarbeit zu ermöglichen. Komplexere Funktionen wie die Plausibilitätstests sollen in Struktogrammen dokumentiert werden.
- Testspezifikation und Dokumentation der Testergebnisse
- Arbeitsjournal

Entsprechend der internationalen Ausrichtung ist die Spezifikation in Englisch geschrieben. Die gesamte IPA-Dokumentation wird in Deutsch erstellt. Sollten Teile der Spezifikation oder andere englischsprachige Dokumente in der IPA-Dokumentation verwendet werden, bleiben sie in der Originalsprache.

1.2.3 Mittel und Methoden

Entwicklungsumgebung:

- Standard-PC mit Microsoft Windows 7
- Microsoft Excel 2007. Programmierung in VBA.
- Erstellung der Dokumentation, der Präsentation und weiterer Dokumente mit Microsoft Office 2007
- Firmen-Richtlinie Codierungsrichtlinie Visual Basic und weitere Vorgaben in der Spezifikation.

1.2.4 Vorkenntnisse

Wanda hat bereits im Vorfeld der IPA an anderen Teilen von TsNet mit der gleichen Arbeitsumgebung gearbeitet.
Programmieren mit VBA ist ihr aus dieser Tätigkeit und aus anderen Projekten bekannt.

1.2.5 Vorarbeiten

Im Vorfeld der IPA wurden bereits Arbeiten von anderen Mitarbeitern am Projekt durchgeführt. Daher setzt Wanda auf einem bestehenden VBA-Projekt auf.
Wanda hat auch bereits mit der bestehenden TsNet-Version als Anwender gearbeitet, so dass ihr der Arbeitsablauf mit diesem Tool bekannt ist.

1.2.6 Neue Lerninhalte

- keine

1.2.7 Arbeiten in den letzten 6 Monaten

- Erstellen von Skripten zum Automatischen Generieren eines Builds für ein anderes Projekt (ca. 2 Monate)
- Weiterarbeit an mehreren Excel VBA-Projekten (ca. 2 Monate)
- Programmierarbeiten im Projekt TsNet V2 in Excel VBA (ca. 2 Monate)

1.3 Projektorganisation

1.3.1 Datensicherung

Die gesamte IPA-Ordnerstruktur mit dem betreffenden Code befindet sich auf den Siemens Server, von welchem täglich automatisch ein Backup gemacht wird. Zusätzlich wird vom IPA-Ordner am Ende des Arbeitstages eine Kopie erstellt, welche auf einer lokalen Festplatte gespeichert wird. Dadurch kann man jederzeit auf alle bisherigen Versionen zurückgreifen, falls Daten gelöscht oder beschädigt worden sind.

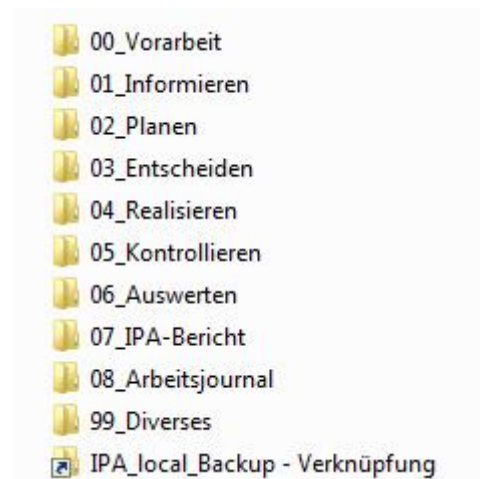


Abbildung 1: Siemens Server

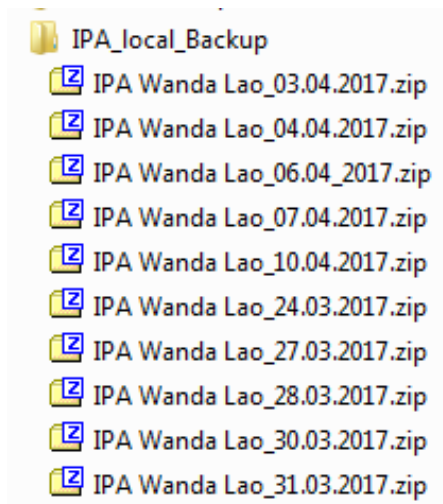


Abbildung 2: Lokaler Backup

1.3.2 Beteiligte Dienste, Fachabteilungen und Personen

Firma	Siemens Schweiz AG, Building Technologies Division
Abteilung	BT CPS R&D ZG CS SAP
Fachvorgesetzter	Michael Speckien
Stellvertreter und Betreuer	Urs Heimann (Stellvertreter und Betreuer während der Abwesenheit von Michael Speckien)
Verwendete Software	<ul style="list-style-type: none"> - Microsoft Office 2007 - HUS Struktogrammer [3]
Verwendete Tools	<ul style="list-style-type: none"> - Snipping Tool (ist Bestandteil von Windows 7)

1.3.3 Abteilung BT CPS R&D ZG CS SAP

Die Abteilung „Applikation“ erstellt Libraries/Funktionsblöcke für freiprogrammierbare Geräte. Diese Geräte werden zur Gebäudeautomation verwendet und steuern Storen, Lichter, Heizungen und Lüftungen. Mit dem Programm ABT kann man sich die notwendigen Funktionen für ein Gerät zusammensuchen und abändern. Schliesslich kompiliert man und lädt die erzeugte Software auf das Gerät (z.B. DXR- oder PXC-Controller).

Die Geräte setzen Befehle von Sensoren oder Interface (Control-Panel/Web-Browser) um. ABT verwendet eine Art graphische oder diagrammähnliche Programmierung.

1.3.4 Verwendete Projektmanagementmethode

Bei der Arbeit an dem Projekt wird die Projektmanagementmethode IPERKA verwendet

Die Abkürzung IPERKA steht für:

- 1) Informieren
- 2) Planen
- 3) Entscheiden
- 4) Realisieren
- 5) Kontrollieren
- 6) Auswerten

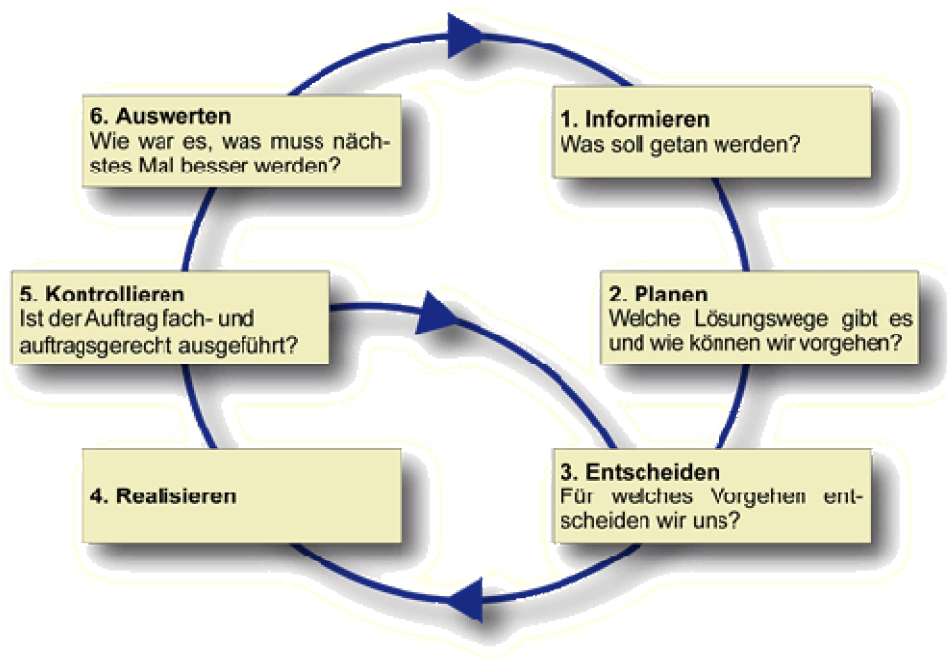


Abbildung 3: IPERKA [2]

Begründung der Wahl

Die ausgewählte Projektmanagementmethode „IPERKA“ passt gut zum Arbeitsstil der IPA-Ausführende, da bei dieser Methode die einzelnen Tätigkeiten, die von ihr erstellt wurden, in die entsprechenden Schritten zugeordnet werden können. Die Projektmethode wird auch in der Schule gelehrt und hat sich in diversen Projekten in der Schule bereits bewährt. Ebenfalls wurde mit dieser Methode in der Lernwerkstatt gearbeitet.

1.3.5 Arbeitsplatz

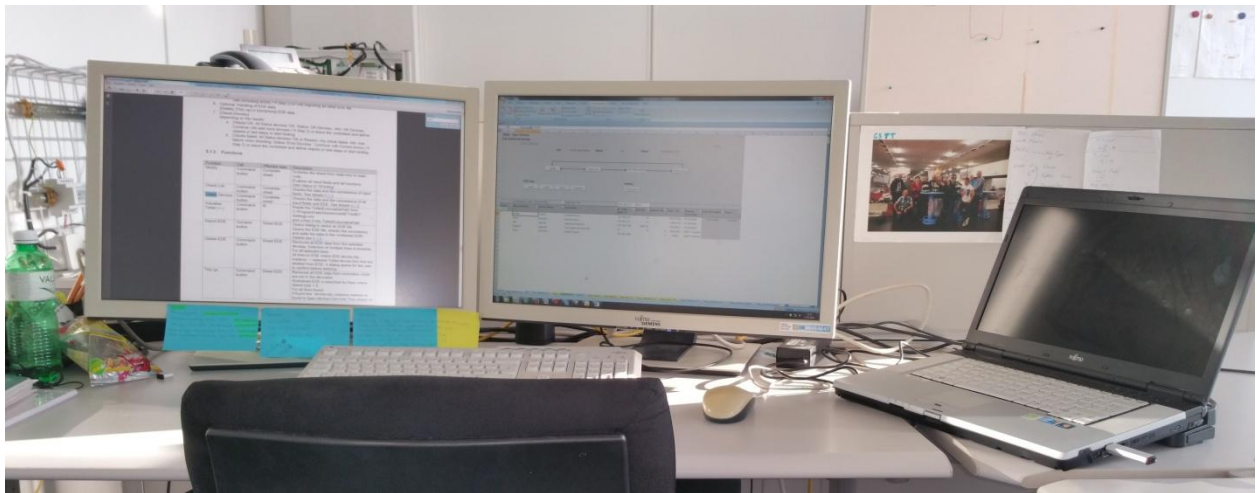


Abbildung 4: Arbeitsumgebung

Der Arbeitsplatz befindet sich in Zug am Zählerweg 5 im vierten Stock. Ein Fujitsu Laptop mit zwei zusätzlichen Monitoren steht für die IPA zur Verfügung.

1.3.6 Ordnerstruktur auf Server

Wie folgt ist der IPA-Ordner so aufgebaut:



Abbildung 5: Ordnerstruktur

1.3.7 Risikobeschreibung

In der nachfolgenden Tabelle „Tabelle 2: Risikobeschreibung“ wird beschrieben, was den Ablauf der IPA in Gefahr bringen könnte.

VBA – Kenntnisse	Falls die IPA-Ausführende bei der Implementierung Probleme haben sollte, könnte es sein, dass der Terminplan nicht eingehalten wird, somit werden die Bestandteile von IPA nicht termingerecht implementiert und getestet. Das würde zu einer Verzögerung im Zeitplan führen.	
	Wahrscheinlichkeit	50%
	Auswirkungsgrad	Kritisch
Akzeptanztest eines Applikationsentwicklers	Für den Akzeptanztest wird mit der Testperson einen Termin vereinbart. Falls der Mitarbeiter nicht zur Verfügung steht, sollte so schnell wie möglich eine Ersatzperson gefunden werden.	
	Wahrscheinlichkeit	10%
	Auswirkungsgrad	Leicht
Fehlschlag von Akzeptanztest	Wenn der Akzeptanztest fehlschlagen würde, so könnte keine Änderungen oder Korrekturen mehr gemacht werden. Der Zeitplan kann nicht eingehalten werden.	
	Wahrscheinlichkeit	25%
	Auswirkungsgrad	Kritisch
Zeitplan	Im Zeitplan wurde eine Pufferzeit von 0.8 h pro Arbeitstag eingeplant. Damit hat die IPA-Ausführende am Schluss acht Stunden als Reservezeit, um allfälligen Verzögerungen zu korrigieren.	
	Wahrscheinlichkeit	10%
	Auswirkungsgrad	Kritisch

Tabelle 2: Risikobeschreibung

1.4 Planung

Die Zeitplanung ist in acht Teilen aufgeteilt und nach dem IPERKA-Modell erstellt worden.

Teil 1	Informieren
Teil 2	Planen
Teil 3	Entscheiden
Teil 4	Realisieren
Teil 5	Kontrollieren
Teil 6	Auswerten
Teil 7	Diverses
Teil 8	Reserve

Tabelle 3: Zeitplan – Teile

Legende

x	Soll-Zeit
x	Ist-Zeit
	Meilenstein
offen	Tätigkeit ist offen (noch nicht angefangen)
i.A.	Tätigkeit ist in Arbeit
ok	Tätigkeit ist abgeschlossen

Tabelle 4: Zeitplan – Legende

1.4.1 Zeitplan: Soll

Nr.	Tätigkeit	Abhängigkeit		Aufwand		Status		FR	MO	DI	DO	FR	MO	DI	DO	FR	MO	
		Vorbereitung	Plan (h)	Ist (h)	Differenz (h)	Priorität	Meilenstein Ende (Datum)	Status	24.03.2017	27.03.2017	28.03.2017	30.03.2017	31.03.2017	03.04.2017	04.04.2017	06.04.2017	07.04.2017	10.04.2017
									Soll	Ist	Soll	Ist	Soll	Ist	Soll	Ist	Soll	Ist
1	Informieren		1.0	0.0														
1.1	Projektauftrag lesen und verstehen, Abläufe für Hilfestellung notieren		1.0			1		ok	1.0									
2	Planen		6.0	0.0														
2.1	Tätigkeiten und Meilensteine definieren		1.1	1.5		1		ok	1.5									
2.2	Soll-Zeitplan erstellen		1.1	2.0		1	24.03.17	ok	2.0									
2.3	Vorlagen erstellen und IPA-Bericht aufbauen		2.1, 2.2	0.5		1		offen	0.5									
2.4	Testkonzept erstellen			2.0		1		offen	1.0	1.0								
2.5	Arbeitsumgebung einrichten			0.0		1		offen	0.0									
3	Entscheiden		2.0	0.0														
3.1	Akzeptanztest-Spezifikation ermitteln			2.0		1		offen		2.0								
4	Realisieren		35.0	0.0														
4.1	Struktogramme erstellen			1.5		1		offen		1.5								
4.2	Bedienoberfläche (ActiveX Steuerelemente) benennen			0.5		1		offen		0.5								
4.3	Bedienoberfläche (Userform) erstellen und benennen			0.5		1	28.03.17	offen			0.5							
4.4	Implementierung von "WBOpenObj"		4.2	1.5		1		offen			1.5							
4.5	Implementierung von "Modify"		4.2	1.0		1		offen			1.0							
4.6	Implementierung von "CheckAlias"		4.2	2.0		1		offen			2.0							
4.7	Implementierung von "AppendFromEDE"		4.1, 4.2, 4.3	3.0		1		offen			3.0							
4.8	Implementierung von "ObjToAlias"		4.1, 4.2, 4.3	3.0		1		offen			3.0							
4.9	Implementierung von "AliasToObjName"		4.1, 4.3	3.0		1		offen				3.0						
5	Implementierung von "BACnetDataEDE"		4.1, 4.2	5.0		1		offen				2.5	2.5					
5.1	Implementierung von "CheckObj"		4.1, 4.2, 4.9	4.0		1		offen					3.5	0.5				
5.2	Implementierung von "Select"		4.2	1.0		2		offen						1.0				
5.3	Implementierung von "Deselect"		4.2	1.0		1		offen						1.0				
5.4	White-Box-Test ermitteln		4.1-5.3	8.0		1	06.04.17	offen						2.5	5.5			
6	Kontrollieren		7.0	0.0														
6.1	White-Box-Test durchführen und berichten		5.4	2.0		1		offen								2.0		
6.2	Fehlerbehebung		6.1	2.0		1		offen								2.0		
6.3	White-Box-Nachtest durchführen und berichten		6.2	1.0		1		offen								1.0		
6.4	Akzeptanztest durchführen		3.1, 4.1-5.3	2.0		1		offen										2.0
7	Auswerten		2.0	0.0														
7.1	Schlussbericht verfassen		6	2.0		1		offen										2.0
8	Diverses		19.0	0.0														
8.1	Zeitplan Ist-Zustand einfügen (10*0.1)			1.0		1		offen	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1
8.2	Arbeitsjournal führen (10*0.2)			2.0		1		offen	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.2
8.3	IPA-Bericht führen (10*1)			10.0		1		offen	0.5	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.5	
8.4	Statusmeeting mit dem Auftraggeber			3.0		1		offen	0.5		0.5		0.5		0.5		0.5	0.5
8.5	Gespräch mit dem Erstexperten			1.0		1		offen		1.0								
8.6	Administratives/Organisatorisches			1.0		1		offen	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	
8.7	Abgabe IPA: drucken, binden, uploaden			1.0		1	10.04.17	offen										1.0
9	Reserve		8.0	0.0														
9.1	Pufferzeit (10*0.8)			8.0					0.8	0.8	0.8	0.8	0.8	0.8	0.8	0.8	0.8	0.8
	Total		80.0	0.0					8.2	0.0	8.2	0.0	7.7	0.0	8.2	0.0	7.7	0.0

Abbildung 6: Zeitplan - Soll

1.4.2 Zeitplan: Soll-Ist-Vergleich

Nr.	Tätigkeit	Abhängigkeit		Aufwand			Status		FR		MO		DI		DO		FR		MO		DI		DO		FR		MO	
		Vorbedingung	Plan (h)	Ist (h)	Differenz (h)	Priorität	Meilenstein Ende (Datum)	Status	24.03.2017		27.03.2017		28.03.2017		30.03.2017		31.03.2017		03.04.2017		04.04.2017		06.04.2017		07.04.2017		10.04.2017	
									Soll	Ist	Soll	Ist	Soll	Ist	Soll	Ist	Soll	Ist	Soll	Ist	Soll	Ist	Soll	Ist	Soll	Ist	Soll	Ist
1	Informieren		1.0	0.0																								
1.1	Projektauftrag lesen und verstehen, Abläufe für Hilfestellung notieren		1.0	0.0		1	ok	1.0	1.0																			
2	Planen		6.0	6.0																								
2.1	Tätigkeiten und Meilensteine definieren		1.1	1.5	1.5	1	ok	1.5	1.5																			
2.2	Soll-Zeitplan erstellen		1.1	2.0	2.0	1	24.03.17	ok	2.0	2.0																		
2.3	Vorlagen erstellen und IPA-Bericht aufbauen		2.1, 2.2	0.5	0.5	1	i.A.	0.5	0.5																			
2.4	Testkonzept erstellen			2.0	2.0	1	i.A.	1.0	1.0	1.0	1.0																	
2.5	Arbeitsumgebung einrichten			0.0	0.0	1	ok	0.0	0.0																			
3	Entscheiden		2.0	2.0																								
3.1	Akzeptanztest-Spezifikation ermitteln			2.0	2.0	1	i.A.			2.0	2.0																	
4	Realisieren		35.0	34.8																								
4.1	Struktogramme/Flussdiagramme erstellen			1.5	2.0	1	i.A.			1.5	2.0																	
4.2	Bedienoberfläche (ActiveX Steuerelemente) benennen			0.5	0.5	1	ok			0.5	0.5																	
4.3	Bedienoberfläche (Userform) erstellen und benennen			0.5	0.5	1	28.03.17	ok					0.5	0.5														
4.4	Implementierung von "WBopenObj"		4.2	1.5	1.5	1	ok						1.5	1.5														
4.5	Implementierung von "Modify"		4.2	1.0	1.0	1	ok						1.0	1.0														
4.6	Implementierung von "CheckAlias"		4.2	2.0	2.0	1	ok						2.0	2.0														
4.7	Implementierung von "AppendFromEDE"		4.1, 4.2, 4.3	3.0	4.0	1	ok						3.0	4.0														
4.8	Implementierung von "ObjToAlias"		4.1, 4.2, 4.3	3.0	3.0	1	ok						3.0	3.0														
4.9	Implementierung von "AliasToObjName"		4.1, 4.2, 4.3	3.0	2.0	1	ok						3.0	2.0														
5	Implementierung von "BACnetDataEDE"		4.1, 4.2, 4.3	5.0	4.5	1	ok						2.5	3.0	2.5	1.5												
5.1	Implementierung von "CheckObj"		4.1, 4.2, 4.9	4.0	3.8	1	ok									3.5	3.5	0.5	0.3									
5.2	Implementierung von "Select"		4.2	1.0	0.5	2	ok											1.0	0.5									
5.3	Implementierung von "Deselect"		4.2	1.0	0.5	2	ok											1.0	0.5									
5.4	White-Box-Test ermitteln		4.1-5.3	8.0	9.0	1	06.04.17	ok										2.5	2.5	5.5	6.5							
6	Kontrollieren		7.0	7.5																								
6.1	White-Box-Test durchführen und berichten		5.4	2.0	2.0	1	ok																2.0	2.0				
6.2	Fehlerbehebung		6.1	2.0	3.0	1	ok																2.0	3.0				
6.3	White-Box-Nachtest durchführen und berichten		6.2	1.0	1.0	1	ok																1.0	1.0				
6.4	Akzeptanztest durchführen		3.1, 4.1-5.3	2.0	1.5	1	ok																		2.0	1.5		
7	Auswerten		2.0	2.0																								
7.1	Schlussbericht verfassen		6	2.0	2.0	1	ok																		2.0	2.0		
8	Diverses		19.0	22.4																								
8.1	Zeitplan Ist-Zustand einfügen (10*0.1)			1.0	1.0	1	ok	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	
8.2	Arbeitsjournal führen (10*0.2)			2.0	2.0	1	ok	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.2	
8.3	IPA-Bericht führen (10*1)			10.0	13.5	1	ok	0.5	0.5	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	2.0	1.0	1.0	1.0	2.0	1.5	3.0		
8.4	Statusmeeting mit dem Auftraggeber			3.0	3.0	1	ok	0.5	0.5			0.5	0.5			0.5	0.5			0.5	0.5			0.5	0.5	0.5	0.5	
8.5	Gespräch mit dem Erstexperten			1.0	0.8	1	ok			1.0	0.8																	
8.6	Administratives/Organisatorisches			1.0	1.1	1	ok	0.1	0.1	0.1	0.2	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	
8.7	Abgabe IPA: drucken, binden, uploaden			1.0	1.0	1	10.04.17	i.A.																	1.0	1.0		
9	Reserve		8.0	5.6																								
9.1	Pufferzeit (10*0.8)			8.0	5.6			0.8	0.0	0.8	0.0	0.8	0.5	0.8	0.5	0.8	1.0	0.8	1.5	0.8	1.0	0.8	0.0	0.8	0.8	0.8	0.8	
	Total			80.0	80.3			8.2	7.4	8.2	7.8	7.7	7.4	8.2	8.9	8.2	7.9	8.2	7.9	7.7	7.7	7.7	7.9	7.7	9.7	8.2	9.2	

Abbildung 7: Zeitplan – Ist

1.4.3 Tätigkeiten

Nr.	Tätigkeit	Beschreibung
1.1	Projektauftrag lesen und verstehen, Abläufe für Hilfestellung notieren.	Die detaillierte Aufgabenstellung gemäss PkOrg lesen und verstehen. (Inklusive die PDF-Datei „Aufgabestellung“) Ebenfalls werde ich hier die Abläufe notieren, um somit einen Anfang zu finden und um die Aufgaben zuzuordnen.
2.1	Tätigkeiten und Meilensteine definieren	Die Tätigkeiten finden und beschreiben. Meilensteine, Zeitpunkt der Fertigstellung einer Aufgabe oder Tätigkeit, definieren und festlegen.
2.2	Soll-Zeitplan erstellen	Im Soll-Zeitplan werden die Stunden und das Datum für jeweilige Tätigkeiten bestimmt.
2.3	Vorlagen erstellen und IPA-Bericht aufbauen	Die Vorlagen für das Arbeitsjournal, den White-Box-Test und Akzeptanztest erstellen. Den IPA-Bericht erstellen und gliedern.
2.4	Testkonzept erstellen	Das Testkonzept erstellen.
2.5	Arbeitsumgebung einrichten	Die Arbeitsumgebung wurde bereits vor der IPA eingerichtet.
3.1	Akzeptanztest-Spezifikation ermitteln	Mit Hilfe der Aufgabenstellung (Detaillierte Aufgabenbeschreibung im PkOrg, die PDF-Datei) wird der Akzeptanztest ermittelt.
4.1	Struktogramme erstellen	Es werden Struktogramme nur für die komplexeren Funktionen „AppendFromEDE“, „ObjToAlias“, „AliasToObjName“, „BACnetDataEDE“, „CheckObjects“ und „CheckAlias“ erstellt.
4.2	Bedienoberfläche (ActiveX Steuerelemente) benennen	Die Elemente der Bedienoberfläche (die ActiveX Steuerelemente, wie z.B. Button [Schaltfläche]) benennen.
4.3	Bedienoberfläche (User Form) erstellen und benennen	Die zusätzlichen User Form der Bedienoberfläche erstellen und benennen.
4.4	Implementierung von "WBopenSpec"	Die Startfunktion wird implementiert.
4.5	Implementierung von "Modify"	Für die ActiveX Steuerelement „Modify“ wird eine Prozedur/Funktion implementiert.
4.6	Implementierung von "CheckAlias"	Für die ActiveX Steuerelement „CheckAlias“ wird eine Prozedur/ Funktion ergänzend implementiert.
4.7	Implementierung von "AppendFromEDE"	Für die ActiveX Steuerelement „AppendFromEDE“ wird eine Prozedur/Funktion gemäss Struktogramm implementiert. Hierzu wird eine User Form aufgerufen, deren Funktionen werden ebenfalls implementiert.
4.8	Implementierung von "ObjToAlias"	Für die ActiveX Steuerelement „ObjToAlias“ wird eine Prozedur/Funktion gemäss Struktogramm implementiert. Hierzu wird eine User Form aufgerufen, deren Funktionen werden ebenfalls implementiert.

4.9	Implementierung von "AliasToObjName"	Für die ActiveX Steuerelement „AliasToObjName“ wird eine Prozedur/Funktion implementiert.
5	Implementierung von "BACnetDataEDE"	Für die ActiveX Steuerelement „BACnetDataEDE“ wird eine Prozedur/Funktion gemäss Struktogramm implementiert. Hierzu wird eine User Form aufgerufen, deren Funktionen werden ebenfalls implementiert.
5.1	Implementierung von "CheckObj"	Für die ActiveX Steuerelement „CheckObj“ wird eine Prozedur/Funktion gemäss Struktogramm implementiert.
5.2	Implementierung von "Select"	Für die ActiveX Steuerelement „Select“ wird eine Prozedur/Funktion implementiert.
5.3	Implementierung von "Deselect"	Für die ActiveX Steuerelement „Deselect“ wird eine Prozedur/Funktion implementiert.
5.4	White-Box-Test ermitteln	Aus den Implementierungen #4.2 bis #5.3 werden jene Funktionen ausgewählt, die sinnvoll für die White-Box-Tests sind. Die Testfälle werden dann ermittelt.
6.1	White-Box-Test durchführen und berichten	Die Testfälle, welche im Schritt #5.4 definiert wurden, durchführen und darüber berichten.
6.2	Fehlerbehebung	Falls in Schritt #6.1 Fehler auftreten, werden diese hier behoben. Reaktionen auf ungewünschte Eingaben werden auch hier korrigiert.
6.3	White-Box-Nachtest durchführen und berichten	Falls der Schritt #6.2 durchgeführt wurde, muss ein Nachtest erfolgen und ebenfalls darüber berichten.
6.4	Akzeptanztest durchführen	Die Testfälle, welche im Schritt #3.1 ermittelt wurden, durchführen und darüber berichten.
7.1	Schlussbericht verfassen	Der Schlussbericht umfasst das Schlusswort der IPA-Ausführende über die Durchführung der IPA. (Rückblick, Reflexion, Fazit wurden nicht unterteilt)
8.1	Zeitplan Ist-Zustand einfügen (10*0.1)	Der Zeitplan wird jeden Tag mit den Ist-Stunden ergänzt.
8.2	Arbeitsjournal führen (10*0.2)	Am Ende des Arbeitstages wird das Arbeitsjournal ausgefüllt.
8.3	IPA-Bericht führen (10*1)	Alle Arbeitsschritte sowie Bestandteile der IPA werden im Bericht festgehalten.
8.4	Statusmeeting mit dem Auftraggeber	Gemäss PkOrg wird jeden zweiten Tag einen Statusmeeting von ca. 30min mit dem Auftragsgeber geführt. Beim Statusmeeting wird ein Bericht über die Tätigkeiten der IPA mündlich abgegeben.
8.5	Gespräch mit dem Erstexperten	Das Gespräch findet am 27.03.2017 statt, am 2. Tag der IPA. An diesem Tag werden Informationen betreffend IPA ausgetauscht.

8.6	Administratives/Organisatorisches	Unter Administratives/Organisatorisches gehören die Tätigkeiten wie E-Mails schreiben, Backup erstellen sowie Termine erstellen.
8.7	Abgabe IPA: drucken, binden, uploaden	Am 10.04.2017 muss der IPA-Bericht mit dem Deckblatt 1 ausgedruckt, gebunden und an den Auftraggeber zugestellt werden. Der Zweitexpert erhält ein zweites Exemplar mit dem Deckblatt 2.
9.1	Pufferzeit (10*0.8)	Die Gesamtreservezeit beträgt acht Stunden.

Tabelle 5: Tätigkeiten

1.4.4 Meilensteine

Nr.	Meilenstein	Erledigte Arbeitsschritte	Datum
2.2	Soll-Zeitplan erstellen	#1.1 bis #2.2	24.03.2017
4.2	Bedienoberfläche (User Form) erstellen und benennen	#2.3 bis #4.3	28.03.2017
5.4	White-Box-Test ermitteln	#4.4 bis #5.4	06.04.2017
8.7	Abgabe IPA: drucken, binden, uploaden	#6 bis #8.7	10.04.2017

Tabelle 6: Meilensteine

1.5 Arbeitsjournal

1.5.1 Zweck des Arbeitsjournals

Das Arbeitsjournal dient zur Orientierung der Fachvorgesetzten und der Experten über den Stand der Tätigkeiten, allfällige Probleme und Hilfestellungen während der IPA.

1.5.2 Anwendungsbereich, Abgrenzung

Das Arbeitsjournal ist nur im Zusammenhang mit der IPA von Wert und ersetzt keine Dokumentation. Es beinhaltet die Tätigkeiten, wichtigen Erkenntnisse und Ergebnisse der Arbeit sowie die nächsten Schritten.

1.5.3 Aufbau

Jeden Tag werden am Morgen alle Tätigkeiten, die erledigt werden müssen, eingetragen. Am Abend werden die einzelnen Tätigkeiten mit deren Status und dem endgültigen Ist- Zeit ergänzt.

- Die ID, Tätigkeit und den Status referenziert sich zum Zeitplan.
- Unter „Hilfestellung“ wird beschrieben, ob die IPA- Ausführende von einer Person Informationen zum Projektauftrag erhalten hat.
- Unter „Projektstatus“ wird festgelegt, in welchem Bearbeitungsstand sich das Projekt befindet.
- Unter „Mailverkehr“ steht, an wen während der IPA E-Mails geschrieben wurden.
- Bei „Notizen“ wird die genaue Beschreibung des Arbeitsverlaufs festgehalten.
- Bei „Ausblick“ werden die Tätigkeiten festgehalten, welche gemäss Zeitplan für den kommenden Arbeitstag anstehen oder vom aktuellen Arbeitstag übernommen wurden.

Erwähnte Personen:

Michael Speckien	Fachvorgesetzter
Urs Heimann	Vertretung von Michael Speckien und Testperson
Marcel Kaufmann	1. Experte
Pascal Näf	2. Experte

1.5.4 Arbeitsjournale vom 24.03.2017 bis 10.04.2017

24.03.2017	Tag	1	2	3	4	5	6	7	8	9	10
Statusmeetings		24.03.2017									
ID	Tätigkeit						Status		Soll	Ist	
1.1	Projektauftrag lesen und verstehen						Abgeschlossen		1.0	1.0	
2.1	Tätigkeiten und Meilensteine definieren						Abgeschlossen		1.5	1.5	
2.2	Soll- Zeitplan erstellen						Abgeschlossen		2.0	2.0	
2.3	Vorlagen und IPA-Bericht aufbauen						Abgeschlossen		0.5	0.5	
2.4	Testkonzept erstellen						In Bearbeitung		1.0	1.0	
2.5	Arbeitsumgebung einrichten						Abgeschlossen		0.0	0.0	
8	Diverses						-		1.4	1.4	
							Arbeitsdauer			7.4 h	
Hilfestellung		Keine									
Projektstatus		Im Zeitplan									
Mailverkehr		Email mit Excel-Tabelle des Zeitplans an Herr Kaufmann									
Notizen											
<p>Heute habe ich mit meiner IPA gestartet. Zunächst habe ich die Aufgabenstellung gemäss PkOrg nochmals genauer durchgelesen. Ebenfalls habe ich die PDF-Datei „Aufgabenstellung“ im PkOrg durchgelesen. Zur detaillierten Aufgabestellung inklusive der PDF-Datei habe ich mir Notizen gemacht. Dabei habe ich eine grobe Zuteilung für die Tätigkeiten gemacht. Dadurch, dass ich weiss, mit welcher Arbeitsmethode ich am arbeiten möchte, konnte ich mir die Zuteilung der Tätigkeiten erleichtern. Ich werde für die IPA mit der IPERKA-Methode arbeiten.</p> <p>Anhand der groben Zuteilung konnte ich den Zeitplan erstellen. Beim Ergänzen des Zeitplans konnte ich die genauen Tätigkeiten ermitteln und danach hinzufügen. Bei der Zeiteinteilung musste ich mir überlegen, wie viel Stunden, ich in eine Tätigkeiten investieren möchte. Dabei hat mir die Aufgabenstellung geholfen, für komplexere Tätigkeiten, vor allem bei der Realisierung, eine sinnvolle Zeiteinteilung zu machen. Die Meilensteine wurden auch hinzugefügt.</p> <p>Nachdem der Soll-Zeitplan erfolgt ist, konnte ich die Beschreibung der einzelnen Tätigkeiten schreiben. Diese habe ich in ein separates Dokument geschrieben. Den Inhalt des Dokumentes werde ich dann im IPA-Bericht hinzufügen. Der Grund warum ich ein separates Dokument erstellt habe ist, dass ich einen besseren Überblick habe. Die Meilensteine wurden im gleichen separaten Dokument festgehalten.</p> <p>Ich habe angefangen, dass Testkonzept zu schreiben. Dazu habe ich noch den IPA-Bericht ergänzt. Um 15.50 Uhr erfolgte das erste Statusmeeting mit dem Auftraggeber. Hierbei wurden hauptsächlich der Zeitplan und deren Tätigkeiten angeschaut und besprochen.</p> <p>Der Zeitplan musste minim bearbeitet werden. Ich bemerkte nämlich bei der Tätigkeiten-Beschreibung, dass mir eine Tätigkeit gefehlt hatte.</p>											
Ausblick											
<p>Am Montag, 27.03.2017 findet das Gespräch mit dem Experten statt, dieses findet um 08.00 Uhr statt. Nachdem Gespräch werde ich das Testkonzept noch fertig schreiben und danach die Akzeptanztest-Spezifikationen ermitteln. Ich werde an diesen Tag ebenfalls die Struktogramme/Flussdiagramme erstellen. Danach folgt das Benennen der Benutzeroberfläche.</p>											

27.03.2017	Tag	1	2	3	4	5	6	7	8	9	10
Statusmeetings		24.03.2017									
ID	Tätigkeit						Status		Soll	Ist	
2.4	Testkonzept erstellen						Abgeschlossen		1.0	1.0	
3.1	Akzeptanztest-Spezifikationen ermitteln						Abgeschlossen		2.0	2.0	
4.1	Struktogramme erstellen						In Bearbeitung		1.5	2.0	
4.2	Bedienoberfläche (ActiveX Steuerelemente) benennen						Abgeschlossen		0.5	0.5	
8	Diverses						-		2.4	2.3	
							Arbeitsdauer			7.8 h	
Hilfestellung		Keine									
Projektstatus		Im Zeitplan									
Mailverkehr		E-Mail an Herr Näf bezüglich der Abgabe des Berichtes E-Mail an Michael Speckien und Urs Heimann für die Termine									
Notizen											
<p>Meinen Arbeitstag wurde mit dem Gespräch mit dem Experten gestartet. Das Gespräch diente zur Kontrolle, allfällige Fragen sowie Bemerkungen der IPA. Das Gespräch dauerte ca. 45 Minuten. Fragen bezüglich der IPA gaben es nur vereinzelt. Die Frage wie das Struktogramm sein sollte (eher grob oder eher detailliert), wurde beantwortet. Ich solle es so machen, wie ich es gelernt habe. <i>Zusätzlich zum Struktogramm werde ich nur für sinnvolle Funktionen Struktogramme erstellen und nicht nur für die vier komplexere Funktionen. Es ist so gewünscht worden, da es besser für eine gute Übersicht ist. Es werden für die Tätigkeiten (Funktionen) #4.6 bis #5.1 Struktogramme erstellt. Für die Schritte #4.4, #4.5, #5.2 und #5.3 werde ich keine Struktogramme erstellen, da es nicht notwendig ist. Jedoch werde ich eine Auflistung mit Beschreibung der Funktionen im Bericht erstellen. Damit schlussendlich alle Funktionen, die innerhalb der IPA implementiert wurden, ersichtlich sind.</i></p> <p>Der Experte bestätigte, dass Urs Heimann die Woche 03.04 – 07.04 übernehmen kann, da Michael Speckien in dieser Woche zu einem Kunden muss. Der Termin für das Fachgespräch wurde provisorisch auf den 04.05 festgelegt. Es werden insgesamt vier Exemplare geben, eines für den Fachvorgesetzten, einen für den Erstexperten, einen für den Zweitexperten und die PDF-Datei fürs PkOrg. Der Fachvorgesetzte und der Experte erhalten ein Exemplar mit dem Deckblatt 1.</p> <p>Nach diesem Gespräch sendete ich ein E-Mail an den Zweitexperten und fragte ihn, ob er den IPA-Bericht gebunden oder in PDF-Format haben möchte. An den Zweitexperten wird ein Exemplar mit dem Deckblatt 2 als PDF versendet.</p> <p>Weiter habe ich das Testkonzept fertig geschrieben und angefangen die Akzeptanztest-Spezifikationen zu ermitteln. Hierbei hilft mir die Aufgabestellung für diese Tätigkeit und somit konnte ich die Testfälle definieren. Diese habe ich abgeschlossen.</p> <p>Die Bedienoberfläche (ActiveX Steuerelemente) habe ich zunächst auf ein Blatt Papier bezeichnet, für das druckte ich die Oberfläche aus. Somit hatte ich eine gute Übersicht und konnte sie übertragen.</p> <p>Die Struktogramme habe ich grob erstellt, doch zunächst musste ich mir einen Pseudocode auf Papier schreiben. Diese hat mir beim Erstellen der Struktogramme etwas erleichtert. Jedoch brauchte ich hierfür ein Wenig mehr Zeit als geplant.</p> <p>Ich habe auch noch alle Statusmeeting Termine mit Michael Speckien und oder Urs Heimann geschickt.</p>											
Ausblick											
<p>Morgen werde ich noch die Bedienoberfläche (User Form) erstellen und diese benennen. Insgesamt existieren drei Dialogfenster. Danach folgt die Implementierung. Die Funktionen „WBOpenObj“ und „Modify“ sollte morgen fertig werden sowie auch die Funktion CheckAlias.</p> <p>Ebenfalls findet morgen ein Statusmeeting statt.</p>											

28.03.2017	Tag	1	2	3	4	5	6	7	8	9	10
Statusmeetings		24.03.2017		28.03.2017							
ID	Tätigkeit					Status			Soll	Ist	
4.3	Bedienoberfläche (User Form) erstellen und benennen					Abgeschlossen			0.5	0.5	
4.4	Implementierung von „WBOpenObj“					Abgeschlossen			1.5	1.5	
4.5	Implementierung von „ModifyObj“					Abgeschlossen			1.0	1.0	
4.6	Implementierung von „CheckAlias“					Abgeschlossen			2.0	2.0	
8	Diverses					-			1.9	1.9	
9	Reserve					-			0.8	0.5	
						Arbeitsdauer			7.4 h		
Hilfestellung		Keine									
Projektstatus		Im Zeitplan									
Mailverkehr		Keinen									
Notizen											
<p>Heute habe ich die Struktogramme noch optimiert, aber auch einige Sachen ergänzt. Da ich für die Struktogramme ein bisschen mehr Zeit brauchte, habe ich es dementsprechend so eingetragen und diese Zeit von heute zur Reservezeit hinzugenommen.</p> <p>Ebenfalls habe ich die Bedienoberfläche der User Formen erstellt. Für die Funktionen „AppendFromEDE“, „AliasToObj“ und „ObjToAlias“ mussten User Formen erstellt werden. Für das habe ich zuerst die drei User Formen skizziert und diese dann so bearbeitet, bis sie brauchbar und optimal ist. Diese habe ich mit den dazugehörigen Steuerelemente erstellt und danach dementsprechend benennt. Für die Funktionen „WBOpenObj“ und „ModifyObj“ werden die gleichen Variablen benutzt. Deswegen habe ich eine zusätzliche Funktion erstellt, die alle Variablen setzten. Diese gebrauchten Variablen habe ich global gemacht. „WBOpenObj“ ist die Startfunktion, sie setzt die Tabelle auf geschützt und deaktiviert alle Buttons, die Felder sind grau. „Modify“ setzt die Tabelle auf ungeschützt, so dass man die Felder bearbeiten kann und die Buttons aktiviert sind, die Felder sind weiss. Alles gemäss Aufgabenstellung. „CheckAlias“ besteht schon teilweise. Deshalb habe ich die fehlende Abläufe noch ergänzt und auch noch das angepasst.</p> <p>Ebenfalls habe ich den IPA-Bericht verfeinert und notiert, was ich alles im Bericht haben möchte. Das Statusmeeting hatte heute mit Urs und Michael stattgefunden. Ich zeigte ihnen kurz meinen Stand. Dabei zeigte ich meinen Zeitplan und die Tätigkeiten, die bis hierher gemacht wurden. Soweit bin ich im Zeitplan mit einem kleinen Verzug aufgrund des Struktogrammes, was nicht schlimm ist. Es wurde keine Diskussion eröffnet.</p>											
Ausblick											
Am Donnerstag möchte ich die Funktionen „AppendFromEDE“ sowie „ObjToAlias“ erstellen und fertig implementieren. Weiterhin werde ich das ganz dokumentieren.											

30.03.2017	Tag	1	2	3	4	5	6	7	8	9	10
Statusmeetings		24.03.2017		28.03.2017							
ID	Tätigkeit					Status			Soll	Ist	
4.7	Implementierung von „AppendFromEDE“					Abgeschlossen			3.0	4.0	
4.8	Implementierung von „ObjToAlias“					Abgeschlossen			3.0	3.0	
8	Diverses					-			1.4	1.4	
9	Reserve					-			0.8	0.5	
						Arbeitsdauer			8.9 h		
Hilfestellung		Keine									
Projektstatus		Im Zeitplan									
Mailverkehr		Keinen									
Notizen											
<p>Bevor ich die Implementierung beginnen konnte, musste ich kurz die Struktogramme analysieren. Für die Funktion „AppendFromEDE“ musste ich das ganze genauer analysieren. Ich erstellte eine Notiz mit einem Pseudocode, die mir Zeit gekostet hat, da ich während dem Implementieren winzige Überlegungsfehler entdeckt hatte. Die Abläufe musste ich mir nochmals grob notieren und es dann umstrukturieren. Es hat aber doch eher Zeit gespart, als gekostet. Denn ohne Pseudocode hätte ich keine gute Übersicht, wie ich was und wie machen wollte. Die ganze User Form und Modul Geschichte hatte mich ein wenig aufgehalten sowie das Programmieren an sich. Ich musste nämlich ein wenig wieder reinkommen. Für diese Tätigkeit brauchte ich 1h mehr als geplant. Das könnte Auswirkungen auf meinen Zeitplan haben. Ich bin jedoch zuversichtlich.</p> <p>Für die Funktion „ObjToAlias“ musste ich keinen Pseudocode erstellen, diese war verständlicher für mich. Hier ging die Sache besser voran, vor allem das Ganze mit dem Modul und der User Form.</p> <p>Beim Realisieren der Funktionen ist mir aufgefallen, dass einige Zelle und Bereich Variablen mehrmals gebraucht wird. Nachdem Erkenntnis habe ich mich entschieden, die wieder verwendbare Variablen global zu machen und sie dann in einer Funktion, welche noch „setUsedVariables“ heisst, zu setzten. Der Name der Funktion wird im Verlauf der IPA noch angepasst.</p> <p>Weiter habe ich gemerkt, dass ich für einige Zellen bzw. Reihen und Spalten lieber „Konstanten“ nehmen sollte. Diese können ja variieren, was nicht oft oder gar nicht vorkommen sollte. Wären diese in Konstanten könnte man diese in der „set“-Funktion definieren. So könnte der nächste Programmierer eine leichtere Anpassung machen. Sonst müsste er alle Zellen durchgehen, um herauszufinden, was alles was ist. Ich habe es mir aufgeschrieben und möchte diese noch innerhalb der IPA realisieren.</p> <p>Für den Akzeptanztest habe ich drei Testfälle hinzugefügt, diese Zeit nehme ich ebenfalls von der Reserve.</p>											
Ausblick											
<p>Morgen, am Freitag, wird ein Statusmeeting stattfinden.</p> <p>Meine Tätigkeiten für morgen wäre die Implementierung für die Funktionen „AliasToObj“ und „BACnetFromEDE“. „AliasToObj“, wird wahrscheinlich ähnlich wie die heutigen Funktionen aufgebaut sein. Diese Funktion muss morgen fertig sein. „BACnetFromEDE“ werde ich morgen nicht fertig programmieren, denn es ist nur 2.5h für morgen eingetragen. Da es eine komplexere Funktion ist, wird die Programmierung länger sein, als bei den anderen.</p>											

31.03.2017	Tag	1	2	3	4	5	6	7	8	9	10
Statusmeetings		24.03.2017		28.03.2017		31.03.2017					
ID	Tätigkeit						Status		Soll	Ist	
4.9	Implementierung von „AliasToObjName“						Abgeschlossen		3.0	2.0	
5	Implementierung von „BACnetDataEDE“						In Bearbeitung		2.5	3.0	
8	Diverses						-		1.9	1.9	
9	Reserve						-		0.8	1.0	
							Arbeitsdauer			7.9 h	
Hilfestellung		Keine									
Projektstatus		Im Zeitplan									
Mailverkehr		Keinen									
Notizen											
<p>Heute habe ich die Funktion „AliasToObjName“ als erstes implementiert. Für diese brauchte ich eine Stunde weniger als geplant, da die Funktion ähnlich aufgebaut ist wie „ObjToAlias“ (#4.8). Jedoch beim Realisieren der Funktion „BACnetDataEDE“ musste ich meine Struktogramm nochmals analysieren. Für das habe ich mir wieder auf einem Blatt Papier eine Pseudocode und Notizen aufgeschrieben, aber nur für kleinere Codesequenzen. Hier habe ich insgesamt drei Stunden gebraucht, das ist 30 Minuten länger als geplant. Denn für diese Tätigkeit war für heute nur 2.5 Stunden gerechnet worden. Für diese habe ich gemäss Zeitplan aber noch am Montag Zeit.</p> <p>Ebenfalls hatte heute das 3.Statusmeeting stattgefunden. Wieder zeigte ich den Zeitplan und zusätzlich alle Funktionen, die bis jetzt implementiert wurden. Beim Vorzeigen klappte nicht alles, was geplant ist, so musste ich die Funktionen „AliasToObjName“ und „ObjToAlias“ nochmals anschauen. Ich bemerke hier, dass eine If-Schleife nicht richtig strukturiert war. Diesen Fehler konnte ich jedoch schnell beheben. Es hatte keinen Einfluss auf meinen Zeitplan.</p> <p>Weiter habe ich am IPA-Bericht gearbeitet. Ebenfalls wurde, wie auch für jeden Tag bis jetzt, ein Backup erstellt.</p>											
Ausblick											
Am Montag ist geplant, dass die Funktion „BACnetDataEDE“ fertig implementiert wird. Ebenfalls wird die Funktion „CheckObjects“ angefangen. Dabei können Struktogramm Anpassungen geben, da für diese Funktion bis jetzt nur ein grobes Struktogramm erstellt wurde.											

03.04.2017	Tag	1	2	3	4	5	6	7	8	9	10
Statusmeetings		24.03.2017		28.03.2017		31.03.2017					
ID	Tätigkeit						Status		Soll	Ist	
5	Implementierung von „BACnetDataEDE“						Abgeschlossen		2.5	1.5	
5.1	Implementierung von „CheckObjects“						In Bearbeitung		3.5	3.5	
8	Diverses						-		1.4	1.4	
9	Reserve						-		0.8	1.5	
							Arbeitsdauer			7.9 h	
Hilfestellung		Keine									
Projektstatus		Im Zeitplan									
Mailverkehr		Keinen									
Notizen											
<p>Heute war mein Ziel die Funktion „BACnetDataEDE“ abzuschliessen.</p> <p>Doch hier ist mir aufgefallen, dass man die Reihen und die Spalten (Rows & Columns) lieber global definiert. Deswegen habe ich die Funktion „setUsedVariables“ erweitert. Zusätzlich ist die neue Funktion „setConstantes“ dazugekommen, diese setzt die „Konstanten“ vom Modul. Da man im VBA keine Konstanten deklarieren kann, sind sie hier einfache globale Variablen.</p> <p>Alle Columns die ich bis hierher im Objects, habe ich mit der Konstante ersetzt. Diese kostet mir Zeit und war nicht geplant, deswegen geht diese Zeit auf die Reservezeit.</p> <p>Die Funktion „BACnetDataEDE“ habe ich abgeschlossen. Ich hatte ein Problem bei einer einfachen Verzweigung in einem Array. Die Funktion an sich funktioniert, jedoch muss ich hier etwas umschreiben. Es sind eher „Kosmetik-Fehler“, das heisst die Funktion an sich funktioniert, jedoch könnte sie besser strukturiert sein.</p> <p>Die Funktion „CheckObjects“ habe ich heute angefangen und konnte sie heute auch fast abschliessen. Diese Funktion beinhaltet viele Abfragen, ist an sich aber nicht allzu anspruchsvoll. Dabei haben mich aber die „Objecttype“ und „Typename“ etwas irritiert. Ich habe es so gelöst, dass ich den „Typename“ auf der Bedienoberfläche „Type“ zu „Typename“ geändert habe.</p>											
Ausblick											
<p>Morgen werde ich die Funktion „CheckObjects“ abschliessen, denn es braucht wirklich nicht viel. Die Funktionen „Select“ und „Deselect“ werde ich nicht viel bearbeiten, denn die Funktion „Select“ besteht. Ich Ebenfalls werde ich noch notwendige Codekommentare schreiben, dies dient zur Übersicht.</p> <p>Ebenfalls werde ich die White-Box-Testfälle ermitteln.</p>											

04.04.2017	Tag	1	2	3	4	5	6	7	8	9	10
Statusmeetings		24.03.2017		28.03.2017		31.03.2017		04.04.2017			
ID	Tätigkeit						Status		Soll	Ist	
5.1	Implementierung von „CheckObjects“						Abgeschlossen		0.5	0.3	
5.2	Implementierung von „Select“						Abgeschlossen		1.0	0.5	
5.3	Implementierung von „Deselect“						Abgeschlossen		1.0	0.5	
5.4	White-Box-Test ermitteln						In Bearbeitung		2.5	2.5	
8	Diverses						-		1.9	2.9	
9	Reserve						-		0.8	1.0	
							Arbeitsdauer			7.7 h	
Hilfestellung		Keine									
Projektstatus		Im Zeitplan									
Mailverkehr		Keinen									
Notizen											
<p>Die Funktion „CheckObjects“ habe ich abgeschlossen, ich musste nämlich nur noch ein paar Zeilen Code hinzufügen. Deshalb brauchte ich heute nicht lange dafür.</p> <p>Die „Select“ und „Deselect“ Funktionen konnten innerhalb einer Stunde vollendet werden. Die Funktion „Select“ besteht schon, deswegen bearbeitete ich sie nur mit Kleinigkeiten. Die volle Funktion kopierte ich und fügte es in die „Deselect“ Funktion, dort musste ich es nur so anpassen, dass es passte. Zusätzlich habe ich für diese zwei Codekommentare geschrieben.</p> <p>Das ganze Modul habe ich auch noch Kommentare hinzugefügt, vor allem für die If- und For-Schleifen, denn es gab mehrere Abfragen nacheinander. Da sind Kommentare nun sinnvoll eingesetzt worden.</p> <p>Um die White-Box-Testfälle zu ermitteln musste ich entscheiden, welche Funktionen überhaupt sinnvoll für den Test ist und welche nicht. Schlussendlich sind es vier Funktionen, „CheckObjects“, „BACnetDataEDE“, „ObjToAlias“ und „AppendFromEDE“. Es fiel mir schwer, für welche Durchläufe ich einen Haltepunkt setzten soll. Darum brauchte ich hier mehr „Denk-Zeit“. Somit konnte ich noch keine Testfälle schreiben, sondern nur entscheiden für welche ich Testfälle schreiben möchte.</p> <p>Die Akzeptanztestfälle habe ich ebenfalls angepasst, dafür habe ich die Reservezeit gebraucht. In dieser Zeit habe ich noch die Struktogramme bearbeitet bzw. angepasst.</p> <p>Heute fand ein Statusmeeting nur mit Urs Heimann statt. Ich habe ihm den aktuellen Stand gezeigt sowie auch die Implementierung. Meine nächsten Arbeitsschritte habe ich ihm auch vorgestellt. Dieses Meeting hat keine Diskussionen geöffnet.</p> <p>Der IPA-Bericht kommt bis jetzt sehr gut voran. Ich habe noch keine Bedenken.</p>											
Ausblick											
<p>Am Donnerstag werde ich die White-Box-Testfälle noch fertig ermitteln. Da am Freitag die Tests gemacht werden können, denke ich, brauche ich sehr wahrscheinlich mehr Zeit als geplant.</p> <p>Weiter werde ich fleissig an meinem IPA-Bericht schreiben.</p>											

06.04.2017	Tag	1	2	3	4	5	6	7	8	9	10
Statusmeetings		24.03.2017		28.03.2017		31.03.2017		04.04.2017			
ID	Tätigkeit						Status		Soll	Ist	
5.4	White-Box-Test ermitteln						Abgeschlossen		5.5	6.5	
8	Diverses						-		1.4	1.4	
9	Reserve						-		0.8	0.0	
							Arbeitsdauer			7.9 h	
Hilfestellung		Keine									
Projektstatus		Im Zeitplan									
Mailverkehr		Akzeptanztestfälle an Urs Heimann (Testperson) schicken									
Notizen											
<p>Heute habe ich vorwiegend die White-Box-Testfälle ermittelt. Diese fiel mir nicht so leicht, da ich lange gebraucht habe, mögliche Testfälle und Haltepunkte herauszufinden. Gestern habe ich mich für vier Funktionen entschieden, die ich testen möchte. Der Grund, warum ich nur diese vier Funktionen im White-Box-Test anschauen möchte, ist dass sie eine zusätzliche detaillierte Beschreibung hatte. Ein anderer Grund ist auch, dass die anderen Funktionen mittels Akzeptanztest genügend getestet werden. Im IPA-Bericht werden die Gründe für die jeweiligen Entscheide beschrieben.</p> <p>Schlussendlich habe ich insgesamt zehn White-Box-Testfälle geschrieben. Bei der Ermittlung ist mir aufgefallen, dass einige mögliche Testfälle nicht so sinnvoll sind und viel Zeit kosten.</p> <p>Für die Tätigkeit #5.4 White-Box-Testfälle ermitteln arbeitete ich länger daran als es geplant war. Meiner Meinung nach denke ich, dass ich trotzdem noch im Zeitplan stehe und genug Zeit für die folgende Tätigkeiten haben.</p> <p>Damit der Akzeptanztest von einer Testperson rechtzeitig durchgeführt werden kann, habe ich schon heute Urs Heimann eine E-Mail mit allen Dateien geschickt, die er braucht.</p> <p>Am IPA-Bericht selber habe ich vor allem am zweiten Teil gearbeitet.</p>											
Ausblick											
<p>Morgen werde ich den White-Box-Test machen und eventuelle Fehler beheben. Nach dem Ausführen vom White-Box-Test werde ich, falls nötig, einen Nachtest machen. Für den Test selbst wie auch für den Nachtest wird ein Bericht der Testergebnisse geschrieben.</p> <p>Auch nicht ganz korrekte Funktionen werden ergänzt und verbessert. Das gilt vor allem für das Färben der falschen Werte im Excel. Ich weiss, dass diese Funktionen noch nicht vervollständigt worden sind.</p> <p>Die Dokumentation sollte morgen fast vollständig sein. Da mein Ziel für morgen, die grobe Vollständigkeit vom Bericht, ist.</p>											

07.04.2017	Tag	1	2	3	4	5	6	7	8	9	10
Statusmeetings		24.03.2017	28.03.2017	31.03.2017	04.04.2017	07.04.2017					
ID	Tätigkeit	Status					Soll	Ist			
6.1	White-Box-Test durchführen und berichten	Abgeschlossen					2.0	2.0			
6.2	Fehlerbehebung	In Bearbeitung					2.0	3.0			
6.3	White-Box-Nachtest durchführen und berichten	In Bearbeitung					1.0	1.0			
8	Diverses	-					1.9	2.9			
9	Reserve	-					0.8	0.8			
Arbeitsdauer							9.7 h				
Hilfestellung		Keine									
Projektstatus		Im Zeitplan									
Mailverkehr		Keinen									
Notizen											
<p>Ich konnte heute alle White-Box-Testfälle durchführen. Vier Tests sind nicht durchgekommen. Bei diesen vier habe ich die Fehler aufgeschrieben und korrigierte sie anschliessend auch. Doch bei einem Testfall war die Korrektur schwieriger als gedacht. Ich habe sie zunächst gelassen und machte den Nachtest für die fehlgeschlagene Testfälle. Diese der, die ich nochmals getestet habe, sind nun durchgekommen.</p> <p>Für den einen Testfall hatte ich noch die übrige Zeit genutzt, um den Fehler zu beheben. Diese konnte ich immer noch nicht beheben. Ich entschied mich diese erst später zu erledigen oder am Montag, da meiner Meinung nach es nicht so wichtig ist.</p> <p>Heute fand noch ein Statusmeeting mit Urs Heimann statt. Ich habe ihm gestern die die Akzeptanztestfälle sowie das Produkt geschickt. Er schickte mir die Ergebnisse heute Morgen. Der Akzeptanztest ist nicht so herausgekommen wie er sein sollte. Es liegt aber daran, dass ich den White-Box-Test noch nicht durchgeführt hatte. Die Fehler, die aufgedeckt wurden, sind im Grunde die Fehler, welche ich heute beim White-Box-Test ebenfalls als fehlgeschlagen habe.</p> <p>Ebenfalls kam noch eine E-Mail von Michael Speckien herein, die besagt, dass er länger beim Kunden sein muss und somit nicht an meiner IPA-Abgabe anwesend sein wird. Für diese Angelegenheit begleitet mich Urs Heimann bis zum Schluss der IPA.</p> <p>Die einzelnen externen Dokumente, welche auch in den IPA-Bericht rein müssen, habe ich so vorbereitet, dass sie nur noch hinzugefügt werden kann. Am IPA-Bericht selber habe ich ebenfalls weiter daran gearbeitet, vor allem am zweiten Teil.</p>											
Ausblick											
<p>Am Montag werde ich Maximum, wenn überhaupt, eine Stunde an der 2. Fehlerbehebungsphase arbeiten, falls diese in dieser Zeit nicht verbessert werden kann, so setzte ich diese Tätigkeit auf nach der IPA. Denn der IPA-Bericht hat für mich höchste Priorität.</p> <p>Ebenfalls werde ich den Akzeptanztest noch durchführen und vorwiegend am IPA-Bericht arbeiten. Dazu kommen auch das Schlusswort und die Bezeichnung für die Tabellen und Abbildungen.</p> <p>Ebenfalls wird ein letztes Statusmeetig durchgeführt.</p>											

10.04.2017	Tag	1	2	3	4	5	6	7	8	9	10		
Statusmeetings		24.03.2017		28.03.2017		31.03.2017		04.04.2017		07.04.2017		10.04.2017	
ID	Tätigkeit						Status			Soll	Ist		
6.4	Akzeptanztest durchführen						Abgeschlossen			2.0	1.5		
7.1	Schlussbericht verfassen						Abgeschlossen			2.0	2.0		
8	Diverses						-			3.4	4.9		
9	Reserve						-			0.8	0.8		
							Arbeitsdauer			9.2 h			
Hilfestellung		Keine											
Projektstatus		Im Zeitplan											
Mailverkehr		Keinen											
Notizen													
<p>Heute habe ich den Akzeptanztest durchgeführt. Jedoch bevor ich das gemacht habe, habe ich den Code von der Funktion „CheckObjects“ nochmals angeschaut. Dabei habe ich gemerkt, dass die Zeit für die zweite Fehlerbehebungsphase nicht reicht. So entschied ich mich, diese nicht zu machen.</p> <p>Vorwiegend habe ich an der Dokumentation gearbeitet. Ich habe alle externe Dateien in den Bericht hinzugefügt und den Bericht ergänzt. Vor allem habe ich im Bericht auf die nicht Vollständigkeit von „CheckObjects“ aufmerksam gemacht. Ergänzt habe ich noch die Begründungen für die jeweiligen Entscheidungen. Es ist für mich wichtig, dass alles begründet ist. Denn ohne diese könnte es zu Missverständnisse kommen.</p> <p>Ebenfalls habe ich alle Tabellen und Abbildungen bezeichnet und dafür ein Tabellen- sowie Abbildungsverzeichnis erstellt. Das Glossar habe ich noch vervollständigt und danach nach Alphabet sortiert. Ansonsten habe ich den Bericht bearbeitet und dementsprechend ergänzt.</p> <p>Der Schlussbericht wurde mit Absicht nur mit dem Titel „Schlusswort“ bezeichnet. Sie beinhaltet den Rückblick, die Reflexion und das Fazit, ist jedoch nicht extra unterteilt.</p>													
Ausblick													
<p>Für die letzten Stunden werde ich das Dokument drucken und binden sowie der IPA-Bericht und die Anhänge auf PkOrg hochladen. Das werde ich machen, wenn das Dokument ausgedruckt ist. Deshalb ist diese Bemerkung in „Ausblick“ beschrieben und nicht unter „Notizen“.</p> <p>Ebenfalls werde ich den Bericht an Herr Näf mit dem Deckblatt 2 schicken. Das Exemplar mit dem Deckblatt 1 gebe ich Herr Speckien und Herr Kaufmann ab.</p>													

2. Teil 2

2.1 IPA Kurzfassung

2.1.1 Ausgangssituation

Das Projekt TsNet V2 besteht aus einem Definitionsteil und einem Run-Time-Teil. Wegen funktionalen Erweiterungen, schlechter Wartbarkeit und alten Excel- und Windows-Versionen muss der Definitionsteil komplett neu erstellt werden. Es sind mehrere Excel-Tabellen vorhanden, die eine neue Benutzeroberfläche mit deren Funktionen braucht. Ziel der IPA ist es, eine Tabelle mit allen zu testenden Variablen, hierbei handelt es sich um BACnet-Objekte, zu erstellen. Dazu müssen Tabellen, Bedienmasken und VBA-Programme gemacht werden, die eine effiziente Eingabe der Daten ermöglichen.

2.1.2 Umsetzung

Im Rahmen der individuellen praktischen Arbeit werden die Anforderungen, die in der Ausgangssituation genannt wurden, umgesetzt. Da für die Implementierung und Dokumentation der IPA-Ausführenden nur zehn Tage zur Verfügung stehen, wurde ein Zeitplan über zehn Tagen erstellt. Bei der Umsetzung wurde die Projektmanagementmethode IPERKA verwendet.

Neben der Zeitplanung gehört in der Planungsphase auch die Tätigkeiten Beschreibung, wie auch ein Testkonzept, welche erstellt wurden. In der Entscheidungsphase wurden die Akzeptanztest-Spezifikationen mit Hilfe der gewünschten Anforderungen ermittelt.

In der Realisierungsphase wurde die Bedienoberfläche, bei der die ActiveX-Steuerelemente verwendet wurden, im Excel optisch angepasst. Diese mussten benannt werden. Auch weitere User Formen mussten erstellt werden. Für eine bessere Übersicht für das Projekt wurden für komplexere Funktionen Struktogramme, mit Hilfe der Anforderungen, erstellt.

Anhand der Struktogramme und der Anforderungen gemäss Requirement-Specification konnten die Funktionen erstellt und realisiert werden. Hierbei werden die Namen und Informationen der BACnet Daten ermittelt, erstellt und überprüft. Hier gibt es vier Funktionen die das Ermitteln und die Erstellung der Daten ermöglichen. Zwei weitere Funktionen überprüfen die eingehenden Daten auf allfällige Fehler. Auch zur Realisierungsphase gehört das Ermitteln der White-Box-Testfälle.

In der Kontrollphase wurden von der IPA-Ausführenden die White Box Tests, die sie während der Implementierung ermittelt hat, ausgeführt. Der Akzeptanztest wurde am Schluss von einem Siemens Mitarbeiter durchgeführt.

2.1.3 Ergebnis

Nach dem Abschluss dieser Arbeit sind alle Anforderungen, ausser die Funktion „CheckObjects“, gemäss Requirement-Specification erfüllt. Die Bedienoberfläche verfügt über 7 funktionierende Buttons, 1 Link, 3 Labels, welche den Status, deren Information und das heutige Datum anzeigen. Für eine gute Orientierung auf der Bedienoberfläche werden die jeweiligen Buttons, die im aktuellen Arbeitsschritt nicht verwendet werden dürfen sowie nicht IPA-relevante Buttons, deaktiviert.

Der Hauptteil des Programmcodes befindet sich im Excel im Modul „mdl_SpecObjects“. Der Programmcode ist kommentiert und hat einen Kommentar-Header. Zusätzlich befinden sich im Code mehrere Funktionen, die für eine einfachere Deklaration der Variablen dienen. Diese ist keine Anforderung gemäss Requirements-Specification. Sie dient zur Übersicht des Programmcodes.

2.2 Realisierung

2.2.1 Bedienoberfläche

Für diese Bedienoberfläche wurden die Funktionen gemäss Spezifikation implementiert. Im Kapitel 2.2.6 *Implementierung* befindet sich die genaue Beschreibung der einzelnen Funktionen.

Die Buttons „from Test“ und „SDU > Comment“ sind nicht IPA relevant und wurden für die IPA dauerhaft deaktiviert.

Select	Alias Name	Alias Controller	Init value	Objectname	Controllername	Description	IP - Addr/Node-ID	Network. No	Dev.Inst.	Typename	Type	Instance
✓	CmdA	AS01		B01F01RSegm01AF01CmdA	AS01	Command analog	192.168.1.20		123000	AV	2	5
✓	Valin1	AS01		B01F01RSegm01AF01Valin1	AS01	Value input 1	192.168.1.20		123000	AV	2	6
✓	Valin2	AS01		B01F01RSegm01AF01Valin2	AS01	Value input 2	192.168.1.20		123000	AV	2	8
✓	Valin3	AS01		B01F01RSegm01AF01Valin3	AS01	Value input 3	192.168.1.20		123000	AV	2	8
✓	Valin4	AS01		B01F01RSegm01AF01Valin4	AS01	Value input 4	192.168.1.20		123000	AV	2	10
✓	PvVal	AS01		B01F01RSegm01AF01PvVal	AS01	Present value	192.168.1.20		123000	AV	2	10
✓	Valid	AS01		B01F01RSegm01AF01Valid	AS01	Valid	192.168.1.20		123000	BV	5	1
✓	DaliBusSta	AS01		DaliBusDaliBusSta	AS01	DALI bus state	192.168.1.20		123000	MV	19	6
✓	DaliBusMgmt	AS01		DaliBusDaliBusMgmt	AS01	DALI bus management	192.168.1.20		123000	MV	19	6
✓	DiagDaliBus	AS01		DiagDaliBus	AS01	Diagnostics for DALI bus	192.168.1.20		123000	MV	19	3
✓	PinkBusSta	AS01		PinkBusPinkBusSta	AS01	KNX PL-Link bus state	192.168.1.20		123000	MV	19	3
✓	DiagPinkBus	AS01		DiagPinkBus	AS01	Diagnostics for KNX PL-Link bus	192.168.1.20		123000	MV	19	4
✓	PinkBusMgmt	AS01		PinkBusPinkBusMgmt	AS01	KNX PL-Link bus management	192.168.1.20		123000	MV	19	4

Abbildung 8: Bedienoberfläche

Die Bedienoberfläche wird unterteilt zwischen Funktionsfeld und Inputfeld. Im Funktionsfeld befinden sich sechs Teile.

Inputfeld

Das Inputfeld beinhaltet die BACnet Daten. Dieses Feld ist weiss markiert. Weiss steht für ein bearbeitbares Feld, sie färbt sich grau ein, wenn die Tabelle schreibgeschützt ist. Diese Einstellung ist in der Spezifikation beschrieben und wurde so umgesetzt.

✓	CmdA	AS01		B01F01RSegm01AF01CmdA	AS01	Command analog	192.168.1.20		123000	AV	2	5
✓	Valin1	AS01		B01F01RSegm01AF01Valin1	AS01	Value input 1	192.168.1.20		123000	AV	2	6
✓	Valin2	AS01		B01F01RSegm01AF01Valin2	AS01	Value input 2	192.168.1.20		123000	AV	2	8
✓	Valin3	AS01		B01F01RSegm01AF01Valin3	AS01	Value input 3	192.168.1.20		123000	AV	2	8
✓	Valin4	AS01		B01F01RSegm01AF01Valin4	AS01	Value input 4	192.168.1.20		123000	AV	2	10
✓	PvVal	AS01		B01F01RSegm01AF01PvVal	AS01	Present value	192.168.1.20		123000	AV	2	10
✓	Valid	AS01		B01F01RSegm01AF01Valid	AS01	Valid	192.168.1.20		123000	BV	5	1
✓	DaliBusSta	AS01		DaliBusDaliBusSta	AS01	DALI bus state	192.168.1.20		123000	MV	19	6
✓	DaliBusMgmt	AS01		DaliBusDaliBusMgmt	AS01	DALI bus management	192.168.1.20		123000	MV	19	6
✓	DiagDaliBus	AS01		DiagDaliBus	AS01	Diagnostics for DALI bus	192.168.1.20		123000	MV	19	3
✓	PinkBusSta	AS01		PinkBusPinkBusSta	AS01	KNX PL-Link bus state	192.168.1.20		123000	MV	19	3
✓	DiagPinkBus	AS01		DiagPinkBus	AS01	Diagnostics for KNX PL-Link bus	192.168.1.20		123000	MV	19	4
✓	PinkBusMgmt	AS01		PinkBusPinkBusMgmt	AS01	KNX PL-Link bus management	192.168.1.20		123000	MV	19	4

Abbildung 9: Inputfeld

Das Funktionsfeld ist dauerhaft grau.

Funktionsfeld

Teil 1

Der erste Teil befindet sich der Status und deren Information sowie das heutige Datum inklusive der aktuellen Uhrzeit. Ein zusätzlicher Link verweist auf die Tabelle „Overview“.

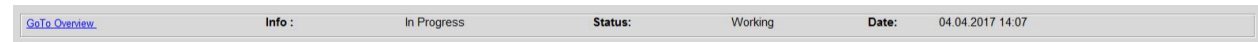


Abbildung 10: Funktionsfeld Teil 1

Teil 2

Im zweiten Teil befinden sich die Überprüfungen von „Alias“ und „Objects“. Ebenfalls zu finden ist die „Modify“-Funktion.

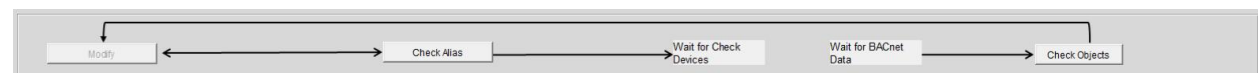


Abbildung 11: Funktionsfeld Teil 2

Teil 3

Der dritte Teil heisst „Append selected Objects“, hier kann man aussuchen, von wo man die Daten der Objekte, die man hinzugefügt haben möchte, kommt. Entweder vom Test oder von EDE. Für die IPA ist jedoch nur der Button „from EDE“ von Wichtigkeit.



Abbildung 12: Funktionsfeld Teil 3

Teil 4

Der Name vom vierten Teil ist „Modify selected lines“. Hier hat man die drei Optionen „Obj > Alias“, „Alias > Obj“ und „SDU > Comment“. Der Button „SDU > Comment“ ist nicht IPA relevant. Es werden Objekt- oder Aliasnamen generiert.



Abbildung 13: Funktionsfeld Teil 4

Teil 5

Der fünfte lautet „BACnet Data“. Hier existiert der Button „from EDE“, welche die BACnet Daten aktualisiert.

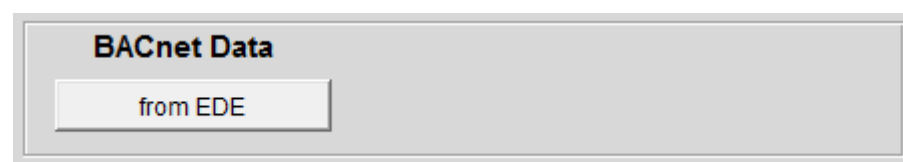


Abbildung 14: Funktionsfeld Teil 5

Teil 6

Im sechsten und auch letzten Teil vom Funktionsfeld befinden sich die Titel für die BACnet Objekte unter anderem befinden sich auch die Buttons „Select“ und „Deselect“.

Abbildung 15: Funktionsfeld Teil 6

Abbildung 16: Funktionsfeld Teil 6.1

Abbildung 17: Funktionsfeld Teil 6.2

Abbildung 18: Funktionsfeld Teil 6.3

Die Titel nach der Reihe von links nach rechts:

- Select/Deselect
- Alias Name
- Alias Controller
- Init value
- Objectname
- Controllername
- Description
- IP-Addr/Node-ID
- Network-No
- Dev-Inst
- Typename
- Type
- Instance

2.2.2 Informationen zur Bedienoberfläche

2.2.2.1 Funktionen

Buttonname (Schaltfläche)	Objektname	Funktion	Wo?
Modify	cmd_ModifyObj	ModifyObj()	Module: mdl_SpecObjects
Check Alias	cmd_CheckAlias	CheckAlais()	Module: mdl_SpecObjects
Check Objects	cmd_CheckObjects	CheckObjects()	Module: mdl_SpecObjects
Append selected Objects: from EDE	cmd_AppendFromEDE	AppendFromEDE()	Module: mdl_SpecObjects
		cmd_Append_Click()	Formulare: frmAppendFromEDE
		cmd_Cancel_Click()	
Obj > Alias	cmd_ObjToAlias	ObjToAlias()	Module: mdl_SpecObjects
		cmd_OverwriteAliasn_Click()	Formulare: frmObjToAlias
		cmd_CancelOtA_Click()	
Alias > Obj	cmd_AliasToObj	AliasToObjName()	Module: mdl_SpecObjects
		cmd_OverwriteObjn_Click()	Formulare: frmAliasToObj
		cmd_CancelAtO_Click()	
BACnet Data from EDE	cmd_BACnetFromEDE	BACnetDataEDE()	Module: mdl_SpecObjects
[✓]	cmd_Select	cmd_Select_Click	Tabelle 16 (Spec- Objects)
[x]	cmd_Deselect	cmd_Deselect_Click	Tabelle 1 (Spec- Objects)

Tabelle 7: Funktionen

2.2.2.2 Felder

Definierte Bereichsnamen	Zellenbereichen	Beschreibung
ctl_InfoObj	=‘Spec-Objects’!\$E\$3	Die letzte detaillierte Status- oder Fehlerinformationen von allen aus diesem Blatt genannten Funktionen werden gezeigt
ctl_StatusObj	=‘Spec-Objects’!\$G\$3	Zeigt den Status der zuletzt ausgeführten Funktion
ctl_DateObj	=‘Spec-Objects’!\$I\$3	Zeigt das Datum der zuletzt ausgeführten Funktion
ctl_FunctionFields	=‘Spec-Objects’!\$A\$1:\$M\$12	Das Funktionsfeld ist durchgehend grau, sie unterscheidet zwischen Funktionsfeld und Inputfeld, ebenfalls sind hier Active-X Steuerelemente vorhanden
ctl_InputFields	=‘Spec-Objects’!\$A\$13:\$M\$95	Das Inputfeld ist weiss, wenn das die Tabelle nicht schreibgeschützt ist, sonst ist sie grau markiert

Tabelle 8: Definierte Felder

2.2.3 Entwicklungsumgebung VBA (Editor)

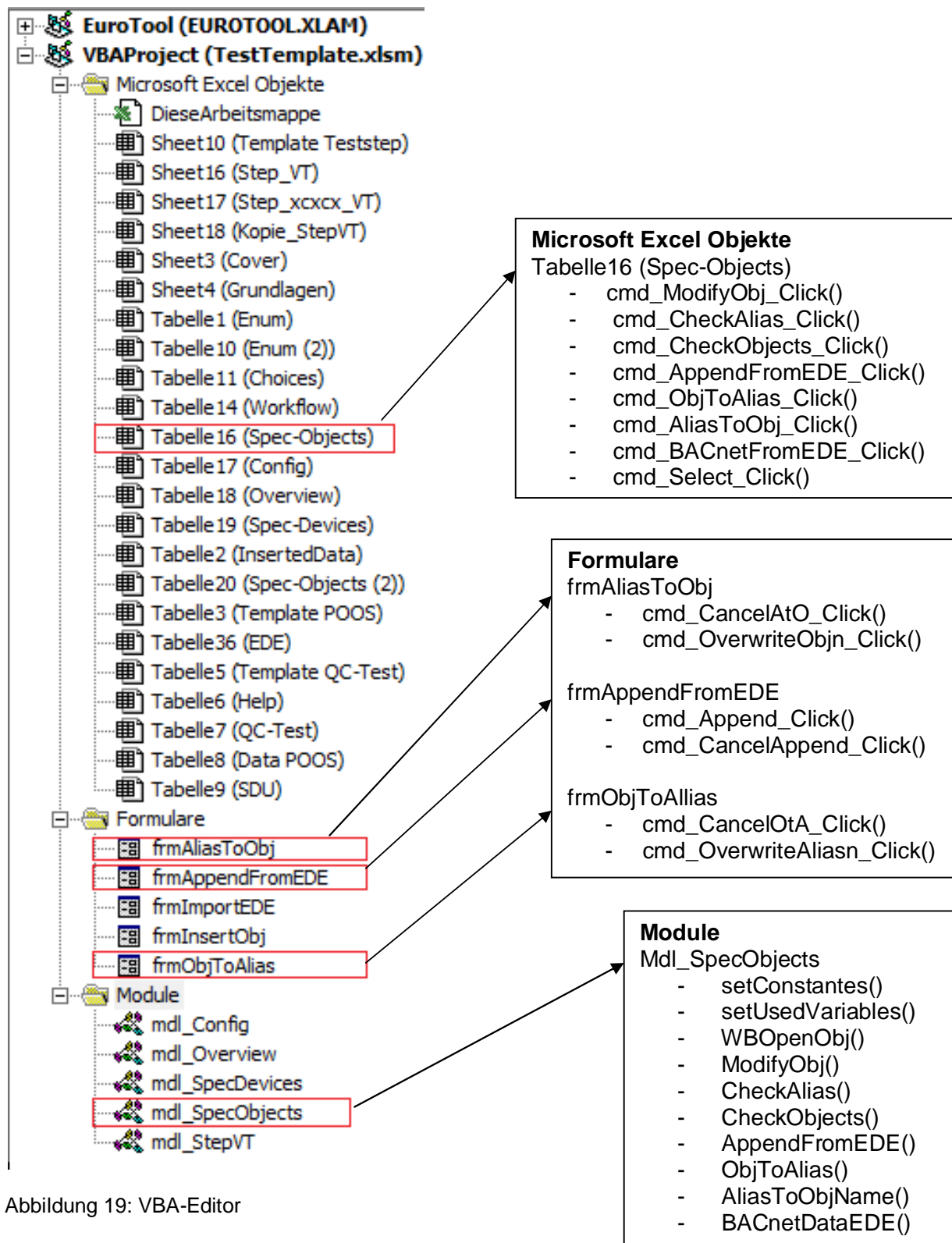


Abbildung 19: VBA-Editor

2.2.4 Struktogramme

Struktogramme sind grafische Darstellungen von Programmlogik für Aufgaben und Lösungen. Um vor dem eigentlichen Programmieren einen Überblick zu bekommen, können Programmierer mit Hilfe von einem Struktogramm erkennen, wie das Programm aufgebaut ist oder aufgebaut sein soll. Ebenso lässt sich herausfinden, wie die Abläufe sein sollen, wo die einzelne Schleifen und die Bedingungen sind.

Für folgende Funktionen wurde ein Struktogramm erstellt:

- AppendFromEDE
- ObjToAlias
- AliasToObjectname
- BACnetDataEDE
- CheckObjects
- CheckAlias

Was man erwähnen muss, ist dass die Funktion „CheckAlias“ bereits als Programmcode existiert, aber nicht weiter nicht gross bearbeitet wurde. Der letzte Teil wurde gemäss Struktogramm implementiert. Das Struktogramm zeigt lediglich die Vorstellung und das Vorgehen der IPA-Ausführende und kann nicht mit dem tatsächlichen Vorgehen im Programmcode verglichen werden.

2.2.4.1 Erklärungen zum Struktogramm

„x“ wird hier als Platzhalter verwendet.


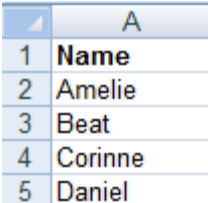

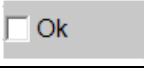
Bezeichnung	Erklärung
=	ist Bsp: $x = y \rightarrow x$ ist y
Sheet(„X“)	Die Tabelle „x“ im Excel Bsp: Sheet(„Info“) \rightarrow Tabelle „Info“ im Excel 
<x>	Variable oder Active-X Steuerelemente Bsp: <var> \rightarrow die Variable heisst „var“
[x]	Range bzw. Bereich im Excel Bps: [Name] \rightarrow Der Bereich „Name“ (A2:A5) im Excel 
btn	Button Bsp: <btnClick> 
chk	Checkbox Bsp: <chkOk> 
{}	Array Bsp: Name{} = Name{Name1, Name2}
‘Select’	✓

Tabelle 9: Erklärung zum Struktogramm

2.2.4.2 Abbildung der Struktogramme

2.2.4.2.1 AppendFromEDE

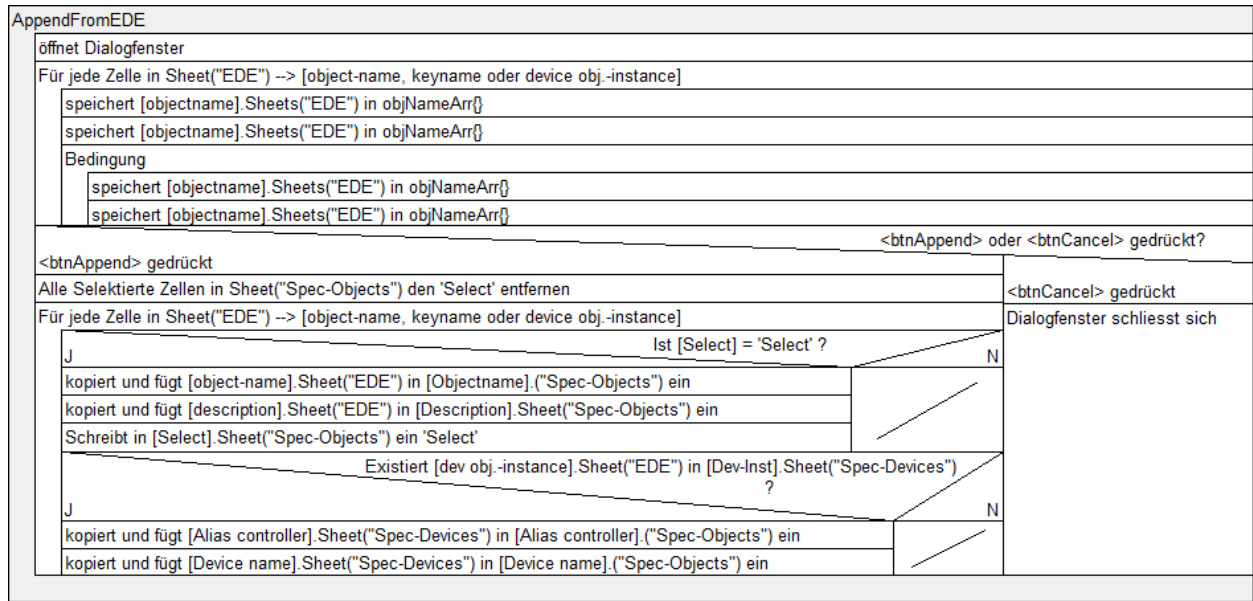


Abbildung 20: Struktogramm AppendFromEDE

2.2.4.2.2 ObjToAlias

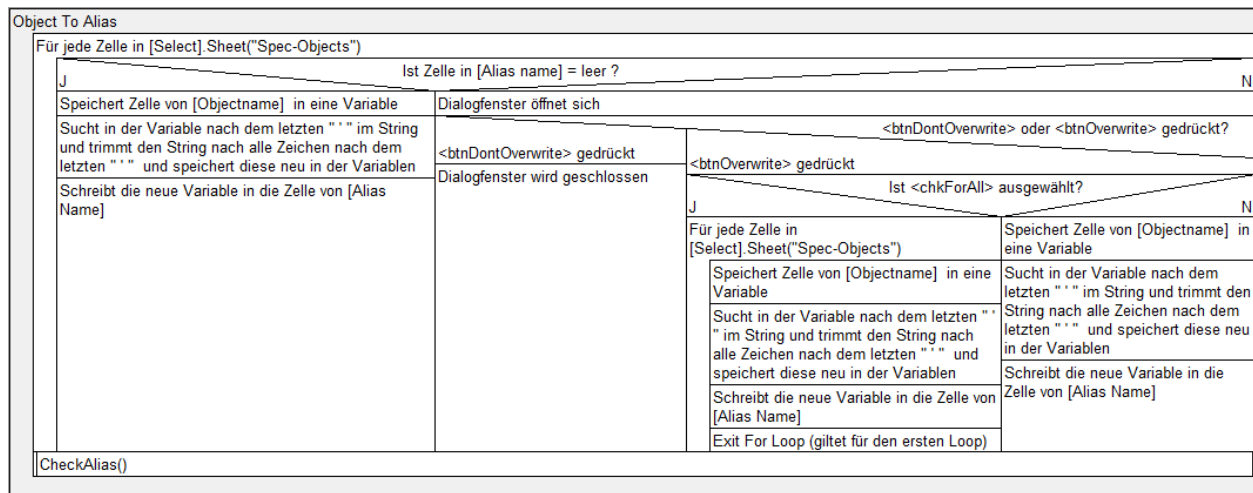


Abbildung 21: Struktogramm ObjToAlias

2.2.4.2.3 AliasToObjectname

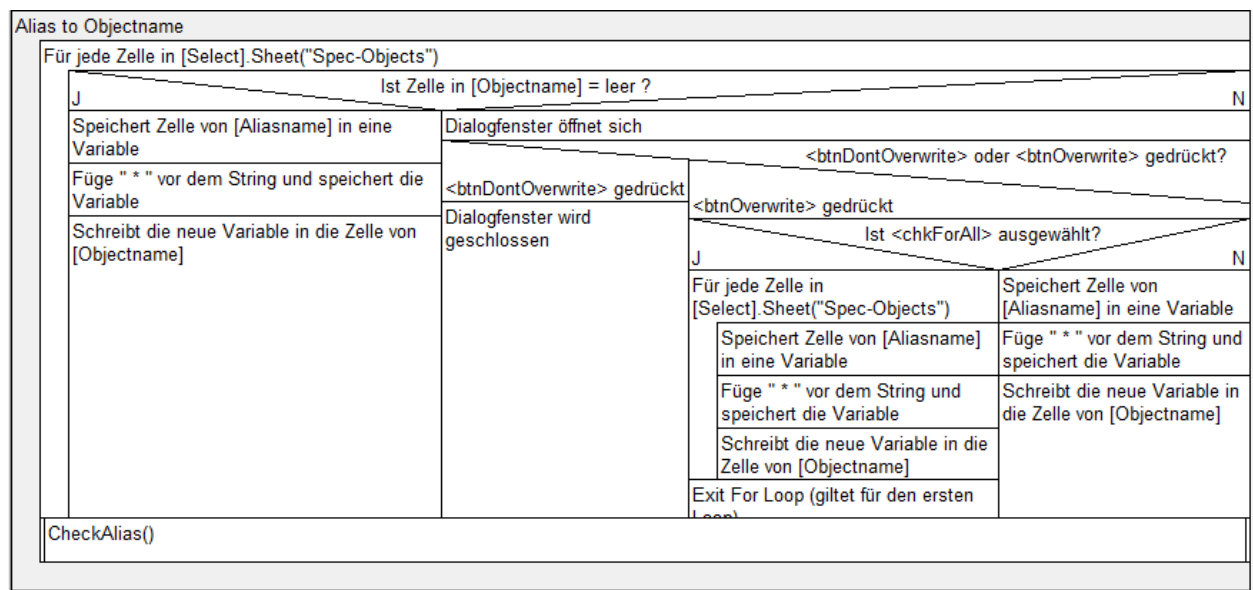


Abbildung 22: Struktogramm AliasToObj

2.2.4.2.4 BACnetDataEDE

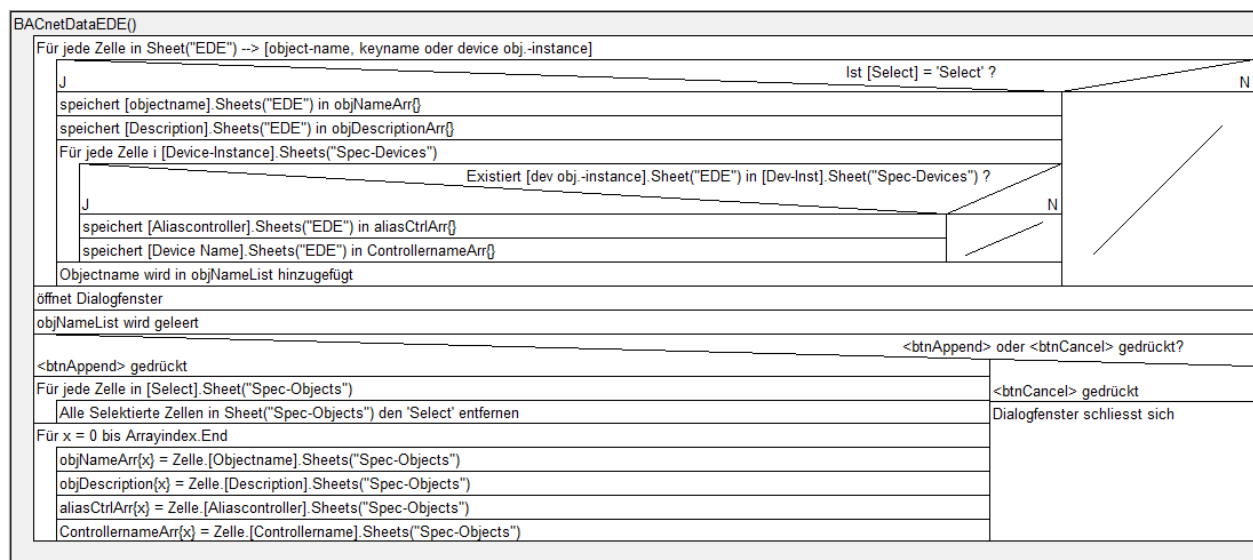


Abbildung 23: Struktogramm BACnetDataEDE

2.2.4.2.5 CheckObjects

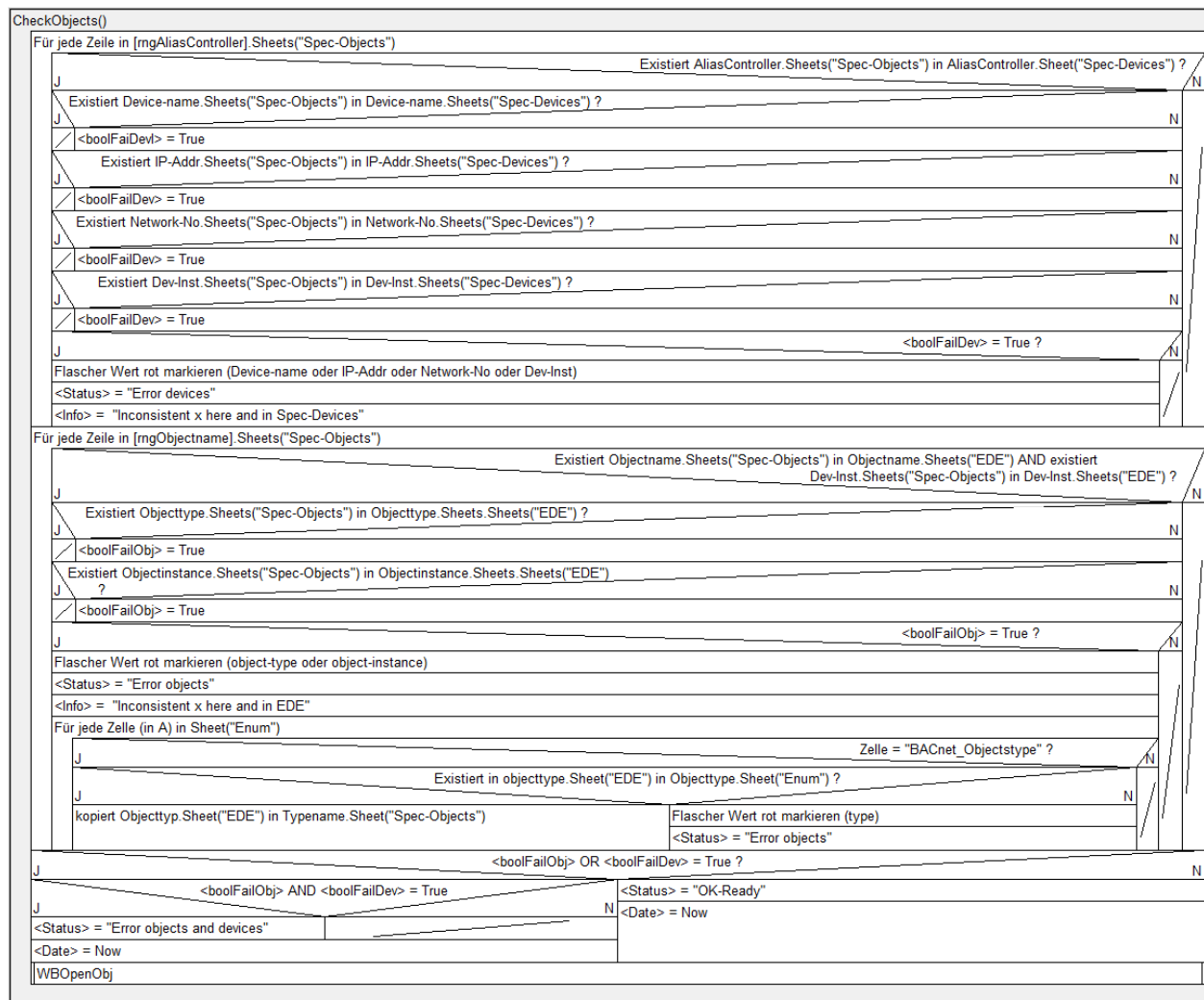


Abbildung 24: Struktogramm CheckObjects

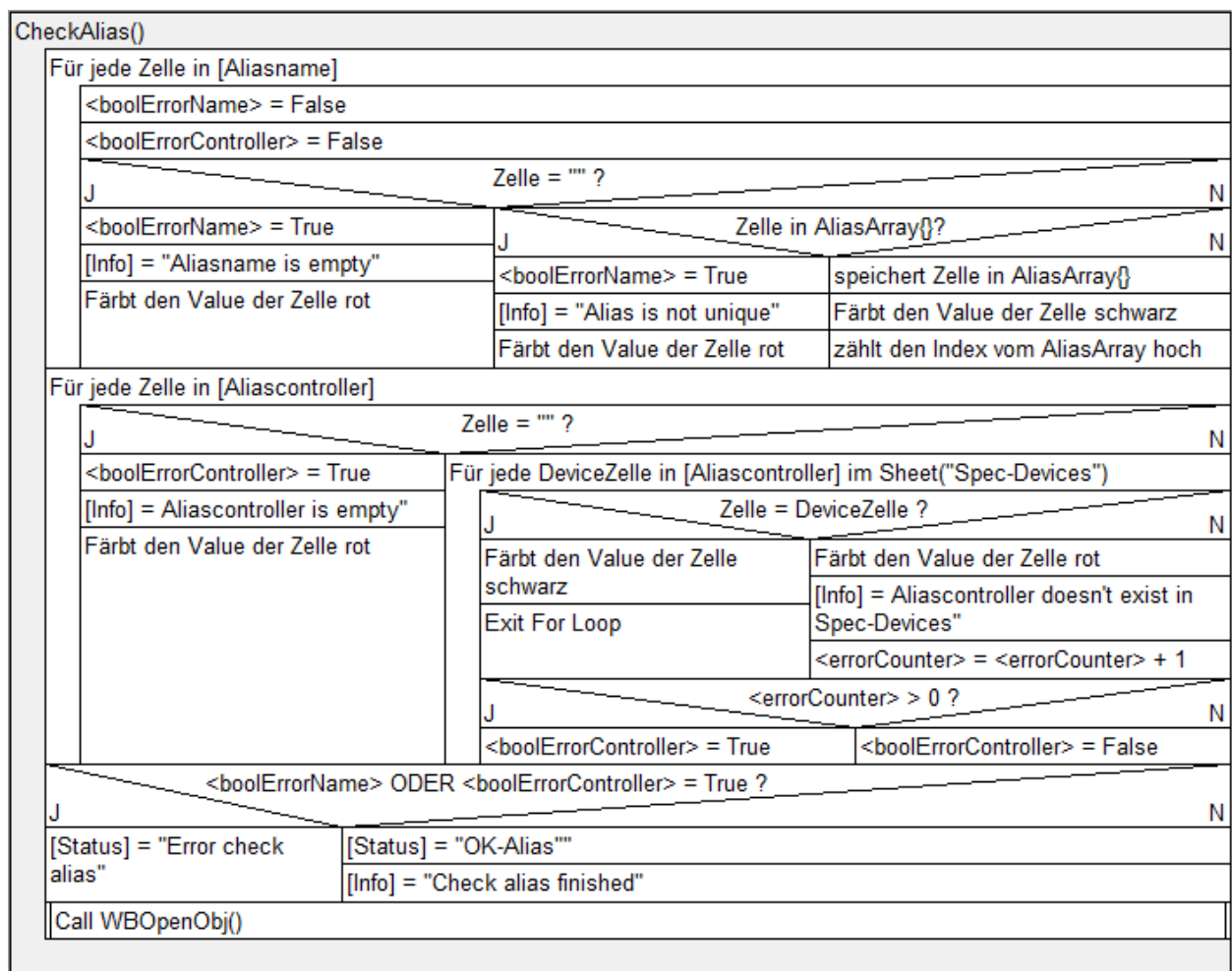
2.2.4.2.6 *CheckAlias*

Abbildung 25: Struktogramm CheckAlias

2.2.5 Beschreibungen der Funktionen ohne Struktogramm

Der Grund warum nicht alle Funktionen ein Struktogramm haben, ist der, dass diese wegen der Komplexität keines benötigen.

Folgende Funktionen brauchen kein Struktogramm:

- setConstantes
- setUsedVariables
- setObjects
- WBOpenObj
- ModifyObj
- Select
- Deselect

2.2.5.1 Funktionsbeschreibung

Funktionsname	Funktionsbeschreibung
setConstantes()	Globale Konstanten, hier im VBA sind es Variablen, werden in dieser Funktion initialisiert. So kann man, falls nötig, vorwiegend die Reihen und Spalten festlegen. (Column und Row)
setUsedVariables()	Globale Variablen werden hier initialisiert. Zurzeit werden in dieser Funktion die Bereiche, die für die Funktionen relevant sind, initialisiert. (Range) Ebenfalls werden die benötigten Tabellen initialisiert. (Worksheet)
setObjects()	Globale Variablen werden in dieser Funktion ebenfalls initialisiert. Diese werden zu den OLEObjects zugeordnet.
WBOpenObj()	Deaktiviert alle Buttons, ausser „Modify“, der Benutzeroberfläche. Ändert die Farbe vom Inputfeld zu grau und schützt die Tabelle. (Schreibgeschützt)
ModifyObj()	Aktiviert alle Buttons, ausser „Modify“, der Benutzeroberfläche. Ändert die Farbe vom Inputfeld zu weiss und setzt die Tabelle auf nicht schreibgeschützt.
Select	Selektiert die ausgewählten Objekte. Schreibt ein ü in die vorhergesehene Zelle.
Deselect	Deselektiert die ausgewählten Objekte. Die Zelle wird leer.

Tabelle 10: Funktionsbeschreibung [ohne Struktogramm]

2.2.6 Implementierung

Nachfolgend werden alle komplexeren Funktionen, für welches ein Struktogramm erstellt wurde, beschrieben. Die Funktion „CheckAlias“ wird nicht beschrieben, da nur einen Teil an Eigenarbeit besteht. Insgesamt sind es fünf Funktionen die hier erläutert werden.

2.2.6.1 AppendFromEDE()

Diese Funktion wird durch das Klicken von Append selected Objects „from EDE“-Button aufgerufen. Sie schaut in der Tabelle „EDE“ nach den ausgewählten Objekten und speichert für jedes einzelne Objekt die Daten von „Alias Controller“, „Objectname“, „Controllername“ und „Description“. Jedoch werden zunächst nur die Daten von „Objectname“ verwendet. Diese wird in die Liste von der User Form „frmAppendFromEDE“ gespeichert. Danach öffnet sich diese User Form sich und zeigt die ausgewählten Objekte an. Man hat zwei Optionen zur Auswahl, nämlich „Cancel“ oder „Append“. Nachdem Ausführen von „Append“ werden die gespeicherte Daten in die Tabelle „Spec-Objects“ hinzugefügt. Zusätzlich werden die neuen Objekte selektiert. Danach wird die Funktion „ObjToAlias“ aufgerufen.

Button:

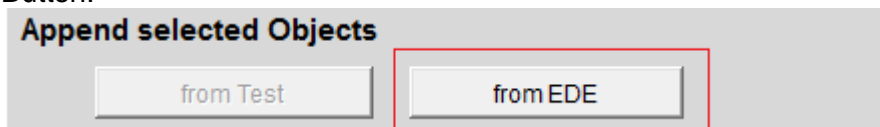


Abbildung 26: AppendFromEDE Button

User Form:

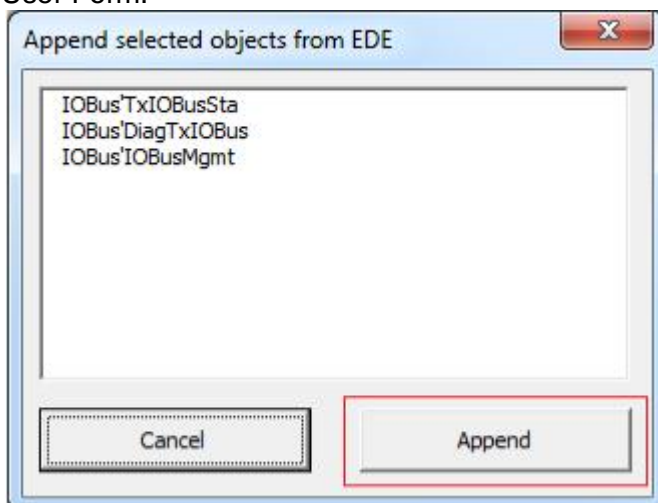


Abbildung 27: AppendFromEDE User Form

Vor der Ausführung:

✓	x	Alias Name	Alias Controller	Init value	Objectname	Controllername	Description
		CmdA	AS01		B01F1r01RSegm01AF01CmdA	AS01	Command analog

Abbildung 28: AppendFromEDE vorher

Nach der Ausführung:

✓	x	Alias Name	Alias Controller	Init value	Objectname	Controllername	Description
		CmdA	AS01		B01F1r01RSegm01AF01CmdA	AS01	Command analog
✓		TxIOBusSta	AS01		IOBusTxIOBusSta	AS01	TX-I/O bus state
✓		DiagTxIOBus	AS01		IOBusDiagTxIOBus	AS01	Diagnostics for TX-I/O bus
✓		IOBusMgmt	AS01		IOBusIOBusMgmt	AS01	I/O bus management

Abbildung 29: AppendFromEDE nachher

2.2.6.2 *ObjToAlias()*

Durch das Klicken vom „Obj > Alias“-Button wird diese Funktion aufgerufen. Für jedes ausgewählte Objekt wird geschaut, ob die Spalte mit „Alias Name“ leer ist oder nicht. Wenn sie leer ist, wird hier der „Objectname“ des Objektes auf den String nach dem letzten Apostroph gekürzt, gespeichert und danach in die Spalte „Alias Name“ vom Objekt hinzugefügt. Falls die Spalte „Alias Name“ vom Objekt nicht leer ist, so wird eine User Form geöffnet. Diese fragt für jedes ausgewählte Objekt nach, ob man sie überschreiben möchten. Zusätzlich befindet sich eine Checkbox, die man anwählen kann, wenn man für jedes überschreiben oder nicht überschreiben möchte. Wenn diese Checkbox ausgewählt wurde, so wird das Verfahren, welches man mittels „Overwrite“-Button und „Don't overwrite“-Button auswählen kann, für jedes ausgewählte Objekt angewendet. Beim Betätigen vom „Overwrite“-Button wird der „Alias Name“ überschrieben. Es wird nichts überschrieben, wenn der „Don't overwrite“-Button gedrückt wurde.

Button:



Abbildung 30: ObjToAlias Button

So sieht die User Form aus:

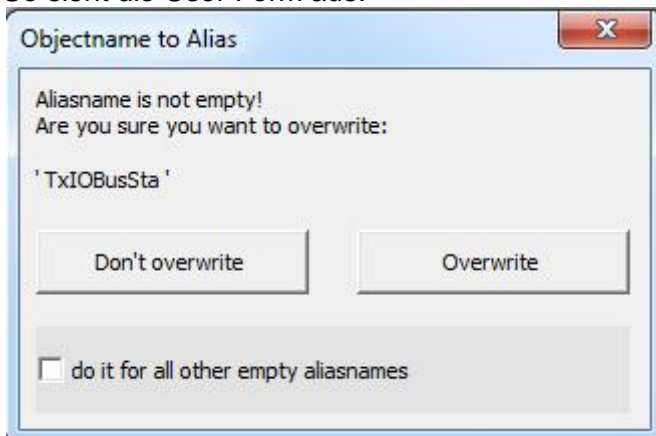


Abbildung 31: ObjToAlias User Form

Das Kürzen von „Objectname“:

Alias Name	Alias Controller	Init value	Objectname
CmdA	AS01		B01'Flr01'RSegm01'AF01'CmdA
TxIOBusSta	AS01		IOBusTxIOBusSta
DiagTxIOBus	AS01		IOBusDiagTxIOBus
IOBusMgmt	AS01		IOBusIOBusMgmt

Abbildung 32: ObjToAlias Ergebnis

2.2.6.3 *AliasToObjName()*

Durch das Klicken vom „Alias > Obj“-Button wird diese Funktion aufgerufen. Für jedes ausgewählte Objekt wird geschaut, ob die Spalte mit „Objectname“ leer ist oder nicht. Falls die Spalte leer ist, so speichert sie den „Alias Name“ und fügt ein Stern (*) an den Anfang von diesem String, diese wird gespeichert und dann in die Spalte „Objectname“ hinzugefügt. Falls die Spalte „Objectname“ vom ausgewählten Objekt nicht leer ist, so erscheint eine User Form. Diese fragt den Benutzer, für jedes ausgewählte Objekt, ob er den „Objectname“ überschreiben möchte oder nicht. Eine Checkbox ist anzuwählen, wenn der Benutzer für alle ausgewählte Objekten das gleiche Verfahren möchte. Falls diese Checkbox angewählt wurde, so wird sie für jedes Objekt das gleiche machen, ohne dass für jedes Objekt eine User Form erscheint.

Button:



Abbildung 33: AliasToObj Button

So sieht die User Form aus:

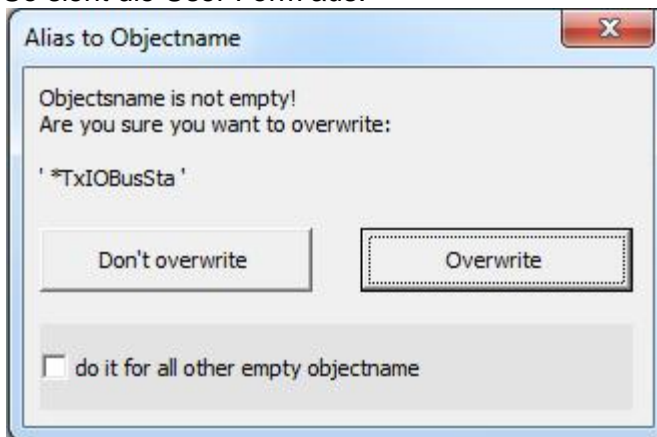


Abbildung 34: AliasToObj User Form

Der generierte „Objectname“:

Alias Name	Alias Controller	Init value	Objectname
CmdA	AS01		B01F1r01'RSegm01'AF01'CmdA
TxIOBusSta	AS01		*TxIOBusSta
DiagTxIOBus	AS01		*DiagTxIOBus
IOBusMgmt	AS01		*IOBusMgmt

Abbildung 35: AliasToObj Ergebnis

2.2.6.4 BACnetDataEDE()

Beim Klicken vom BACnet Data „from EDE“-Button wird die Funktion aufgerufen. Sie führt zuerst die Funktion „CheckAlias“ aus, falls diese Fehl schlägt, so bricht sie die Funktion ab. Wenn alles in Ordnung ist, fragt sie mit einer Messagebox nach, ob man sich sicher ist, ob man alle BACnet Daten löschen möchte. Bei einem „Nein“ bricht sie die Funktion ab. Bei einem „Ja“ löscht sie die Daten Controllername, IP-Addr, Network-No, Dev-Inst, Typename, Type und Instance. Nach dem alle gelöscht worden sind, fügt sie die neue Daten, aus der Tabelle Spec-Devices und EDE, in die Tabelle Spec-Objects hinzu.

Button:

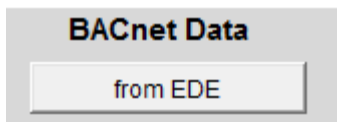


Abbildung 36: BACnet Data Button

So sieht die User Form aus:

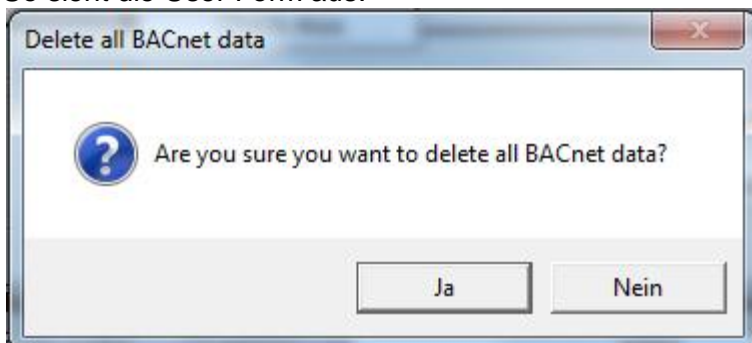


Abbildung 37: BACnet Data User Form

Das Löschen der Daten:

Objectname	Controllername	Description	IP - Addr/Node-ID	Network- No	Dev-Inst	Typename	Type	Instance
B01FIR01RSegm01AF01CmdA		Command analog						
B01FIR01RSegm01AF01Valln4		Value input 4						
B01FIR01RSegm01AF01PrVal		Present value						
B01FIR01RSegm01AF01Valid		Valid						

Abbildung 38: BACnet Data Löschen

Das Hinzufügen der Daten:

Objectname	Controllername	Description	IP - Addr/Node-ID	Network- No	Dev-Inst	Typename	Type	Instance
B01FIR01RSegm01AF01CmdA	AS01	Command analog	192.168.1.20		123000	AV	2	5
B01FIR01RSegm01AF01Valln4	AS01	Value input 4	192.168.1.20		123000	AV	2	9
B01FIR01RSegm01AF01PrVal	AS01	Present value	192.168.1.20		123000	AV	2	10
B01FIR01RSegm01AF01Valid	AS01	Valid	192.168.1.20		123000	BV	5	1

Abbildung 39: BACnet Data Hinzufügen

2.2.6.5 *CheckObjects()*

Diese Funktion wird durch den „Check Objects“-Buttons aufgerufen. Als erstes führt es die CheckAlias-Funktion aus, falls diese fehl schlägt, so bricht sie die Funktion ab. Falls nicht, dann kontrolliert sie die Daten Device-Name, IP-Addr, Network-No, Dev-Inst, Type, Instance und Typename, ob sie richtig sind. Falls eines dieser Daten nicht mit den Daten von der Tabelle „Spec-Devices“ oder „EDE“ übereinstimmt, so zeigt es einen Fehler an. Dazu gibt es zwei Fehlertypen, nämlich „Error devices“ oder „Error objects“.

Grund für die nicht Vollständigkeit

Die Ausgaben vom Status und Info werden nicht korrekt dargestellt. Dieser Fehler wurde beim White-Box-Test aufgedeckt und konnte noch nicht behoben werden.

Aufgrund mangelnde VBA-Kenntnisse und einer vorgegebene Zeit konnte der Fehler nicht beheben. Da der IPA-Bericht höchste Priorität hat, wurde entschieden, die Fehler so zu lassen und erst nach der IPA zu beheben.

2.3 Kontrolle

2.3.1 Testkonzept

2.3.1.1 Testumgebung

Die Testdurchführung findet in der folgenden Entwicklungsumgebung, ausgelesen aus dem Microsoft Hilfsprogramm „msinfo32“, statt:

Element	Wert
Betriebssystemname	Microsoft Windows 7 Enterprise
Version	6.1.7601 Service Pack 1 Build 7601
Zusätzliche Betriebssystembeschränkungen	Nicht verfügbar
Betriebssystemhersteller	Microsoft Corporation
Systemname	MD1K3VVC
Systemhersteller	FUJITSU
Systemmodell	LIFEBOOK E736
Systemtyp	x64-basierter PC
Prozessor	Intel(R) Core(TM) i5-6300U CPU @ 2.40GHz, 2401 MHz, 2 Kern(e), 4 logische(r) Prozessor(en)
BIOS-Version/-Datum	FUJITSU // Insyde Software Corp. Version 1.18, 20.05.2016
SMBIOS-Version	2.8
Windows-Verzeichnis	C:\WINDOWS
Systemverzeichnis	C:\WINDOWS\system32
Startgerät	\Device\HarddiskVolume1
Gebietsschema	Vereinigte Staaten von Amerika
Hardwareabstraktionsebene	Version = "6.1.7601.17514"
Benutzername	AD001\z0039cje
Zeitzone	Mitteuropäische Zeit
Installierter physikalischer Speicher	8.00 GB
Gesamter realer Speicher	7.91 GB
Verfügbarer realer Speicher	3.63 GB
Gesamter virtueller Speicher	15.8 GB
Verfügbarer virtueller Speicher	11.0 GB
Größe der Auslagerungsdatei	7.91 GB
Auslagerungsdatei	C:\pagefile.sys

Abbildung 40: Testumgebung

2.3.1.2 Testdaten

Testdaten sind in der Datei „TestTemplate.xlsx“ vorhanden. Grundsätzlich beinhaltet die Datei alle Daten, mit der man die Funktionen testen kann.

Die Daten werden aus den Tabellen „Spec-Objects“, „Spec-Devices“, „EDE“ und „Enum“ benutzt.

2.3.1.3 Testablauf

Innerhalb der IPA-Vorarbeit entwickelte Module oder angepasste Funktionen werden beim Testen überprüft. Falls Codestellen von den Änderungen betroffen sind, werden im Visual Basic Editor Tests ausgeführt oder allenfalls angepasst.

Das Testen besteht aus zwei Komponenten:

- White-Box-Test
- Akzeptanztest

White – Box – Testfälle

Die einzelnen Testfälle wurden innerhalb der IPA von der IPA-Ausführenden ermittelt. Um die neuen Funktionen genau zu testen und dabei auch alle erstellten Funktionen gegen Fehlbedienungen und fehlerhafte Daten und Dateien zu überprüfen, werden White Box Test verwendet. Diese Tests werden während der Realisierungsphase durchgeführt, werden auch Fehlerfälle genau getestet. Bei Bedarf kann man im Visual Basic Editor mit verschiedenen Möglichkeiten vom Debuggen profitieren.

Im Vorfeld muss man erwähnen, dass nicht für alle Funktionen ein White-Box-Test stattfindet. Für diese Funktionen wurden keine White-Box-Tests durchgeführt, da sie nicht sinnvoll ist, aufgrund der Komplexität der Funktionen. Sie werden mittels Akzeptanztest schon ausreichend getestet.

- setConstantes
- setUsedVariables
- setObjects
- WBOpen
- Modify
- Select
- Deselect

Eine Funktion, die nicht getestet wird ist „CheckAlias“, da der Check schon implementiert wurde und aus Zeitgründen mit Absicht für den White-Box-Test ausgelassen wird.

Für die Funktion „AliasToObjName“ wird ebenfalls mit Absicht ausgelassen. Ebenfalls aus Zeitgründen und der Komplexität der Funktion. Dem Entscheid nach, reicht auch hier nur ein Akzeptanztest.

Akzeptanztestfälle

Die Testfälle wurden in der Entscheidungsphase anhand der detaillierten Aufgabenstellung ermittelt und in der Kontrollphase von einem Siemens Mitarbeiter, der vor der IPA schon eine kleine Einführung bekommen hatte, durchgeführt. Urs Heimann, die Testperson, wird ein Testprotokoll erhalten, das sie während der Durchführung von einzelnen Testfällen ausfüllen soll.

Nur die IPA relevante Tabelle „Spec-Objects“, mit den folgenden Buttons, wird mit den Testfällen abgedeckt:

- Modify
- Check Alias
- Check Objects
- Append selected Objects: from EDE
- Obj > Alias
- Alias > Obj
- BACnet Data: from EDE
- Select
- Deselect

Die folgenden Funktionen der Buttons sind nicht IPA relevant und wurden somit nicht mit Testfällen abgedeckt. Zusätzlich wurden sie deaktiviert.

- Append selected Objects: from Test
- SDU>Comment

2.3.2 White-Box-Test

2.3.2.1 Testfälle

1 AppendFromEDE

Test ID : 1.0	
Name	Arrays erstellen aus den Tabellen Spec-Objects und Spec-Devices
Testvoraussetzungen	<ul style="list-style-type: none"> - Tabelle „Spec-Objects“ ist geöffnet - Tabelle „EDE“ muss vorhanden sein - Modify bereits gedrückt, Tabelle ist nicht schreibgeschützt - Bei Append selected Objects muss der Button „from EDE“ aktiviert sein - Haltepunkt bei <i>objNameArr(i) = [1]</i> , <i>objDescription(i) = [2]</i>, <i>devAliasCtrlArr(i) = [3]</i> und <i>devDevNameArr(i) = [4]</i> setzen - Haltepunkt[5] bei <i>frmAppendFromEDE.Show</i> setzen - <i>objNameArr</i>, <i>objDescriptionArr</i>, <i>devAliasCtrlArr</i>, <i>devDevNameArr</i> und die Variable <i>i</i> zur Überwachung hinzufügen
Testablauf	<ol style="list-style-type: none"> 1. Zur Tabelle „EDE“ gehen 2. Gewünschte Objekte auswählen 3. Zur Tabelle „Spec-Objects“ gehen 4. Button „from EDE“ drücken 5. Hält bei Haltepunkt [1] an 6. Schrittt „F5“ bis Haltepunkt [5] erreicht 7. Überwachungsausdrücke anschauen und mit erwarteten Resultat vergleichen
Erwartetes Resultat	<ul style="list-style-type: none"> - In allen Arrays die gleiche Anzahl Daten im Array vorhanden – Objekte in den Arrays erstellt, Index = 0 bis Anzahl ausgewählte Objekte in „EDE“ - <i>i</i> = Anzahl ausgewählte Objekte in „EDE“

Test ID : 1.1	
Name	Einträge aus dem Array in die Zeilen der Tabelle füllen
Testvoraussetzungen	<ul style="list-style-type: none"> - Tabelle „Spec-Objects“ ist geöffnet - Tabelle „EDE“ muss vorhanden sein - Modify bereits gedrückt, Tabelle ist nicht schreibgeschützt - Bei Append selected Objects muss der Button „from EDE“ aktiviert sein - Haltepunkt [1] beim 1. <i>For x = 0 to i – 1</i> setzen - objNameArr, objDescriptionArr, devAliasCtrlArr und devDevNameArr zur Überwachung hinzufügen
Testablauf	<ol style="list-style-type: none"> 1. Zur Tabelle „EDE“ gehen 2. Gewünschte Objekte auswählen 3. Zur Tabelle „Spec-Objects“ gehen 4. Button „from EDE“ drücken 5. User Form erscheint, Button „Append“ drücken 6. Hält bei Haltepunkt [1] an 7. Überwachungsausdrücke anschauen und mit den Einträgen in den Tabellen vergleichen 8. Einzelschritt „F8“, Schritt 5 bis 7 wiederholen bis das höchste Arrayindex (Anzahl der ausgewählten Objekt in EDE) erreicht worden ist
Erwartetes Resultat	<ul style="list-style-type: none"> - Alle ausgewählte Objekte sind mit Objectname, Description, Alias Controller und Controllernamen in der Tabelle nun ersichtlich - Bei den ausgewählten Objekten ist ein ü beim Select-Bereich markiert

2 ObjToAlias

Test ID : 2.0	
Name	Checkbox gewählt und Button Overwrite
Testvoraussetzungen	<ul style="list-style-type: none"> - Tabelle „Spec-Objects“ ist geöffnet - Modify bereits gedrückt, Tabelle ist nicht schreibgeschützt - Haltepunkt [1] bei IF checkAll = True gesetzt - checkAll und frmObjToAlias.ObjOK zur Überwachung hinzufügen
Testablauf	<ol style="list-style-type: none"> 1. Objekte mit Alias Name mit Ü markieren/auswählen 2. Button „Obj > Alias“ drücken 3. Checkbox im Useform frmObjToAlias ankreuzen 4. Button „Overwrite“ drücken 5. Hält bei Haltepunkt [1] an 6. Überwachungsausdrücke anschauen und vergleichen 7. Schritt „F5“
Erwartetes Resultat	<ul style="list-style-type: none"> - frmObjToAlias.ObjOK ist True - checkAll ist True - Beendet die For-Schleife → User Form wird nicht für jedes Objekt immer wieder angezeigt - Alle Objekte sind überschrieben worden
Test ID : 2.1	
Name	Checkbox gewählt und Button Overwrite
Testvoraussetzungen	<ul style="list-style-type: none"> - Tabelle „Spec-Objects“ ist geöffnet - Modify bereits gedrückt, Tabelle ist nicht schreibgeschützt - Haltepunkt [1] bei IF checkAll = True gesetzt - checkAll und frmObjToAlias.ObjOK zur Überwachung hinzufügen
Testablauf	<ol style="list-style-type: none"> 1. Objekte mit Alias Name mit Ü markieren/auswählen 2. Button „Obj > Alias“ drücken 3. Checkbox im Useform frmObjToAlias ankreuzen 4. Button „Overwrite“ drücken 5. Hält bei Haltepunkt [1] an 6. Schritt „F5“ 7. Überwachungsausdrücke anschauen und vergleichen
Erwartetes Resultat	<ul style="list-style-type: none"> - frmObjToAlias.ObjOK ist False - checkAll ist True - Beendet die For-Schleife → User Form wird nicht für jedes Objekt immer wieder angezeigt - Alle Objekte sind unverändert

3 BACnetDataEDE

Test ID : 3.0	
Name	Abbruch wenn CheckAlias fehlgeschlagen ist
Testvoraussetzungen	<ul style="list-style-type: none"> - Tabelle „Spec-Objects“ ist geöffnet - Modify bereits gedrückt, Tabelle ist nicht schreibgeschützt - Haltepunkt [1] bei IF [ctl_StatusObj]. Value <> „Error“ gesetzt - [ctl_StatusObj].Value zur Überwachung hinzufügen - CheckAlias ist fehlgeschlagen (Aliasname doppel oder
Testablauf	<ol style="list-style-type: none"> 1. In BACnet data Button „from EDE“ drücken 2. Hält bei Haltepunkt [1] an 3. Überwachungsausdrücke anschauen und vergleichen 4. Schritt „F5“
Erwartetes Resultat	<ul style="list-style-type: none"> - [ctl_StatusObj].Value beinhaltet „Error“ - Tabelle ist schreibgeschützt - Es wurde nichts geändert - BACnet data abgebrochen und fehlgeschlagen
Test ID : 3.1	
Name	Vor dem Aktualisieren die Daten löschen
Testvoraussetzungen	<ul style="list-style-type: none"> - Tabelle „Spec-Objects“ ist geöffnet - Modify bereits gedrückt, Tabelle ist nicht schreibgeschützt - Haltepunkt [1] bei <i>Next</i> (von <i>For Each objCell In rngAliasCotroller</i>) gesetzt - Haltepunkt [2] bei <i>For Each objCell In rngAliasController</i> gesetzt - Daten sind vorhanden (Controllername, IP-Addr, Dev-Inst, Typname, Type, Instance (+ Network-No)) - CheckAlias ist nicht fehlgeschlagen
Testablauf	<ol style="list-style-type: none"> 1. MessageBox erscheint 2. Benutzer klick „Yes“ 3. Hält bei Haltepunkt [1] an 4. Benutzer öffnet Tabelle „Spec-Objects“ 5. Benutzer kontrolliert die Daten 6. Schritt „F5“ solange bis Haltepunkt [2] erreicht 7. Hält bei Haltepunkt [2] an 8. Benutzer kontrolliert ob Controllername, IP-Addr, Dev.Inst, Typname, Type und Instance (+ Network-No) leer sind
Erwartetes Resultat	<ul style="list-style-type: none"> - Daten sind nicht vorhanden (leer, gelöscht) (Controllername, IP-Addr, Dev-Inst, Typname, Type, Instance (+ Network-No))

4 CheckObjects

Test ID : 4.0	
Name	Abbruch wenn CheckAlias fehlgeschlagen ist
Testvoraussetzungen	<ul style="list-style-type: none"> - Tabelle „Spec-Objects“ ist geöffnet - Modify bereits gedrückt, Tabelle ist nicht schreibgeschützt - Haltepunkt [1] bei IF [ctl_StatusObj]. Value <> „Error“ gesetzt - [ctl_StatusObj].Value zur Überwachung hinzufügen - CheckAlias ist fehlgeschlagen
Testablauf	<ol style="list-style-type: none"> 1. Button „CheckObjects“ drücken 2. Hält bei Haltepunkt [1] an 3. Überwachungsausdrücke anschauen und vergleichen 4. Schritt „F5“
Erwartetes Resultat	<ul style="list-style-type: none"> - [ctl_StatusObj].Value beinhaltet „Error“ - Tabelle ist schreibgeschützt - Check abgebrochen und fehlgeschlagen
Test ID : 4.1	
Name	boolFailDev ist wahr
Testvoraussetzungen	<ul style="list-style-type: none"> - Tabelle „Spec-Objects“ ist geöffnet - Modify bereits gedrückt, Tabelle ist nicht schreibgeschützt - Haltepunkt [1] bei <i>If boolFailDev = True Then</i> gesetzt - boolFailDev zur Überwachung hinzufügen - Devicename, IP-Addr, Network-No oder Dev-Inst zu einem ungültigen Wert manipulieren - CheckAlias ist nicht fehlgeschlagen
Testablauf	<ul style="list-style-type: none"> - Button „CheckObjects“ drücken - Hält bei Haltepunkt [1] an - Überwachungsausdrücke anschauen und vergleichen - Schritt „F5“ - Tabelle „Spec Objects“ öffnen - Status und Info anschauen
Erwartetes Resultat	<ul style="list-style-type: none"> - Status: „Error devices“ - Info: „Inconsistent (Devicename, IP-Addr, Network-No oder Dev-Inst) here and in Spec-Devices“ - Bearbeitete Zelle deren Inhalt ist rot markiert

Test ID : 4.2	
Name	boolFailObj ist wahr
Testvoraussetzungen	<ul style="list-style-type: none"> - Tabelle „Spec-Objects“ ist geöffnet - Modify bereits gedrückt, Tabelle ist nicht schreibgeschützt - Haltepunkt [1] bei <i>If boolFailObj = True Then</i> gesetzt - boolFailObj zur Überwachung hinzufügen - Type oder Instance zu einem ungültigen Wert manipulieren - CheckAlias ist nicht fehlgeschlagen
Testablauf	<ul style="list-style-type: none"> - Button „CheckObjects“ drücken - Hält bei Haltepunkt [1] an - Überwachungsausdrücke anschauen und vergleichen - Schritt „F5“ - Tabelle „Spec Objects“ öffnen - Status und Info anschauen
Erwartetes Resultat	<ul style="list-style-type: none"> - Status: „Error objects“ - Info: „Inconsistent (Type oder Instance) here and in Spec-Devices“ - Bearbeitete Zelle deren Inhalt ist rot markiert
Test ID : 4.3	
Name	boolFailObjName ist wahr
Testvoraussetzungen	<ul style="list-style-type: none"> - Tabelle „Spec-Objects“ ist geöffnet - Modify bereits gedrückt, Tabelle ist nicht schreibgeschützt - Haltepunkt [1] bei <i>If boolFailObjName = True Then</i> gesetzt - boolFailObjName zur Überwachung hinzufügen - Objectname zu einem ungültigen Wert manipulieren - CheckAlias ist nicht fehlgeschlagen
Testablauf	<ul style="list-style-type: none"> - Button „CheckObjects“ drücken - Hält bei Haltepunkt [1] an - Überwachungsausdrücke anschauen und vergleichen - Schritt „F5“ - Tabelle „Spec Objects“ öffnen - Status und Info anschauen
Erwartetes Resultat	<ul style="list-style-type: none"> - Status: „Error objects“ - Info: „Inconsistent Objectname here and in Spec-Devices“ - Bearbeitete Zelle deren Inhalt ist rot markiert

2.3.2.2 Testergebnis

Datum		07.04.2017	
Testperson		Wanda Lao	
ID	Erwartung erfüllt?	Kommentar	Weitere Schritte
1.0	Ja	Anzahl vom Arrayindex bei allen korrekt	Keine
1.1	Ja	Alle ausgewählte Objekte in EDE sind im Spec-Objects mit Alias Controller, Objectname, Controllernamen und Description gefüllt	Keine
2.0	Ja	<ul style="list-style-type: none"> - frmObjToAlias.ObjOK war True - checkAll war True Objekte sind überschrieben worden, Kontrolle durch das Ändern vom Alias Name	Keine
2.1	Ja	<ul style="list-style-type: none"> - frmObjToAlias.ObjOK war False - checkAll war True Objekte sind unverändert, Kontrolle durch das Ändern vom Alias Name	Keine
3.0	Ja	<ul style="list-style-type: none"> - CheckAlias musste fehlschlagen, geändert durch falschen Alias Controller - Keine Aktion zum „BACnet Data from EDE“ gemacht 	Keine
3.1	Nein	Für jede Objekt wurden die Daten gelöscht und direkt neu hinzugefügt	Code bearbeiten: Zuerst alle Daten löschen, erst danach hinzufügen Nachtest
4.0	Ja	<ul style="list-style-type: none"> - CheckAlias musste fehlschlagen, geändert durch falschen Alias Controller - CheckObjekts wurde abgebrochen 	Keine
4.1	Nein	Wird rot markiert, aber Status zeigt trotz Fehler immer noch OK-Ready anstatt Error, boolFailDev ist auf False	Fehler suchen und Code bearbeiten Nachtest
4.2	Nein	Wird rot markiert, aber Status zeigt trotz Fehler immer noch OK-Ready anstatt Error, boolFailObj ist auf False	Fehler suchen und Code bearbeiten Nachtest
4.3	Nein	Wird rot markiert, aber Status zeigt trotz Fehler immer noch OK-Ready anstatt Error, boolFailDev ist auf False	Fehler suchen und Code bearbeiten Nachtest

2.3.2.3 Korrektur

Datum	07.04.2017
Testperson	Wanda Lao
ID	Korrektur
3.1	<pre> For Each objCell In rngAliasCotroller With aSheet .Cells(objCell.Row, CTRLNAME_COL).ClearContents .Cells(objCell.Row, IP_COL).ClearContents .Cells(objCell.Row, NETWORKNO_COL).ClearContents .Cells(objCell.Row, DEVINST_COL).ClearContents .Cells(objCell.Row, TYPEINT_COL).ClearContents .Cells(objCell.Row, INSTANCE_COL).ClearContents .Cells(objCell.Row, TYPESTR_COL).ClearContents End With Next For Each objCell In rngAliasCotroller With aSheet .Cells(objCell.Row, CTRLNAME_COL).ClearContents .Cells(objCell.Row, IP_COL).ClearContents .Cells(objCell.Row, NETWORKNO_COL).ClearContents .Cells(objCell.Row, DEVINST_COL).ClearContents .Cells(objCell.Row, TYPEINT_COL).ClearContents .Cells(objCell.Row, INSTANCE_COL).ClearContents .Cells(objCell.Row, TYPESTR_COL).ClearContents End With CODE Next </pre> <p>è Separate Loop zum alles zu löschen, ohne dass nachher direkt hinzugefügt wird</p>
4.1	<pre> For Each devCell In rngDevAliasController boolFailDev = False objCell.Font.Color = vbBlack If objCell.Value = devCell.Value Then If XX then .Font.Color = vbRed boolFailDev = True strReason = "XX" Else .Font.Color = vbBlack End If End If Next </pre> <p>è boolFailDev = False darf nicht vorkommen è Farben richtig setzen è Für jede Abfrage (Devicename, IP-Addr, Network-No und Dev-Inst)</p>

4.2	<pre> If objCell.Value = edeCell.Value Then objCell.Font.Color = vbBlack boolFailObjName = False If Dev-Inst(Spec-Objects) = DevInst(EDE) Then objCell.Font.Color = vbBlack 'Check type If XX Then .Font.Color = vbRed boolFailObj = True strReason = "XX" End If End If End If If EDE <> Enum Then boolFailObj = True strReason = "Objecttype" .Value = "" Else boolFailObj = False .Value = .Value(Enum) End If è boolFailObj = False darf nicht vorkommen è Farben richtig setzen è Für jede Abfrage (Type, Instance) è (Typename à konnte nicht gelöst werden) </pre>
4.3	Konnte nicht gelöst werden

2.3.2.4 Nachtest

Datum	07.04.2017		
Testperson	Wanda Lao		
ID	Erwartung erfüllt?	Kommentar	Weitere Schritte
3.1	Ja		
4.1	Ja	Schriftfarbe ist nun wieder schwarz, wenn es korrekt ist	
4.2	Nein	Fehler: Entweder immer Error oder immer OK-Ready	Nach der IPA zu verbessern, im Bericht ausführlich begründen
4.3	Nein	Fehler: Entweder immer Error oder immer OK-Ready	Nach der IPA zu verbessern, im Bericht ausführlich begründen

Die Punkte 4.2 und 4.3 konnten aufgrund mangelnde VBA-Kenntnisse und Zeit nicht behoben werden.

2.3.3 Akzeptanz-Test

2.3.3.1 Testfälle

1 Starbedingung und Modify Button

Test ID : 1.0	
Name	Startbedingung
Testvoraussetzungen	<ul style="list-style-type: none"> - TestTemplate.xlsx ist geöffnet - Tabelle „Spec-Objects“ ist geöffnet
Testablauf	<ul style="list-style-type: none"> - Benutzer wählt eine beliebige Zelle ab A12 - Benutzer tippt etwas in die Zelle (oder Doppelklick)
Erwartetes Resultat	<ul style="list-style-type: none"> - Tabelle ist schreibgeschützt - Benutzer wird mit einem Dialogfenster informiert
Test ID : 1.1	
Name	Button: Modify
Testvoraussetzungen	<ul style="list-style-type: none"> - Tabelle „Spec-Objects“ ist geöffnet
Testablauf	<ul style="list-style-type: none"> - Benutzer klick auf den Button „Modify“ - Benutzer wählt eine Zelle an (Doppelklick)
Erwartetes Resultat	<ul style="list-style-type: none"> - Tabelle ist verfügbar - Inputfelder sind weiss - Benutzer kann Änderungen durchführen - Andere Buttons sind nun verfügbar - Status: „Working“ - Button „Modify“ ist deaktiviert

2 Check Alias

Test ID : 2.0	
Name	Mit vorhandenen Testdaten
Testvoraussetzungen	<ul style="list-style-type: none"> - Tabelle „Spec-Objects“ ist geöffnet - Modify bereits gedrückt (Tabelle ist nicht schreibgeschützt) - Testdaten müssen vorhanden sein
Testablauf	<ul style="list-style-type: none"> - Benutzer klickt auf den Button „CheckAlias“
Erwartetes Resultat	<ul style="list-style-type: none"> - Tabelle wird wieder schreibgeschützt - Status: „OK-Alias“ - Datum ist aktuell
Test ID : 2.1	
Name	Mit falschem Alias Controller
Testvoraussetzungen	<ul style="list-style-type: none"> - Tabelle „Spec-Objects“ ist geöffnet - Modify bereits gedrückt (Tabelle ist nicht schreibgeschützt) - Testdaten müssen vorhanden sein
Testablauf	<ul style="list-style-type: none"> - Benutzer bearbeitet einen beliebigen Alias Controller <ul style="list-style-type: none"> · Alias Controller darf nicht in der Tabelle „Spec-Devices“ vorhanden sein - Benutzer klickt auf den Button „CheckAlias“
Erwartetes Resultat	<ul style="list-style-type: none"> - Tabelle wird wieder schreibgeschützt - Status: „Error“ - Info: „Controller not in Spec-Devices“ - Vom Benutzer bearbeitete Zelle wird rot markiert - Button „Modify“ ist aktiviert - Andere Buttons sind deaktiviert
Test ID : 2.2	
Name	Mit zwei gleichen Alias Name
Testvoraussetzungen	<ul style="list-style-type: none"> - Tabelle „Spec-Objects“ ist geöffnet - Modify bereits gedrückt (Tabelle ist nicht schreibgeschützt) - Testdaten müssen vorhanden sein
Testablauf	<ul style="list-style-type: none"> - Benutzer kopiert ein beliebiges Objekt (ganze Zeile) - Benutzer fügt die Zeile ein (unter dem letzten Eintrag[Objekt]) - Benutzer klickt auf den Button „CheckAlias“
Erwartetes Resultat	<ul style="list-style-type: none"> - Tabelle wird wieder schreibgeschützt - Status: „Error“ - Info: „Alias not unique“ - Vom Benutzer hinzugefügtes Objekt und das Original der Kopie deren Alias Name werden rot markiert - Button „Modify“ ist aktiviert - Andere Buttons sind deaktiviert

3 Append from EDE

Test ID : 3.0	
Name	Button "Append" gedrückt
Testvoraussetzungen	<ul style="list-style-type: none"> - Tabelle „Spec-Objects“ ist geöffnet - Modify bereits gedrückt (Tabelle ist nicht schreibgeschützt) - Testdaten in der Tabelle „EDE“ müssen vorhanden sein
Testablauf 1	<ul style="list-style-type: none"> - Benutzer öffnet Tabelle „EDE“ - Benutzer markiert gewünschtes Objekt/gewünschte Objekte mit einem Ü (bei Select) - Benutzer öffnet die Tabelle „Spec-Objects“ - Benutzer klick im Feld „Append selected objects“ auf den Button „from EDE“
Erwartetes Resultat 1	<ul style="list-style-type: none"> - Dialogfenster wird geöffnet - Alle selektierte Objekte werden angezeigt
Testablauf 2	<ul style="list-style-type: none"> - Benutzer klick auf „Append“
Erwartetes Resultat 2	<ul style="list-style-type: none"> - Dialogfester wird geschlossen - Alle Objekte in der Tabelle „Spec-Objects“ sind deselektiert - Alle gewählte Objekte aus der Tabelle „EDE“ wird mit den Inhalt von „Objectname“ und „Description“ in „Spec-Objects“ hinzugefügt - Neue Einträge werden selektiert (Ü)
Test ID : 3.1	
Name	Button "Cancel" gedrückt
Testvoraussetzungen	<ul style="list-style-type: none"> - Tabelle „Spec-Objects“ ist geöffnet - Modify bereits gedrückt (Tabelle ist nicht schreibgeschützt) - Testdaten in der Tabelle „EDE“ müssen vorhanden sein
Testablauf 1	<ul style="list-style-type: none"> - Benutzer öffnet Tabelle „EDE“ - Benutzer markiert gewünschtes Objekt/gewünschte Objekte mit einem Ü (bei Select) - Benutzer öffnet die Tabelle „Spec-Objects“ - Benutzer klick im Feld „Append selected objects“ auf den Button „from EDE“
Erwartetes Resultat 1	<ul style="list-style-type: none"> - Dialogfenster wird geöffnet - Alle selektierte Objekte werden angezeigt
Testablauf 2	<ul style="list-style-type: none"> - Benutzer klick auf „Cancel“
Erwartetes Resultat 2	<ul style="list-style-type: none"> - Fenster wird geschlossen

4 Object to Alias

Test ID : 4.0	
Name	Ohne Aliasname
Testvoraussetzungen	<ul style="list-style-type: none"> - Tabelle „Spec-Objects“ ist geöffnet - Modify bereits gedrückt (Tabelle ist nicht schreibgeschützt) - Testdaten müssen vorhanden sein
Testablauf	<ul style="list-style-type: none"> - Benutzer löscht den Aliasnamen eines beliebigen Objektes - Benutzer selektiert in der Tabelle die ohne Aliasname mit Select (Ü) - Benutzer klickt auf den Button „Obj > Alias“
Erwartetes Resultat	<ul style="list-style-type: none"> - Aliasname wird hinzugefügt
Test ID : 4.1	
Name	Mit Aliasname und Button „Overwrite“
Testvoraussetzungen	<ul style="list-style-type: none"> - Tabelle „Spec-Objects“ ist geöffnet - Modify bereits gedrückt (Tabelle ist nicht schreibgeschützt) - Testdaten müssen vorhanden sein
Testablauf 1	<ul style="list-style-type: none"> - Benutzer selektiert in der Tabelle ein beliebiges Objekt mit Aliasname mit Select (Ü) - Benutzer klickt auf den Button „Obj > Alias“
Erwartetes Resultat 1	<ul style="list-style-type: none"> - Dialogfenster wird geöffnet
Testablauf 2	<ul style="list-style-type: none"> - Benutzer klickt für jedes ausgewähltes Objekt (in Testablauf 1) den Button „Overwrite“
Erwartetes Resultat 2	<ul style="list-style-type: none"> - Dialogfenster schliesst sich - Aliasname wird hinzugefügt - Tabelle ist schreibgeschützt
Test ID : 4.2	
Name	Mit Aliasname und Button „Don't overwrite“
Testvoraussetzungen	<ul style="list-style-type: none"> - Tabelle „Spec-Objects“ ist geöffnet - Modify bereits gedrückt (Tabelle ist nicht schreibgeschützt) - Testdaten müssen vorhanden sein
Testablauf 1	<ul style="list-style-type: none"> - Benutzer selektiert in der Tabelle ein beliebiges Objekt mit Aliasname mit Select (Ü) - Benutzer klickt auf den Button „Obj > Alias“
Erwartetes Resultat 1	<ul style="list-style-type: none"> - Dialogfenster wird geöffnet
Testablauf 2	<ul style="list-style-type: none"> - Benutzer klickt für jedes ausgewähltes Objekt (in Testablauf 1) den Button „Don't overwrite“
Erwartetes Resultat 2	<ul style="list-style-type: none"> - Dialogfenster schliesst sich - Aliasname wird nicht hinzugefügt - Tabelle ist schreibgeschützt

Test ID : 4.3	
Name	Mit Aliasname und Button "Overwrite" und Checkbox gewählt
Testvoraussetzungen	<ul style="list-style-type: none"> - Tabelle „Spec-Objects“ ist geöffnet - Modify bereits gedrückt (Tabelle ist nicht schreibgeschützt) - Testdaten müssen vorhanden sein
Testablauf 1	<ul style="list-style-type: none"> - Benutzer selektiert in der Tabelle ein beliebiges Objekt mit Aliasname mit Select (Ü) - Benutzer klickt auf den Button „Obj > Alias“
Erwartetes Resultat 1	- Dialogfenster wird geöffnet
Testablauf 2	<ul style="list-style-type: none"> - Benutzer wählt Checkbox an - Benutzer klickt den Button „Overwrite“
Erwartetes Resultat 2	<ul style="list-style-type: none"> - Dialogfenster schliesst sich - Aliasname wird hinzugefügt - Tabelle ist schreibgeschützt
Test ID : 4.4	
Name	Mit Aliasname und Button "Don't overwrite" und Checkbox gewählt
Testvoraussetzungen	<ul style="list-style-type: none"> - Tabelle „Spec-Objects“ ist geöffnet - Modify bereits gedrückt (Tabelle ist nicht schreibgeschützt) - Testdaten müssen vorhanden sein
Testablauf 1	<ul style="list-style-type: none"> - Benutzer selektiert in der Tabelle ein beliebiges Objekt mit Aliasname mit Select (Ü) - Benutzer klickt auf den Button „Obj > Alias“
Erwartetes Resultat 1	- Dialogfenster wird geöffnet
Testablauf 2	<ul style="list-style-type: none"> - Benutzer wählt Checkbox an - Benutzer den Button „Don't overwrite“
Erwartetes Resultat 2	<ul style="list-style-type: none"> - Dialogfenster schliesst sich - Aliasname wird nicht hinzugefügt - Tabelle ist schreibgeschützt

5 Alias To Objectname

Test ID : 5.0	
Name	Ohne Objektname
Testvoraussetzungen	<ul style="list-style-type: none"> - Tabelle „Spec-Objects“ ist geöffnet - Modify bereits gedrückt (Tabelle ist nicht schreibgeschützt) - Testdaten müssen vorhanden sein
Testablauf	<ul style="list-style-type: none"> - Benutzer löscht den Objektname eines beliebigen Objektes - Benutzer selektiert in der Tabelle die ohne Objektname mit Select (Ü) - Benutzer klickt auf den Button „Alias > Obj“
Erwartetes Resultat	<ul style="list-style-type: none"> - Objektname wird hinzugefügt
Test ID : 5.1	
Name	Mit Objektname und Button “Overwrite“
Testvoraussetzungen	<ul style="list-style-type: none"> - Tabelle „Spec-Objects“ ist geöffnet - Modify bereits gedrückt (Tabelle ist nicht schreibgeschützt) - Testdaten müssen vorhanden sein
Testablauf 1	<ul style="list-style-type: none"> - Benutzer selektiert in der Tabelle ein beliebiges Objekt mit Objektname mit Select (Ü) - Benutzer klickt auf den Button „Alias > Obj“
Erwartetes Resultat 1	<ul style="list-style-type: none"> - Dialogfenster wird geöffnet
Testablauf 2	<ul style="list-style-type: none"> - Benutzer klickt für jedes ausgewähltes Objekt (in Testablauf 1) den Button „Overwrite“
Erwartetes Resultat 2	<ul style="list-style-type: none"> - Dialogfenster schliesst sich - Objektname wird hinzugefügt
Test ID : 5.2	
Name	Mit Objektname und Button “Don’t overwrite“
Testvoraussetzungen	<ul style="list-style-type: none"> - Tabelle „Spec-Objects“ ist geöffnet - Modify bereits gedrückt (Tabelle ist nicht schreibgeschützt) - Testdaten müssen vorhanden sein
Testablauf 1	<ul style="list-style-type: none"> - Benutzer selektiert in der Tabelle ein beliebiges Objekt mit Objektname mit Select (Ü) - Benutzer klickt auf den Button „Alias > Obj“
Erwartetes Resultat 1	<ul style="list-style-type: none"> - Dialogfenster wird geöffnet
Testablauf 2	<ul style="list-style-type: none"> - Benutzer klickt für jedes ausgewähltes Objekt (in Testablauf 1) den Button „Don’t overwrite“
Erwartetes Resultat 2	<ul style="list-style-type: none"> - Dialogfenster schliesst sich - Objektname wird nicht hinzugefügt

Test ID : 5.3	
Name	Mit Objektname und Button "Overwrite" und Checkbox gewählt
Testvoraussetzungen	<ul style="list-style-type: none"> - Tabelle „Spec-Objects“ ist geöffnet - Modify bereits gedrückt (Tabelle ist nicht schreibgeschützt) - Testdaten müssen vorhanden sein
Testablauf 1	<ul style="list-style-type: none"> - Benutzer selektiert in der Tabelle ein beliebiges Objekt mit Objektname mit Select (Ü) - Benutzer klickt auf den Button „Alias > Obj“
Erwartetes Resultat 1	- Dialogfenster wird geöffnet
Testablauf 2	<ul style="list-style-type: none"> - Benutzer wählt Checkbox an - Benutzer klickt den Button „Overwrite“
Erwartetes Resultat 2	<ul style="list-style-type: none"> - Dialogfenster schliesst sich - Objektname wird hinzugefügt
Test ID : 5.4	
Name	Mit Objektname und Button "Don't overwrite" und Checkbox gewählt
Testvoraussetzungen	<ul style="list-style-type: none"> - Tabelle „Spec-Objects“ ist geöffnet - Modify bereits gedrückt (Tabelle ist nicht schreibgeschützt) - Testdaten müssen vorhanden sein
Testablauf 1	<ul style="list-style-type: none"> - Benutzer selektiert in der Tabelle ein beliebiges Objekt mit Objektname mit Select (Ü) - Benutzer klickt auf den Button „Alias > Obj“
Erwartetes Resultat 1	- Dialogfenster wird geöffnet
Testablauf 2	<ul style="list-style-type: none"> - Benutzer wählt Checkbox an - Benutzer den Button „Don't overwrite“
Erwartetes Resultat 2	<ul style="list-style-type: none"> - Dialogfenster schliesst sich - Objektname wird nicht hinzugefügt

6 BACnet Data from EDE

Test ID : 6.0	
Name	Button „Yes“
Testvoraussetzungen	<ul style="list-style-type: none"> - Tabelle „Spec-Objects“ ist geöffnet - Modify bereits gedrückt (Tabelle ist nicht schreibgeschützt) - Testdaten müssen vorhanden sein - Keine Doppeleinträge (für Aliasname)
Testablauf	<ul style="list-style-type: none"> - Benutzer klickt auf den Button „from EDE“ im Feld BACnet Data
Erwartetes Resultat	<ul style="list-style-type: none"> - Dialogfenster öffnet sich
Testablauf	<ul style="list-style-type: none"> - Benutzer klickt auf den Button „Yes“
Erwartetes Resultat	<ul style="list-style-type: none"> - Dialogfenster öffnet sich - Einträge von Controllernamen, IP-Addr/Node-ID, Network-No Dev-Inst, Typename, Type und Instance werden aktualisiert/hinzugefügt
Test ID : 6.1	
Name	Button „No“
Testvoraussetzungen	<ul style="list-style-type: none"> - Tabelle „Spec-Objects“ ist geöffnet - Modify bereits gedrückt (Tabelle ist nicht schreibgeschützt) - Testdaten müssen vorhanden sein - Keine Doppeleinträge (für Aliasname)
Testablauf	<ul style="list-style-type: none"> - Benutzer klickt auf den Button „from EDE“ im Feld BACnet Data
Erwartetes Resultat	<ul style="list-style-type: none"> - Dialogfenster öffnet sich
Testablauf	<ul style="list-style-type: none"> - Benutzer klickt auf den Button „No“
Erwartetes Resultat	<ul style="list-style-type: none"> - Dialogfenster schliesst sich - <i>Nichts wird geändert</i>

7 Check Objects

Test ID : 7.0	
Name	Button: CheckObjects
Testvoraussetzungen	<ul style="list-style-type: none"> - Tabelle „Spec-Objects“ ist geöffnet - Modify bereits gedrückt (Tabelle ist nicht schreibgeschützt) - Testdaten müssen vorhanden sein - Alle Einträge existieren
Testablauf	<ul style="list-style-type: none"> - Benutzer klickt auf den Button „CheckObjects“
Erwartetes Resultat	<ul style="list-style-type: none"> - Status: Objects-Ready
Test ID : 7.1	
Name	Button: CheckObjects
Testvoraussetzungen	<ul style="list-style-type: none"> - Tabelle „Spec-Objects“ ist geöffnet - Modify bereits gedrückt (Tabelle ist nicht schreibgeschützt) - Testdaten müssen vorhanden sein
Testablauf	<ul style="list-style-type: none"> - Benutzer ändert Controllername, IP-Addr, Network-No oder Dev-Inst zu einem ungültigen oder nicht existierenden Wert - Benutzer klickt auf den Button „CheckObjects“
Erwartetes Resultat	<ul style="list-style-type: none"> - Geänderte Zelle wird rot markiert - Status: Error device - Info: Inconsistent <Controllername, IP-Addr, Network-No oder Dev-Inst> here and in Spec-Devices
Test ID : 7.2	
Name	Button: CheckObjects
Testvoraussetzungen	<ul style="list-style-type: none"> - Tabelle „Spec-Objects“ ist geöffnet - Modify bereits gedrückt (Tabelle ist nicht schreibgeschützt) - Testdaten müssen vorhanden sein
Testablauf	<ul style="list-style-type: none"> - Benutzer ändert Objectname, Obj-Type oder Obj-Instance zu einem ungültigen oder nicht existierenden Wert - Benutzer klickt auf den Button „CheckObjects“
Erwartetes Resultat	<ul style="list-style-type: none"> - Geänderte Zelle wird rot markiert - Status: Error Objects - Info: Inconsistent < Objectname, Obj-Type oder Obj-Instance> here and in EDE

8 Select / Deselect

Gültigen Bereich: Ab Reihe 13, ungültiger Bereich: oberhalb Reihe 13

Test ID : 8.0	
Name	Zelle selektieren im gültigen Bereich
Testvoraussetzungen	<ul style="list-style-type: none"> - Die Tabelle „Spec-Objects“ ist aktiviert - Modify bereits gedrückt (Tabelle ist nicht schreibgeschützt) - Testdaten müssen vorhanden sein - Zelle darf nicht schon im Voraus selektiert sein
Testablauf	<ul style="list-style-type: none"> - User wählt eine Zelle, mehrere Zellen nacheinander oder mehrere Zellen mit Abständen im gültigen Bereich - User klickt auf Select Button [<input type="checkbox"/>]
Erwartetes Resultat	<ul style="list-style-type: none"> - Ausgewählte Zelle/n wird/werden im Bereich [Select] mit einem <input type="checkbox"/> versehen
Test ID : 8.1	
Name	Zelle deslektieren im gültigen Bereich
Testvoraussetzungen	<ul style="list-style-type: none"> - Tabelle „Spec-Objects“ ist geöffnet - Modify bereits gedrückt (Tabelle ist nicht schreibgeschützt) - Testdaten müssen vorhanden sein - Zelle/n im Bereich [Select] muss/müssen selektiert sein (mit einem <input type="checkbox"/> markiert)
Testablauf	<ul style="list-style-type: none"> - User wählt eine Zelle, mehrere Zellen nacheinander oder mehrere Zellen mit Abständen im gültigen Bereich (mit einem Häkchen [<input type="checkbox"/>]) - User klickt auf Select Button [x]
Erwartetes Resultat	<ul style="list-style-type: none"> - Ausgewählte Zelle/n im Bereich [Select] ist/sind nun leer
Test ID : 8.2	
Name	Zelle selektieren im ungültigen Bereich
Testvoraussetzungen	<ul style="list-style-type: none"> - Die Tabelle „Spec-Objects“ ist aktiviert - Modify bereits gedrückt (Tabelle ist nicht schreibgeschützt) - Testdaten müssen vorhanden sein - Zelle darf nicht schon im Voraus selektiert sein
Testablauf	<ul style="list-style-type: none"> - User wählt eine Zelle, mehrere Zellen nacheinander oder mehrere Zellen mit Abständen im gültigen Bereich - User klickt auf Select Button [<input type="checkbox"/>]
Erwartetes Resultat	<ul style="list-style-type: none"> - Zelle wurden nicht selektiert
Test ID : 8.3	
Name	Zelle deslektieren im ungültigen Bereich
Testvoraussetzungen	<ul style="list-style-type: none"> - Die Tabelle „Spec-Objects“ ist aktiviert - Modify bereits gedrückt (Tabelle ist nicht schreibgeschützt) - Testdaten müssen vorhanden sein - Zelle darf nicht schon im Voraus selektiert sein - Zelle/n muss/müssen selektiert sein (mit einem <input type="checkbox"/> versehen)
Testablauf	<ul style="list-style-type: none"> - User wählt eine Zelle, mehrere Zellen nacheinander oder mehrere Zellen mit Abständen im gültigen Bereich (mit einem Häkchen [<input type="checkbox"/>]) - User klickt auf Select Button [x]
Erwartetes Resultat	<ul style="list-style-type: none"> - Zellen wurden nicht selektiert

2.3.3.2 Testergebnis**2.3.3.2.1 Testergebnis von Urs Heimann**

Datum		06.04.2017	
Testperson		Urs Heimann	
ID	Erwartung erfüllt?	Kommentar	Weitere Schritte
1.0	Ja		
1.1	Ja		
2.0	Ja		
2.1	Ja	Gute Idee, die leeren Zellen mit „XXXX“ zu füllen damit die rote Farbe sichtbar wird.	
2.2	Ja		
3.0	Ja		
3.1	Ja		
4.0	Ja		
4.1	Ja	<p>Verbesserungsvorschlag 1: Im Dialog wird gefragt "Are you sure you want to overwrite: 'ValIn4' ?" Besser wäre "Are you sure you want to overwrite: 'ValIn4' with 'ValOut4' ?" Das würde in einigen Fällen erlauben den Dialog zu beantworten ohne den Eintrag in der Liste zu suchen.</p> <p>Verbesserungsvorschlag 2: Der Dialog "Are you sure you want to overwrite...?" wird auch angezeigt wenn Obj > Alias am bestehenden Alias Name gar nichts ändert. Besser wäre auf das Einblenden des Dialogs zu verzichten falls bestehender und neuer „Alias Name“ identisch sind.</p> <p>Beides ist in der Spezifikation nicht explizit gefordert und für den Einsatz des Tools nicht zwingend notwendig. Deshalb ist dies lediglich als Wunsch anzusehen.</p>	Ergänzen falls genügend Zeit vorhanden.
4.2	Ja		
4.3	Ja		
4.4	Ja		

5.0	Ja		
5.1	Ja		
5.2	Ja		
5.3	Nein	Sind mehrere Objekte selektiert, wird nach dem bestätigen mit „Overwrite“ nur das im Dialog erwähnte Objekt überschrieben, die folgenden nicht. Hinweis: Bei der Funktion „Obj > Alias“ funktioniert dies korrekt.	Bitte korrigieren
5.4	Ja		
6.0	Ja		
6.1	Ja		
7.0	Ja		
7.1	Teilweise	<ol style="list-style-type: none"> 1. Geänderte Zellen werden wie gefordert rot markiert, „Info“ und „Status“ suggerieren aber an dass der Check erfolgreich war. 2. Werden die fehlerhaften Inhalte korrigiert und der Check wiederholt, bleiben die Zellen rot markiert, die Farbe wird nicht entfernt. 	Bitte korrigieren
7.2	Teilweise	Dito Test 7.1 Zudem: Stimmt die „Device-Instance“ nicht mit der EDE-Liste überein wird diese nicht rot markiert und die Felder „Type“ und „Instance“ in der Folge nicht geprüft.	Bitte korrigieren
8.0	Ja		
8.1	Ja		
8.2	Ja		
8.3	Ja		

2.3.3.2.2 Testergebnis von der IPA-Ausführende

Datum		10.04.2017	
Testperson		Wanda Lao	
ID	Erwartung erfüllt?	Kommentar	Weitere Schritte
1.0	Ja	-	-
1.1	Ja	-	-
2.0	Ja	-	-
2.1	Ja	-	-
2.2	Ja	-	-
3.0	Ja	-	-
3.1	Ja	-	-
4.0	Ja	-	-
4.1	Ja	-	-
4.2	Ja	-	-
4.3	Ja	-	-
4.4	Ja	-	-
5.0	Ja	-	-
5.1	Ja	-	-
5.2	Ja	-	-
5.3	Ja	-	-
5.4	Ja	-	-
6.0	Ja	-	-
6.1	Ja	-	-
7.0	Ja	-	-
7.1	Teilweise	Zeigt nur OK-Ready an, egal ob es falsch oder richtig ist. Wird aber rot angezeigt.	Fehler nach der IPA beheben
7.2	Teilweise	Zeigt nur OK-Ready an, egal ob es falsch oder richtig ist. Wird aber rot angezeigt.	Fehler nach der IPA beheben
8.0	Ja	-	-
8.1	Ja	-	-
8.2	Ja	-	-
8.3	Ja	-	-

2.4 Auswertung

2.4.1 Schlusswort

Ich habe bereits im Vorfeld der IPA einen Einblick ins TsNet bekommen und habe mich damit auch schon beschäftigt. Das Programmieren mit VBA war nicht neu für mich, da ich schon öfters damit gearbeitet habe.

Da ich IPERKA als Projektmanagementmodell ausgewählt habe, erleichterte es einiges bei der IPA. Das Modell ist mir durchaus bekannt und habe schon auch nach diesem Schema gearbeitet.

Das Informieren und die Planung sowie auch das Entscheiden verliefen gut und nach Zeitplan. Das Erstellen vom Zeitplan ist eine schwierige Aufgabe, jedoch gut gemeistert worden, da ich schon einige Male einen Zeitplan erstellen musste.

Beim Realisieren merkte ich, etwa in der Hälfte, dass nicht alles genau nach Zeitplan ist. Es gab jedoch keine grossen Differenzen. Es sind nur kleine Fehleinschätzungen der Anzahl Stunden für die jeweiligen Funktionen.

Die White-Box-Test Ermittlung wurde mit genügender Zeit geplant, jedoch brauchte diese Tätigkeit noch mehr Zeit. Es war schwierig für mich zu entscheiden, was ich alles testen möchte.

Rückblickend reichen die erstellten White-Box-Testfälle aus. Fehler, welche Auswirkungen auf den Akzeptanztest hatten, wurden durch den Test entdeckt. Dank der Entscheidung, den Akzeptanztest schon während der White-Box-Tests an Urs Heimann abzugeben, konnte man die Auswirkungen von den Fehlern einsehen.

Bei der Kontrolle brauchte ich ebenfalls mehr Zeit als geplant. Die Problematik war hier die Fehlerbehebung nach dem White-Box-Test, aufgrund mangelnde VBA-Kenntnisse und fehlender Zeit. Dieses Risiko wurde im IPA-Bericht beschrieben. Damit es keine grössere Verzögerungen auftreten, musste ich mir Prioritäten setzen und konnte nicht alle Fehler beheben. Dieser Entscheid wurde ebenfalls im Bericht begründet.

Im Grossen und Ganzen kann ich behaupten, dass die IPA an sich gut verlaufen ist. Die Probleme, die ich bei der IPA gehabt habe, konnte ich selbstständig lösen. Alle Entscheidungen, die während der IPA gefallen sind, wurden im Bericht begründet. Der Bericht und die Arbeitsjournale wurden fortlaufend ergänzt.

Dass es nach fast jeder Ausführung eines Buttons, die Excel-Tabelle schreibgeschützt wurde, fand ich nicht relevant. Die Tabelle ist dann ständig in einem schreibgeschützten Modus, was lästig sein kann und nicht überall notwendig ist. Man könnte überlegen, die Anforderungen einzelner Funktionen zu bearbeiten. Gemäss Spezifikation ist sie nun so implementiert.

Die Funktion, die von mir implementiert wurden könnte man an einige Stellen besser machen. Für die Spalten und Reihen wurden Konstanten erstellt, jedoch könnte mehr Konstante definiert sein, da es im Code immer noch Stellen zu finden sind, wo die Bereiche (mit Reihen und Spalte) numerisch und statisch codiert worden sind.

Für nachfolgende Abnehmer vom Projekt könnte die statische Codierung zu Zeitverzögerungen kommen, da man überall nach einem fixem Bereich suchen muss, um diese Ändern zu können. Mit Konstanten werden sie am Anfang vom Code definiert und initialisiert, da können sie jeder Zeit angepasst werden.

2.5 Glossar

ID	Begriff	Erklärung
1	ABT	ABT (Automation Building Tool) Gebäudeautomation Tool für Engineering, Inbetriebnahme und Wartung von BACS.
2	ActiveX Steuerelemente [4]	ActiveX bezeichnet ein veraltetes Softwarekomponenten-Modell von Microsoft für aktive Inhalte. ActiveX-Komponenten erweitern die Component-Object-Model-Standards (COM) von Microsoft.
3	Akzeptanztest	Überprüft, ob die Software die funktionalen Erwartungen und Anforderungen im Gebrauch erfüllt.
4	Applications	Gesamte Applikation zum Steuern und Regeln einer Aufgabenstellung der Gebäudeautomation in einem Controller. Die Applikation umfasst auch das Erfassen und Ausgeben von Daten von/an Feldgeräte, Bedienpanels, Leitsysteme und andere Controller. Beispiel: Lichtsteuerung und Temperaturregelung für mehrere Räume. Eine Applikation besteht aus Applikationsfunktionen (AFs) und Charts.
5	BACnet	Building Automation and Control Networks, ist ein Netzwerkprotokoll für die Gebäudeautomation
6	Columns	Spalte einer Excel-Tabelle
7	Controller	Unter Controller versteht man frei programmierbares Automationsgerät für HLK. Die Firmware beinhaltet bereits alle Funktionen zur Kommunikation mit den Feldgeräten und mit dem Anwender. Eine spezifische HLK Applikation wird mit dem ABT konstruiert und auf den Controller geladen.
8	EDE	BACnet Engineering Data Exchange wird verwendet, um Daten von einem BACnet Engineering System auf ein anderes zu übertragen. Der Inhalt der Datei ist standardisiert
9	IPA	IPA (Individuelle praktische Arbeit oder auch individuelle Projekt Arbeit) wird am Ende der Erstausbildung im Lehrbetrieb gemacht und bildet einen Teil im Qualifikationsbereich "Praktische Arbeiten".
10	IPERKA	IPERKA ist eine Projektmanagementmethode, die aus sechs Schritten besteht. (Informieren, Planen, Entscheiden, Realisieren, Kontrollieren, Auswerten)
11	Modul	Module sind Container für Variablen und für Code
12	OLEObjects [5]	Jedes OLEObject-Objekt stellt ein ActiveX-Steuerelement oder ein verknüpftes bzw. eingebettetes OLE-Objekt dar.
13	Prozeduren / Funktionen	Prozeduren: Gruppen von Anweisungen haben aber kein Rückgabewert Funktionen: wie Prozeduren aber mit einem Rückgabewert
14	Range	Bereich einer Excel-Tabelle
15	Rows	Reihe einer Excel-Tabelle
16	Struktogramm	Struktogramme werden auch als Nassi-Shneiderman-Diagramme bezeichnet. Sie stellen Programmstrukturen dar.

17	TsNet	Testtool zum Open-Loop Test für Applikationen. Es besteht unter anderem aus einem Excel-File zur Testspezifikation mit der Definition von Eingangsbedingungen und den erwarteten Reaktionen. Testskripts ermöglichen das automatisierte Abfahren und Auswerten dieser Testspezifikation.
18	VBA	VBA (Visual Basic Application), ist die Programmiersprache von Microsoft Office. Und wird beim Automatisieren von wiederholten Aufgaben verwendet
19	White-Box-Test	White-Box-Test ist eine Methode für Software- Tests. Man hat Zugriff auf den Code
20	Workflow	Ein Workflow ist ein Block von Aktivitäten und Ergebnisse. Es fasst die wichtigsten Prozessschritte für eine bestimmte Disziplin zusammen. Somit stellt ein Workflow immer nur eine begrenzte Sicht des Gesamtprozesses und muss nicht vollständig sein. Die Ansicht von Phasen müssen immer für eine vollständige Darstellung aller Aktivitäten verwendet werden

Tabelle 11: Glossar

2.6 Quellen

Nr. [x]	Thema	Quelle	Datum
[1]	Projektauftrag	www.pkOrg.ch	31.03.2017
[2]	IPERKA	http://www.tgabathuler.ch/Iperka/Bilder/IPERKA_Modell.svg	31.03.2017
[3]	HUS Struktogramm	http://www.struktogrammer.ch/	31.03.2017
[4]	ActiveX Steuerelemente	https://de.wikipedia.org/wiki/ActiveX	10.04.2017
[5]	OLEObjects	https://msdn.microsoft.com/de-de/library/office/ff840244.aspx	10.04.2017

Tabelle 12: Quellenverzeichnis

2.7 Anhang

Im Anhang befindet sich der Quellcode. Diese umfasst alle Programmierarbeiten, die während der IPA geleistet wurden. Sie ist wie folgt aufgebaut:

In der Userformen:

- AppendFromEDE
- ObjToAlias
- AliasToObjName

Im Modul „mdl_SpecObjects“:

- Header
- Globale Variablen und Konstanten
- Funktion: setConstantes()
- Funktion: setUsedVariables()
- Funktion: setObjects()
- Funktion: WBOpenObj
- Funktion: ModifyObj
- Funktion: CheckAlias
- Funktion: CheckObjects
- Funktion: AppendFromEDE
- Funktion: ObjToAlias
- Funktion: AliasToObjName
- Funktion: BACnetDataEDE
-

In Microsoft Excel Objekte:

- Cmd_x_Click
- Funktion: Select
- Funktion: Deselect

Der nicht Eigenanteil wird so mit oranger Farbe markiert:

```
'TESTCODE
Sub Test ()
    If eigenanteil Then
        'code
    Else
        'code
    End If
End Sub
```

Abbildung 41: Beispiel – nicht Eigenanteil

2.7.1 Code

2.7.1.1 *User Form: AppendFromEDE*

```
Option Explicit

Public appendClick As Boolean

Private Sub cmd_Append_Click()
    appendClick = True
    frmAppendFromEDE.Hide
End Sub

Private Sub cmd_CancelAppend_Click()
    appendClick = False
    frmAppendFromEDE.Hide
End Sub
```

2.7.1.2 *User Form: ObjToAlias*

```
Option Explicit

Public ObjOK As Boolean

Private Sub cmd_CancelOtA_Click()
    ObjOK = False
    frmObjToAlias.Hide
End Sub

Private Sub cmd_OverwriteAliasn_Click()
    ObjOK = True
    frmObjToAlias.Hide
End Sub
```

2.7.1.3 *User Form: AliasToObjName*

```
Option Explicit

Public AliasOK As Boolean

Private Sub cmd_CancelAtO_Click()
    AliasOK = False
    frmAliasToObj.Hide
End Sub

Private Sub cmd_OverwriteObjn_Click()
    AliasOK = True
    frmAliasToObj.Hide
End Sub
```

2.7.1.4 Header

Option Explicit

```

'-----
' Name:          mdl_SpecObjects
' File:          TestTemplate.xlsm
'-----
' Copyright @2017,  SIEMENS Building Technologies
'-----
' OS:
' Lang:
'-----
' Project:       TsNet
' Author:        Wanda Lao , Michael Speckien
' Version:       V.1
' Date:          03-08-2016
'-----
' Description:    The functions for the table "Spec-Objects" is implemented in mdl_SpecObjects
'-----
' Parameters:
'-----
' History:        03-08-2016  Wanda Lao
'                  V.1 Document Creation
'-----
'                  28-03-2017  Wanda Lao
'                  V.1 setObject()
'                  V.1 WBOpenObj()
'                  V.1 Modify()
'                  V.1 CheckAlias()
'-----
'                  30-03-2017  Wanda Lao
'                  V.1 AppendFromEDE()
'                  V.1 ObjToAlias()
'                  V.1 setObject()
'-----
'                  31-03-2017  Wanda Lao
'                  V.1 AliasToObj()
'                  V.1 setUsedVariables()
'                  created BACnetDataEDE()
'-----
'                  03-04-2017  Wanda Lao
'                  created Constantes for Sheet Objects
'                  V.1 BACnetDataEDE()
'                  V.1 CheckObjects
'-----
'                  07-04-2017  Wanda Lao
'                  Bugfixing in BACneDataEDE(), AliasToObj(), CheckObjects()
'-----

```

2.7.1.5 Globale Variablen und Konstante

```

'-----
' VARIABLES AND CONSTANTES -----
'-----

'CONSTANTES FOR setConstantes -----
'Constantes for Sheet Spec-Objects
Public SELECT_COL As Integer
Public ANAME_COL As Integer
Public ACTRL_COL As Integer
Public OBJNAME_COL As Integer
Public CTRLNAME_COL As Integer
Public DESCRIPTION_COL As Integer
Public IP_COL As Integer
Public NETWORKNO_COL As Integer
Public DEVINST_COL As Integer
Public TYPESTR_COL As Integer '1.TYPE
Public TYPEINT_COL As Integer '2.TYPE
Public INSTANCE_COL As Integer
Public OBJSTART_ROW As Integer

'VARIABLES FOR setObject() -----
'Variables for OLEObjects
Public objModify As OLEObject
Public objCheckAlias As OLEObject
Public objCheckObjects As OLEObject
Public objAppendFromEDE As OLEObject
Public objObjectToAlias As OLEObject
Public objAliasToObj As OLEObject
Public objBACnetFromEDE As OLEObject
Public objSelect As OLEObject
Public objDeselect As OLEObject

'TODO: set more ranges for specific defined ranges(var)
'TODO: get the last cell with value in sheet - replace with lastEntryXX (only one lastEntry)

'VARIABLES FOR setUsedVariables() -----
'VARIABLES FOR SHEET SPEC-OBJECTS -----
Public aSheet As Worksheet
Public objCell As range 'for each cell (count)
Public rngSelectObj As range 'to deselect and select
Public rngObjNameObj As range
Public lastEntryObjName As Long 'last row in rngSelectObj, add 1 for each new appended line

Public rngAliasName As range
Public lastEntryAliasName As Long

Public rngAliasCotroller As range
Public lastEntryAliasController As Long

'VARIABLES FOR SHEET EDE -----
Public edeSheet As Worksheet
Public edeCell As range 'for each cell (count)
Public lastEntryNameEDE As Long
Public rngSelectEDE As range
Public rngObjNameEDE As range

'VARIABLES FOR SHEET SPEC-DEVICES -----
Public devSheet As Worksheet
Public devCell As range 'for each cell (count)
Public lastEntryDevInst As Long
Public lastEntryAliasCtrl As Long
Public rngDevInst As range
Public rngDevAliasController As range

'VARIABLES FOR SHEET ENUM -----
Public enumSheet As Worksheet
Public enumCell As range 'for each cell (count)
Public lastEntryEnum As Long
Public rngObjtypeEnum As range

```


2.7.1.6 Funktion: setConstantes()

```

' -----
' Description:      Set the global "constantes"
'                  MODIFY IF NECESSARY
' -----

Sub setConstantes()
    'SHEET SPEC-OBJECTS
    SELECT_COL = 1
    ANAME_COL = 2
    ACTRL_COL = 3
    OBJNAME_COL = 5
    CTRLNAME_COL = 6
    DESCRIPTION_COL = 7
    IP_COL = 8
    NETWORKNO_COL = 9
    DEVINST_COL = 10
    TYPESTR_COL = 11
    TYPEINT_COL = 12
    INSTANCE_COL = 13

    OBJSTART_ROW = 13
End Sub

```

2.7.1.7 Funktion: setUsedVariables()

```

' -----
' Description:      Set the global variables
' -----

Sub setUsedVariables()

    Call setConstantes

    'set variables for EDE
    Set edeSheet = ActiveWorkbook.Sheets("EDE")
    With edeSheet
        lastEntryNameEDE = .Cells(Rows.Count, 4).End(xlUp).Row 'last entry in row, from objectname
        Set rngSelectEDE = .range(.Cells(8, 1), .Cells(lastEntryNameEDE, 1))
        Set rngObjNameEDE = .range(.Cells(8, 4), .Cells(lastEntryNameEDE, 4))
    End With

    'set variables for spec devices
    Set devSheet = ActiveWorkbook.Sheets("Spec-Devices")
    With devSheet
        lastEntryDevInst = .Cells(Rows.Count, 1).End(xlUp).Row
        Set rngDevInst = .range(.Cells(20, 1), .Cells(lastEntryDevInst, 2))

        lastEntryAliasCtrl = .Cells(Rows.Count, 3).End(xlUp).Row
        Set rngDevAliasController = .range(.Cells(20, 3), .Cells(lastEntryAliasCtrl, 2))
    End With

    'set variables for enum
    Set enumSheet = ActiveWorkbook.Sheets("Enum")
    With enumSheet
        lastEntryEnum = .Cells(Rows.Count, 1).End(xlUp).Row
        Set rngObjtypeEnum = .range(.Cells(2, 1), .Cells(lastEntryEnum, 1))
    End With

    'set variables for spec objects
    Set aSheet = ActiveWorkbook.Sheets("Spec-Objects")
    With aSheet

        'lastEntry in range objectname (from EDE - only objectname entires exists) look for the last entry
        'lastEntryObjName = range("ctl_InputFields").End(xlDown).Row
        lastEntryObjName = .Cells(Rows.Count, OBJNAME_COL).End(xlUp).Row
        lastEntryAliasName = .Cells(Rows.Count, ANAME_COL).End(xlUp).Row
        lastEntryAliasController = .Cells(Rows.Count, ACTRL_COL).End(xlUp).Row
        Set rngSelectObj = .range(.Cells(OBJSTART_ROW, SELECT_COL), .Cells(lastEntryObjName, SELECT_COL))
        Set rngAliasName = .range(.Cells(OBJSTART_ROW, ANAME_COL), .Cells(lastEntryAliasName, ANAME_COL))
        Set rngObjNameObj = .range(.Cells(OBJSTART_ROW, OBJNAME_COL), .Cells(lastEntryObjName, OBJNAME_COL))
        Set rngAliasController = .range(.Cells(OBJSTART_ROW, ACTRL_COL), .Cells(lastEntryAliasController, ACTRL_COL))
    End With
End Sub

```

2.7.1.8 Funktion: setObject()

```

' -----
' Description:      Assign objects to variables
' -----
Sub setObject()
    Call setUsedVariables
    With aSheet
        Set objModify = .OLEObjects("cmd_ModifyObj")
        Set objCheckAlias = .OLEObjects("cmd_CheckAlias")
        Set objCheckObjects = .OLEObjects("cmd_CheckObjects")
        Set objAppendFromEDE = .OLEObjects("cmd_AppendFromEDE")
        Set objObjectToAlias = .OLEObjects("cmd_ObjToAlias")
        Set objAliasToObj = .OLEObjects("cmd_AliasToObj")
        Set objBACnetFromEDE = .OLEObjects("cmd_BACnetFromEDE")
        Set objSelect = .OLEObjects("cmd_Select")
        Set objDeselect = .OLEObjects("cmd_Deselect")
    End With
End Sub

```

2.7.1.9 Funktion: WBOpenObj

```

' -----
' Description:      Opens the sheet in read only mode and disables all functions except Modify Button
' -----
Sub WBOpenObj()
    Call setObject
    'Set Properties
    objModify.Enabled = True
    objCheckAlias.Enabled = False
    objCheckObjects.Enabled = False
    objAppendFromEDE.Enabled = False
    objObjectToAlias.Enabled = False
    objAliasToObj.Enabled = False
    objBACnetFromEDE.Enabled = False
    objSelect.Enabled = False
    objDeselect.Enabled = False
    'Protect sheet
    [ctl_InputFields].Interior.Color = RGB(216, 216, 216)
    ActiveSheet.Protect
End Sub

```

2.7.1.10 Funktion: ModifyObj

```

' -----
' Description:      Switches the sheet from read-only to read-write.
'                   Enables all input fields and all functions. Set Status, Info and current Date
' -----
Sub ModifyObj()
    Call setObject
    'Set Properties
    objModify.Enabled = False
    objCheckAlias.Enabled = True
    objCheckObjects.Enabled = True
    objAppendFromEDE.Enabled = True
    objObjectToAlias.Enabled = True
    objAliasToObj.Enabled = True
    objBACnetFromEDE.Enabled = True
    objSelect.Enabled = True
    objDeselect.Enabled = True
    'Unprotect sheet
    ActiveSheet.Unprotect
    [ctl_InputFields].Interior.Color = RGB(255, 255, 255)
    'Set Status, Info, Date
    [ctl_StatusObj].Value = "Working"
End Sub

```

2.7.1.11 **Funktion: CheckAlias**

```

'-----
' Description:      Checks the data and the consistence of input fields. Empty fields are allowed.
'                  Alias Name: Unique and not empty
'                  Alias Controller: must match Alias Device from Spec-Device
'-----
Sub CheckAlias()

'DECLARE AND SET VARIABLES -----
    Dim rngAliasName, cell, CellDev As range
    Dim rngAliasController, rngDeviceList As range
    Dim ResultErr, DevFound As Boolean
    Dim x, y, z
    ResultErr = False
    DevFound = False

    Set rngAliasName = range("B13:B" & Cells(Rows.Count, "E").End(xlUp).Row) 'CHANGED B TO E
    rngAliasName.Select
    Set rngAliasController = range("C13:C" & Cells(Rows.Count, "C").End(xlUp).Row)
    y = Worksheets("Spec-Devices").Cells(1000, 2).Row
    y = Worksheets("Spec-Devices").Cells(1000, 2).End(xlUp).Row
    Set rngDeviceList = Worksheets("Spec-Devices").range("B20:B" & y)
    [ctl_InfoObj] = "Check Alias started"

'FUNCTIONAL PART-----
    'Alias name unique AND not empty
    'Check for duplications in column Device Name
    For Each cell In rngAliasName
        'set the default color to black
        cell.Offset(0, 0).Font.Color = vbBlack
        ' Check for empty cells
        If cell.Offset(0, 0) = "" Then
            cell.Offset(0, 0) = "XXXX"
            cell.Offset(0, 0).Font.Color = vbRed
            [ctl_InfoObj] = "Undefined alias name"
            ResultErr = True
        End If
        'Find the duplicate values in range Device Name
        If Application.Evaluate("COUNTIF(" & rngAliasName.Address & "," & cell.Address & ")") > 1 Then
            'mark the duplicate values to red
            cell.Offset(0, 0).Font.Color = vbRed
            [ctl_InfoObj] = "Alias not unique"
            ResultErr = True
        End If
    Next cell

```

```
For Each cell In rngAliasController
    'set the default color to black
    cell.Offset(0, 0).Font.Color = vbBlack
    x = cell.Offset(0, 0).Value
    ' Check for empty cells
    If cell.Offset(0, 0).Value = "" Then
        cell.Offset(0, 0) = "XXXX"
        cell.Offset(0, 0).Font.Color = vbRed
        [ctl_InfoObj] = "Undefined controller name"
        ResultErr = True
    Else
        'Find the Controller in Spec-Devices
        DevFound = False
        For Each CellDev In rngDeviceList
            y = CellDev.Offset(0, 0)
            If CellDev.Offset(0, 0) = cell.Offset(0, 0) Then
                DevFound = True
                'Comment for IPA
                'cell.Offset(0, 3) = CellDev.Offset(0, 1) 'Controllername
                'cell.Offset(0, 5) = CellDev.Offset(0, 7) 'Ip-Addr
                'cell.Offset(0, 7) = CellDev.Offset(0, 8) 'Dev-Instance
                'cell.Offset(0, 6) = CellDev.Offset(0, 9) 'Network -No
                Exit For
            End If
        Next CellDev
        If Not DevFound Then
            cell.Offset(0, 0).Font.Color = vbRed
            [ctl_InfoObj] = "Controller not in Spec-Devices"
            ResultErr = True
        End If
    End If
Next cell

'Check if everything is OK
If Not ResultErr Then
    [ctl_StatusObj].Value = "OK-Alias"
    [ctl_InfoObj] = "Check Alias finished"
Else
    [ctl_StatusObj].Value = "Error"
End If

[ctl_DateObj] = Now()
'Set sheet to read-only mode
Call WBOpenObj
End Sub
```

2.7.1.12 Funktion: CheckObjects

```

' TODO - FIX CHECK OBJECTS AND CHECK DEVICES (?) -----'
' -----'
' Description: Check the data and the consistence of all input fields, device list and EDE
' -----'
Sub CheckObjects()

'DECLARE AND SET VARIABLES -----'
    Call setUsedVariables

    Dim boolFailDev, boolFailObj, boolFailObjName As Boolean
    boolFailDev = False
    boolFailObj = False
    boolFailObjName = False

    Dim strReason As String

    Dim objTypeFail As String
    objTypeFail = 0

'FUNCTIONAL PART-----'
    Call CheckAlias

    If [ctl_StatusObj].Value <> "Error" Then
        Call ModifyObj

        'Check devices
        For Each objCell In rngAliasCotroller
            For Each devCell In rngDevAliasController

                If objCell.Value = devCell.Value Then
                    'Check device-name
                    If aSheet.Cells(objCell.Row, CTRLNAME_COL) <> devSheet.Cells(devCell.Row, 3) Then
                        aSheet.Cells(objCell.Row, CTRLNAME_COL).Font.Color = vbRed
                        boolFailDev = True
                        strReason = "Device name"
                    Else 'NEU
                        aSheet.Cells(objCell.Row, CTRLNAME_COL).Font.Color = vbBlack
                    End If
                    'Check ip-addr
                    If aSheet.Cells(objCell.Row, IP_COL) <> devSheet.Cells(devCell.Row, 8) Then
                        aSheet.Cells(objCell.Row, IP_COL).Font.Color = vbRed
                        boolFailDev = True
                        strReason = "IP-Addr"
                    Else 'NEU
                        aSheet.Cells(objCell.Row, IP_COL).Font.Color = vbBlack
                    End If
                    'Check network-no
                    If aSheet.Cells(objCell.Row, NETWORKNO_COL) <> devSheet.Cells(devCell.Row, 10) Then
                        aSheet.Cells(objCell.Row, NETWORKNO_COL).Font.Color = vbRed
                        boolFailDev = True
                        strReason = "Network-No"
                    Else 'NEU
                        aSheet.Cells(objCell.Row, NETWORKNO_COL).Font.Color = vbBlack
                    End If
                    'Check dev-inst
                    If aSheet.Cells(objCell.Row, DEVINST_COL) <> devSheet.Cells(devCell.Row, 9) Then
                        aSheet.Cells(objCell.Row, DEVINST_COL).Font.Color = vbRed
                        boolFailDev = True
                        strReason = "Dev-Inst"
                    Else 'NEU
                        aSheet.Cells(objCell.Row, DEVINST_COL).Font.Color = vbBlack
                    End If
                End If
            Next
        Next
    Next
Next

```

```

'Check devices failed:
If boolFailDev = True Then
    [ctl_StatusObj].Value = "Error devices"
    [ctl_InfoObj].Value = "Inconsistent " & strReason & " here and in Spec-Devices"
End If

'Check objects
For Each objCell In rngObjNameObj
    For Each edeCell In rngObjNameEDE
        If objCell.Value = edeCell.Value Then

            'objCell.Font.Color = vbBlack
            boolFailObjName = False

            If aSheet.Cells(objCell.Row, DEVINST_COL).Value = edeSheet.Cells(edeCell.Row, 3).Value Then
                objCell.Font.Color = vbBlack
                'Check type
                If aSheet.Cells(objCell.Row, TYPEINT_COL) <> edeSheet.Cells(edeCell.Row, 5) Then
                    aSheet.Cells(objCell.Row, TYPEINT_COL).Font.Color = vbRed
                    boolFailObj = True
                    strReason = "Type"
                Else
                    aSheet.Cells(objCell.Row, TYPEINT_COL).Font.Color = vbBlack
                    boolFailObj = False
                End If
                'Check instance
                If aSheet.Cells(objCell.Row, INSTANCE_COL) <> edeSheet.Cells(edeCell.Row, 6) Then
                    aSheet.Cells(objCell.Row, INSTANCE_COL).Font.Color = vbRed
                    boolFailObj = True
                    strReason = "Instance"
                Else
                    aSheet.Cells(objCell.Row, INSTANCE_COL).Font.Color = vbBlack
                    boolFailObj = False
                End If

            End If
            Exit For

        Else
            boolFailObjName = True
            objCell.Font.Color = vbRed
            strReason = "Object name"
        End If
    Next
Next

If boolFailObj Or boolFailObjName = True Then
    [ctl_StatusObj].Value = "Error objects"
    [ctl_InfoObj].Value = "Inconsistent " & strReason & " here and in EDE"
End If

'Check if error or not
If boolFailDev Or boolFailObj Or boolFailObjName = True Then
    If boolFailDev And boolFailObj = True Then
        [ctl_InfoObj].Value = "Error devices and objects"
    End If
Else
    [ctl_StatusObj].Value = "OK-Ready"
    [ctl_InfoObj].Value = "Check Objects finished"
End If

[ctl_DateObj].Value = Now()
Call WBOpenObj

End If
End Sub

```

2.7.1.13 *Funktion: AppendFromEDE*

```

'-----'
' Description:      Append objectname from all selected lines of EDE list
'-----'
Sub AppendFromEDE ()

'DECLARE AND SET VARIABLES -----'
Call setUsedVariables

'Arrays
Dim objNameArr()
Dim objDescriptionArr()
Dim devInstArray()
Dim devAliasCtrlArr()
Dim devDevNameArr()

'count variables
Dim x As Integer
Dim y As Integer
Dim i As Integer

i = 0 'Array index
x = 0 'Array counter
y = 1 'Row counter

'FUNCTIONAL PART-----'

'Search all selected objects in EDE and insert in objNameList
For Each edeCell In rngSelectEDE
    If edeCell.Value = ChrW(&H2713) Then
        ReDim Preserve objNameArr(i)
        ReDim Preserve objDescriptionArr(i)

        'ReDim Preserve devInstArray(i)

        ReDim Preserve devAliasCtrlArr(i)
        ReDim Preserve devDevNameArr(i)

        With edeSheet
            objNameArr(i) = .Cells(edeCell.Row, edeCell.Column + 3)
            objDescriptionArr(i) = .Cells(edeCell.Row, edeCell.Column + 6)

            For Each devCell In rngDevInst
                If edeSheet.Cells(edeCell.Row, edeCell.Column + 2).Value = devSheet.Cells(devCell.Row, devCell.Column + 8).Value Then
                    With devSheet

                        devAliasCtrlArr(i) = .Cells(devCell.Row, devCell.Column + 1).Value
                        devDevNameArr(i) = .Cells(devCell.Row, devCell.Column + 2).Value

                    End With
                Else
                    'Empty
                End If
            Next

            End With

            frmAppendFromEDE.objNameList.AddItem (objNameArr(i))
            i = i + 1
        End If
    Next
End Sub

```

```

'Code for showing all selected lines in EDE to add in userform
frmAppendFromEDE.Show

'Clear Listbox (Reset)
frmAppendFromEDE.objNameList.Clear

'Append clicked
If frmAppendFromEDE.appendClick = True Then

    'Deselect all selected Cells in Objects
    For Each objCell In rngSelectObj
        If objCell.Value = ChrW(&H2713) Then
            objCell.Value = ""
        End If
    Next

    'TO DO in the future
    If lastEntryObjName >= lastEntryAliasName Then
        'append the values
        For x = 0 To i - 1
            With aSheet
                .Cells(lastEntryObjName + y, SELECT_COL) = ChrW(&H2713)
                .Cells(lastEntryObjName + y, OBJNAME_COL) = objNameArr(x)
                .Cells(lastEntryObjName + y, DESCRIPTION_COL) = objDescriptionArr(x)
                .Cells(lastEntryObjName + y, DEVINST_COL) = devInstArray(x)
                .Cells(lastEntryObjName + y, ACTRL_COL) = devAliasCtrlArr(x)
                .Cells(lastEntryObjName + y, CTRLNAME_COL) = devDevNameArr(x)
            End With
            y = y + 1
        Next
    Else
        'append the values
        For x = 0 To i - 1
            With aSheet
                .Cells(lastEntryAliasName + y, SELECT_COL) = ChrW(&H2713)
                .Cells(lastEntryAliasName + y, OBJNAME_COL) = objNameArr(x)
                .Cells(lastEntryAliasName + y, DESCRIPTION_COL) = objDescriptionArr(x)
                .Cells(lastEntryAliasName + y, DEVINST_COL) = devInstArray(x)
                .Cells(lastEntryAliasName + y, ACTRL_COL) = devAliasCtrlArr(x)
                .Cells(lastEntryAliasName + y, CTRLNAME_COL) = devDevNameArr(x)
            End With
            y = y + 1
        Next
    End If
    'Insert the created aliasname
    Call ObjToAlias
End If

End Sub

```

2.7.1.14 *Funktion: ObjToAlias*

```

'-----'
' Description:      Reduce the Objectname to the short name and insert it in Aliasname
'-----'
Sub ObjToAlias()

'DECLARE AND SET VARIABLES -----'
Call setUsedVariables

Dim newAliasName As String
Dim checkAll As Boolean

checkAll = False

'FUNCTIONAL PART-----'
For Each objCell In rngSelectObj
    If objCell.Value = ChrW(&H2713) Then
        With aSheet
            If .Cells(objCell.Row, ANAME_COL).Value = "" Then 'if aliasname is empty
                newAliasName = .Cells(objCell.Row, OBJNAME_COL).Value
                newAliasName = Replace(newAliasName, Left(newAliasName, InStrRev(newAliasName, "'")), "")
                newAliasName = Replace(newAliasName, Left(newAliasName, InStrRev(newAliasName, "**")), "")
                .Cells(objCell.Row, objCell.Column + SELECT_COL).Value = newAliasName
            Else 'alias is not empty

                frmObjToAlias.lbl_Aliasname.Caption = "' " & Cells(objCell.Row, ANAME_COL).Value & " '"
                frmObjToAlias.Show

                If frmObjToAlias.ObjOK = True Then 'overwrite
                    newAliasName = .Cells(objCell.Row, OBJNAME_COL).Value
                    newAliasName = Replace(newAliasName, Left(newAliasName, InStrRev(newAliasName, "'")), "")
                    newAliasName = Replace(newAliasName, Left(newAliasName, InStrRev(newAliasName, "**")), "")
                    .Cells(objCell.Row, ANAME_COL).Value = newAliasName

                    'check if checkbox is selected
                    If frmObjToAlias.chk_AllAliasnames.Value = True Then
                        checkAll = True
                        Exit For
                    End If

                Else 'don't overwrite
                    'check if checkbox is selected
                    If frmObjToAlias.chk_AllAliasnames.Value = True Then
                        checkAll = True
                        Exit For
                    End If

                End If

            End With
        End With
    End If
Next

'if checkbox is selected
If checkAll = True Then
    If frmObjToAlias.ObjOK = True Then
        For Each objCell In rngSelectObj
            If objCell.Value = ChrW(&H2713) Then
                With aSheet
                    newAliasName = .Cells(objCell.Row, OBJNAME_COL).Value
                    newAliasName = Replace(newAliasName, Left(newAliasName, InStrRev(newAliasName, "'")), "")
                    newAliasName = Replace(newAliasName, Left(newAliasName, InStrRev(newAliasName, "**")), "")
                    .Cells(objCell.Row, ANAME_COL).Value = newAliasName
                End With
            End If
        Next
    Else 'user don't want to overwrite - overwrites only the empty ones
        For Each objCell In rngSelectObj
            If objCell.Value = ChrW(&H2713) And Cells(objCell.Row, ANAME_COL).Value = "" Then
                With aSheet
                    newAliasName = .Cells(objCell.Row, OBJNAME_COL).Value
                    newAliasName = Replace(newAliasName, Left(newAliasName, InStrRev(newAliasName, "'")), "")
                    newAliasName = Replace(newAliasName, Left(newAliasName, InStrRev(newAliasName, "**")), "")
                    .Cells(objCell.Row, ANAME_COL).Value = newAliasName
                End With
            End If
        Next
    End If
End If

Call CheckAlias

End Sub

```

2.7.1.15 Funktion: AliasToObjName

```

' -----
' Description:      Adds a "*" to the Aliasname and insert it to the Objectname
' -----
Sub AliasToObjName()
'DECLARE AND SET VARIABLES -----
Call setUsedVariables

Dim newObjName As String
Dim checkAll As Boolean

checkAll = False
'FUNCTIONAL PART-----

    For Each objCell In rngAliasName
        With aSheet
            If aSheet.Cells(objCell.Row, SELECT_COL).Value = ChrW(&H2713) Then
                'cell is empty
                If .Cells(objCell.Row, OBJNAME_COL).Value = "" Then
                    newObjName = .Cells(objCell.Row, ANAME_COL).Value
                    newObjName = Replace(newObjName, newObjName, "*" & newObjName)
                    .Cells(objCell.Row, OBJNAME_COL).Value = newObjName
                Else 'cell is not empty

                    frmAliasToObj.lbl_Objectname.Caption = " " & Cells(objCell.Row, OBJNAME_COL).Value & " "
                    frmAliasToObj.Show

                    If frmAliasToObj.AliasOK = True Then
                        newObjName = .Cells(objCell.Row, ANAME_COL).Value
                        newObjName = Replace(newObjName, newObjName, "*" & newObjName)
                        .Cells(objCell.Row, OBJNAME_COL).Value = newObjName
                    End If
                    'check if checkbox is selected
                    If frmAliasToObj.chk_AllObjectnames.Value = True Then
                        checkAll = True
                        Exit For
                    End If
                End If
            End If
        End With
    Next

    'if checkbox is selected
    If checkAll = True Then
        'overwrite all
        If frmAliasToObj.AliasOK = True Then
            For Each objCell In rngSelectObj
                If objCell.Value = ChrW(&H2713) Then 'And Cells(objCell.Row, objCell.Column + 4).Value <> "" Then
                    With aSheet
                        newObjName = .Cells(objCell.Row, ANAME_COL).Value
                        newObjName = Replace(newObjName, newObjName, "*" & newObjName)
                        .Cells(objCell.Row, OBJNAME_COL).Value = newObjName
                    End With
                End If
            Next
        ElseIf frmAliasToObj.AliasOK = False Then
            'check if checkbox is selected but don't want to overwrite
            For Each objCell In rngAliasName
                If objCell.Value = ChrW(&H2713) And Cells(objCell.Row, OBJNAME_COL).Value = "" Then
                    With aSheet
                        newObjName = .Cells(objCell.Row, ANAME_COL).Value
                        newObjName = Replace(newObjName, newObjName, "*" & newObjName)
                        .Cells(objCell.Row, OBJNAME_COL).Value = newObjName
                    End With
                End If
            Next
        End If
    End If

    [ctl_DateObj] = Now()
End Sub

```

2.7.1.16 Funktion: BACnetDataEDE

```

'-----
' Description: Delete all BACnet data (Controllername, IP-Addr, Dev-Inst, Network-No, Type, Instance)
'             Insert new data with help of the device list
'-----
Sub BACnetDataEDE()
'DECLARE AND SET VARIABLES -----
    Dim answerMsg As Integer
    Call setUsedVariables

'FUNCTIONAL PART-----
    Call CheckAlias

    If [ctl_StatusObj].Value <> "Error" Then

        ModifyObj
        answerMsg = MsgBox("Are you sure you want to delete all BACnet data?", vbYesNo + vbQuestion, "Delete all BACnet data")

        If answerMsg = vbYes Then
            'delete all bacnet data --> delete all content or just overwrite it later?

            'ANOTHER WAY
            For Each objCell In rngAliasCotroller

                'Delete all data
                With aSheet
                    .Cells(objCell.Row, CTRLNAME_COL).ClearContents
                    .Cells(objCell.Row, IP_COL).ClearContents
                    .Cells(objCell.Row, NETWORKNO_COL).ClearContents
                    .Cells(objCell.Row, DEVINST_COL).ClearContents
                    .Cells(objCell.Row, TYPEINT_COL).ClearContents
                    .Cells(objCell.Row, INSTANCE_COL).ClearContents
                    .Cells(objCell.Row, TYPESTR_COL).ClearContents
                End With
            Next

            For Each objCell In rngAliasCotroller

                For Each devCell In rngDevInst
                    If objCell.Value = devSheet.Cells(devCell.Row, devCell.Column + 1).Value Then
                        'insert device name, ip, networkno and dev inst
                        Cells(objCell.Row, CTRLNAME_COL).Value = devSheet.Cells(devCell.Row, 3).Value
                        Cells(objCell.Row, IP_COL).Value = devSheet.Cells(devCell.Row, 8).Value
                        Cells(objCell.Row, NETWORKNO_COL).Value = devSheet.Cells(devCell.Row, 10).Value
                        Cells(objCell.Row, DEVINST_COL).Value = devSheet.Cells(devCell.Row, 9).Value
                    End If
                Next

            Next

            For Each objCell In rngObjNameObj
                For Each edeCell In rngObjNameEDE
                    'TODO: check if objectname got a "" then check the last string after "" and "" is equal
                    If objCell.Value = edeCell.Value And aSheet.Cells(objCell.Row, DEVINST_COL) = edeSheet.Cells(edeCell.Row, edeCell.Column - 1) Then
                        'insert objecttype, objectinstance
                        Cells(objCell.Row, TYPEINT_COL).Value = edeSheet.Cells(edeCell.Row, 5).Value
                        Cells(objCell.Row, INSTANCE_COL).Value = edeSheet.Cells(edeCell.Row, 6).Value

                        For Each enumCell In rngObjTypeEnum
                            If enumCell = "BACnet_Objecttype" Then
                                If edeSheet.Cells(edeCell.Row, 5).Value = enumSheet.Cells(enumCell.Row, 3).Value Then
                                    aSheet.Cells(objCell.Row, TYPESTR_COL).Value = enumSheet.Cells(enumCell.Row, 2).Value
                                End If
                            End If
                        Next
                    End If
                Next
            Next

            End If

            [ctl_InfoObj].Value = "BACnet data actualized"
        End If

    End Sub

```

2.7.1.17 Cmd_x_Click*Option Explicit*

```
Private Sub cmd_ModifyObj_Click()
    Call ModifyObj
End Sub
```

```
Private Sub cmd_CheckAlias_Click()
    Call CheckAlias
End Sub
```

```
Private Sub cmd_CheckObjects_Click()
    Call CheckObjects
End Sub
```

```
'Append selected objects
Private Sub cmd_AppendFromEDE_Click()
    Call AppendFromEDE
End Sub
```

```
'Modify selected lines
Private Sub cmd_ObjToAlias_Click()
    Call ObjToAlias
End Sub
```

```
Private Sub cmd_AliasToObj_Click()
    Call AliasToObjName
End Sub
```

```
'BACnet Data
Private Sub cmd_BACnetFromEDE_Click()
    Call BACnetDataEDE
End Sub
```

2.7.1.18 Funktion: Select

```
Private Sub cmd_Select_Click()
    'Variables
    Dim MinRow, i, ColOffset, SelectColumn As Long
    MinRow = 13 ' first row of the selection area
    SelectColumn = 1 ' column, where the selection can be done

    'Functional part
    If Selection.Row < MinRow Then Exit Sub

    ColOffset = SelectColumn - Selection.Column

    If Selection.Column > SelectColumn Then Selection.Offset(, ColOffset).Select

    If Selection.Areas.Count > 1 Then
        For Each i In Selection.Areas
            i.Columns(SelectColumn).Value = ChrW(&H2713) 'Unicode character Check Mark
        Next i
    Else
        Selection.Columns(SelectColumn).Value = ChrW(&H2713)
    End If

    Selection.Select

End Sub
```

2.7.1.19 Funktion: Deselect

```
Private Sub cmd_Deselect_Click()  
    'Variables  
    Dim MinRow, i, ColOffset, SelectColumn As Long  
    MinRow = 13 ' first row of the selection area  
    SelectColumn = 1 ' column, where the selection can be done  
  
    'Functional part  
    If Selection.Row < MinRow Then Exit Sub  
  
    ColOffset = SelectColumn - Selection.Column  
  
    If Selection.Column > SelectColumn Then Selection.Offset(, ColOffset).Select  
  
    If Selection.Areas.Count > 1 Then  
        For Each i In Selection.Areas  
            i.Columns(SelectColumn).Value = ""  
        Next i  
    Else  
        Selection.Columns(SelectColumn).Value = ""  
    End If  
  
    Selection.Select  
  
End Sub
```