Diplomarbeit

Visualisierung für das FTS – Leitsystem OS300 (SPS)

Stefan Soltau

August 2005

Universität Lüneburg – Fachbereich Automatisierungstechnik

Zusammenfassungsblatt zur Diplomarbeit

Titel der Diplomarbeit: Visualisierung für das FTS – Leitsystem OS300 (SPS)

Stichworte: Visualisierung, Fahrerloses Transportsystem, FTF, FTS

Zusammenfassung

Das OS300 ist ein auf SPS- basierendes Leitsystem für *Fahrerlose Transportsysteme (FTS)*. Zum Beobachten und Bedienen des Systems werden zur Zeit handelsübliche Visualisierungssysteme wie z.B. Wincc (Siemens) oder Intouch (Wonderware) eingesetzt, deren Funktionalitäten auf allgemeine Industrieanwendungen ausgerichtet sind. Die speziellen Anforderungen einer FTS - Anwendung können in der Regel nur durch einen hohen Engineeringaufwand erfüllt werden.

Im Zuge der Harmonisierung der verschiedenen FTS - Leitsysteme der E&K AUTOMATION - Gruppe wurde entschieden, die Visualisierung des FTS - Leitsystems OS800 auch für das OS300 zu nutzen. Das OS800 ist ein PC-basiertes Leitsystem, welches mit dem Echtzeitbetriebssystem RMOS zusammenarbeitet. Die Visualisierung (BuB – Beobachten und Bedienen) ist eine Windowsanwendung.

Im Rahmen der Diplomarbeit soll die bestehende Schnittstelle zwischen BuB und OS800 analysiert werden. Danach erfolgt die Umsetzung (Programmierung) dieser Schnittstelle für das OS300.

Betreuender Professor: Prof. Dr.-Ing Ralf Hadeler

Zweitprüfer: Prof. Dr. rer. nat. Helmut Faasch

Hochschule: Universität Lüneburg

Fachbereich: Automatisierungstechnik

Studiengang: Angewandte Automatisierungstechnik

Firma: E&K AUTOMATION - Eilers & Kirf GmbH

Betreuender Mitarbeiter: Dipl.-Ing. Dieter Sagewitz

Abgegeben am: 15.08.2005

Danksagung

An dieser Stelle möchte ich mich bei allen, die mich während meiner Diplomarbeitszeit, aber auch während des Studiums unterstützt haben, bedanken.

Ich bedanke mich bei der Firma **E&K - Automation Eilers & Kirf GmbH**, die es mir ermöglicht hat, meine Diplomarbeit in ihrem Hause anzufertigen.

Ein besonderer Dank gilt dabei meinem Betreuer **Herrn Sagewitz**, der mich sehr bei der Bearbeitung des Themas unterstützt hat.

Des Weiteren möchte ich mich bei **Herrn Prof. Dr. Hadeler,** für die sehr gute Betreuung von Seiten der Universität Lüneburg, und bei **Herrn Prof. Dr. Faasch,** für die Übernahme der Zweitkorrektur, bedanken.

Abschließend danke ich **meinen Eltern** für ihre Unterstützung während meiner Studienzeit.

1. Die E&K AUTOMATION Gruppe

Das Unternehmen Eilers & Kirf GmbH (EK-R), welches zur E&K AUTOMATION Gruppe gehört, ist hauptsächlich in zwei Bereichen tätig. In der Prozessautomatisierung (Abteilung CONTROL, in der die Diplomarbeit erstellt wurde) und in der Erstellung von Fahrerlosen Transportsystemen (Abteilung FTS). Im Laufe der Jahre hat das Unternehmen Eilers & Kirf GmbH mehrere Unternehmen aufgekauft. Diese Tochterfirmen sind nur auf dem FTS – Markt tätig. Das Tochterunternehmen Indumat (IND-R), mit Hauptsitz in Reutlingen, hat ein eigenes FTS – System entwickelt. In der Abteilung CONTROL werden außerdem die Leitsysteme und die Visualisierungen (speziell für Induktiv geführte FTS – Anlagen) erstellt.

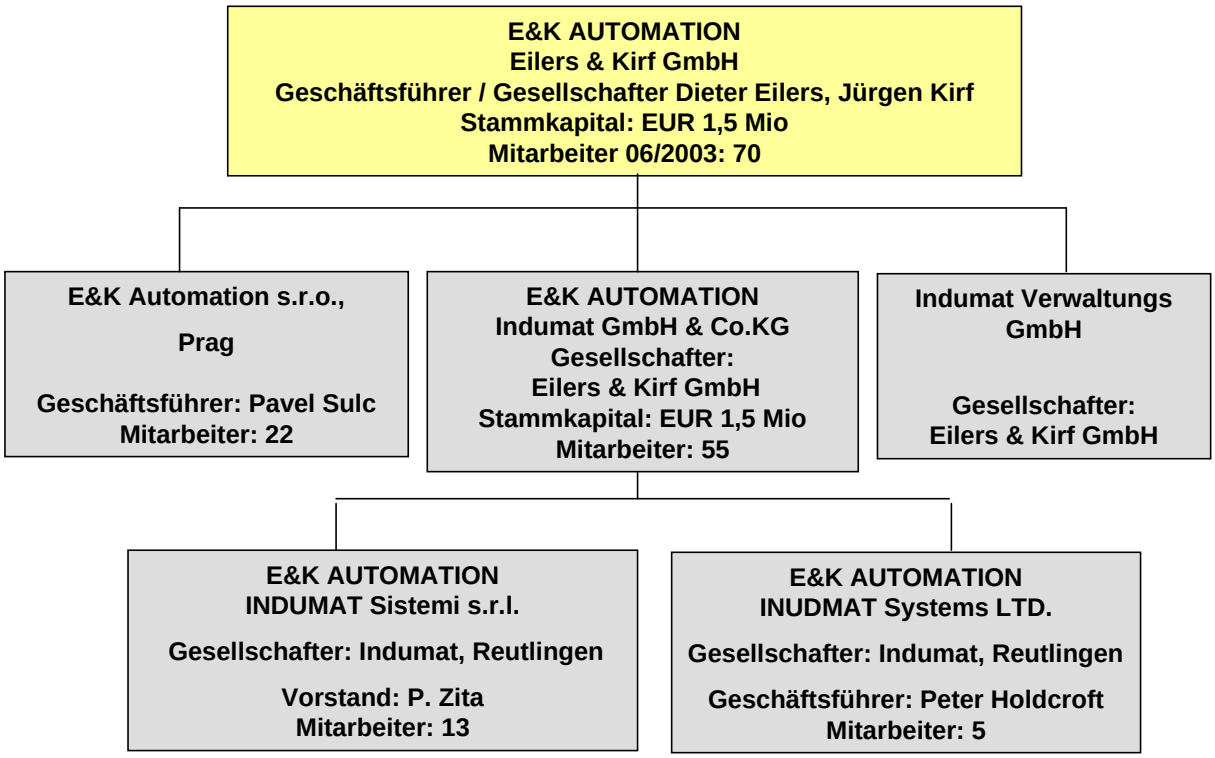


Abbildung 1 : Unternehmensstruktur – E&K Automation[1]

1.1. Problemstellung

Es sollen die Techniken der verschiedenen Standorte vereinheitlicht werden. Im Zuge dessen soll die Visualisierung des OS800 (IND-R) an das OS300 (EK-R) angebunden werden. Damit würde dem Kunden ein einheitliches Visualisierungssystem an allen Standorten angeboten werden. Bisher setzt EK-R als Visualisierung WinCC, ein universelles Visualisierungstool von Siemens, ein.

[1] E&K Firmenpräsentation; Stand 07.2005

1.2. Zielsetzung

Es soll zuerst ein Konzept, für die Realisierung eines Interfaces im OS300 zur Anbindung der Visualisierung des OS800, erstellt werden. Das Interface soll als ein in sich geschlossenes Modul realisiert werden, welches über eine interne Schnittstelle an die existierende OS300 Software angebunden wird. Eine Verzahnung mit der existierenden Software ist zu vermeiden!

Die Realisierung dieser Aufgabe teilt sich in folgende Teilaufgaben auf:

- Einarbeitung in die existierende Schnittstelle zum OS800.
  (Einweisung und Unterstützung durch erfahrene Mitarbeiter von INDUMAT)
- Sichten bzw. Ergänzen der vorhandenen Dokumentation.
- Prüfen der Kommunikationsmechanismen auf Verwendbarkeit für die Anbindung einer SPS (SIMATIC S7-300 / 400)
- Realisierung der Software für das Interfacemodul auf Seiten der SPS (STEP7-AWL Programm)
- Empfang / Interpretation von folgenden Telegrammen
  - SK – Betriebsarten Telegramm
  - AT – Auftragstelegramm
  - MA – fahrzeugbezogenes Auftragstelegramm
  - FK – Fahrzeugtelegramm
  - ZK – Stationstelegramm
  - TI – Zeit Telegramm
- Senden / Zusammenstellen von folgenden Telegrammen
  - Z0 – Anlagenzustandstele.
  - A0 – Auftragstelegramm
  - M0 – fahrzeugbezogenes Auftragstelegramm
  - F0 – Fahrzeugtelegramm
  - S0 – Stationstelegramm
  - QT – Quittungstelegramm
  - DT – SMS – Telegramm
  - EA – Digitale E/A Telegramm
  - AK – Anlagenkonstantentele.
- Versorgen / Abarbeiten der internen Schnittstelle
- Prüfen der Software mit Hilfe eines Testsystems
- Erstellen der Dokumentation[2]

[2] Pflichtenheft-Diplomarbeit Stefan Soltau.doc

2. Grundlagen

Die Schnittstelle, die im Rahmen der Diplomarbeit analysiert und programmiert werden soll, befindet sich zwischen der Visualisierung (BuB – System) und dem Leitsystem (siehe Abbildung 2) des Fahrerlosen Transportsystems.
Um die Funktion der Schnittstelle zwischen OS300 und dem BuB - System verstehen zu können, werden hier die Grundlagen etwas näher erläutert.

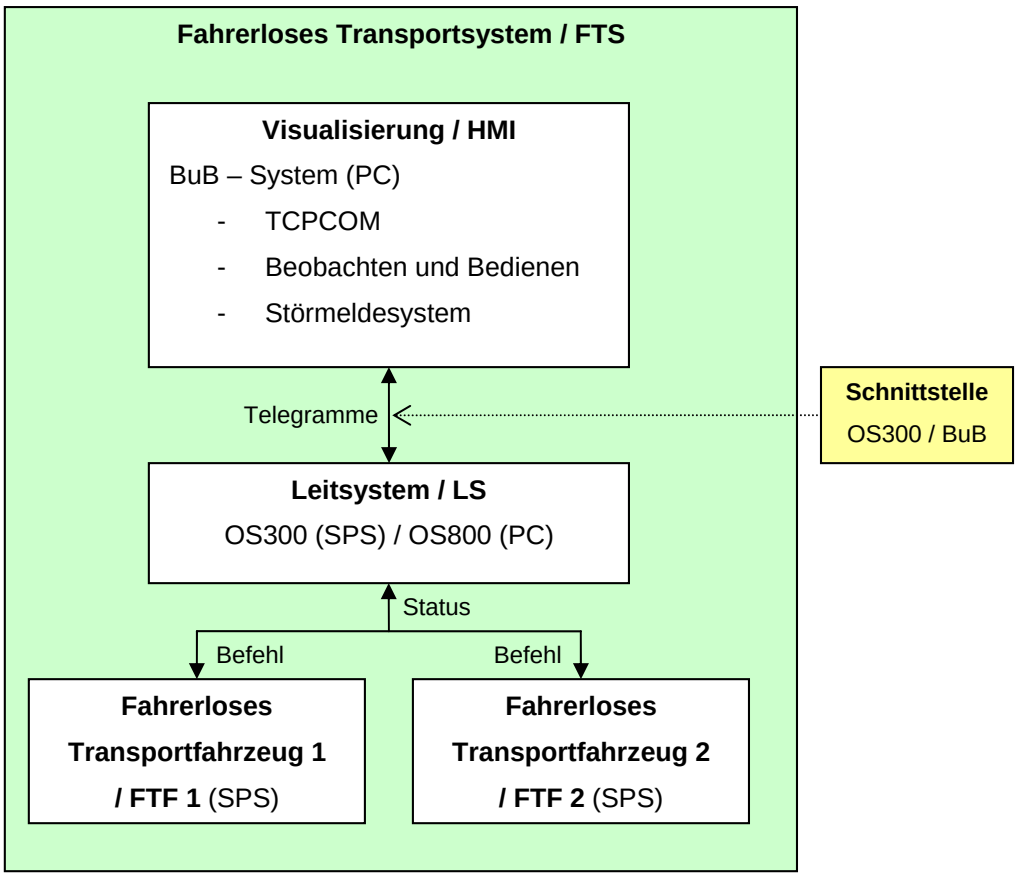


Abbildung 2 : Übersicht Grundlagen

2.1. Abkürzungen / Fachbegriffe

Abk.	Bedeutet	Erklärung
AGV	Automatic guided vehicle	Fahrerloses Transportfahrzeug (engl.)
AWL	Anweisungsliste	Assembler – ähnliche Programmiersprache in der SPS
BuB	Beobachten und Bedienen	siehe 2.4 - Das Beobachten und Bedienen
CP	communication prozessor	Kommunikationsbaugruppe der SPS
DB	Datenbaustein	Baustein der Variablen enthält (Speicher)
FC	Function	Funktion in der SPS
FTF	Fahrerloses Transportfahrzeug	siehe 2.2 - Fahrerloses Transportsystem (FTS)
FTS	Fahrerloses Transportsystem	siehe 2.2 - Fahrerloses Transportsystem (FTS)
HMI	Human Machine Interface	Visualisierung
LS	Leitsystem **à** OS	siehe 2.2 - Fahrerloses Transportsystem (FTS)
MMF	Memory Mapped File	siehe 2.4.3.1 - Die MMF
MP	Meldepunkt	siehe 2.2 - Fahrerloses Transportsystem (FTS)
OB	Operationsbaustein	OB1: Baustein, der in der SPS zyklisch aufgerufen wird
OS	Operating system - OS300 - OS800	siehe 2.2 - Fahrerloses Transportsystem (FTS) - SPS (Eilers & Kirf) - RMOS (Echtzeitbetriebssystem) INDUMAT
RMOS		Echtzeitbetriebssystem von Siemens
SMS	Störmeldesystem	Protokollieren von Fehler- und Störmeldungen
WinCC		Visualisierungstool von Siemens

Tabelle 1 : Abkürzungen

2.2. Fahrerloses Transportsystem (FTS)

Um den innerbetrieblichen Materialfluss effizient, schnell und kostengünstig zu realisieren werden häufig Fahrerlose Transportsysteme eingesetzt. Dabei ist ein FTS kein Serienprodukt, es handelt sich vielmehr um eine Sonderkonstruktion. Die Anlagen werden jeweils so konzipiert, dass sie den Wünschen und Anforderungen des Kunden entsprechen.
Ein FTS besteht aus den Kernkomponenten:

- Bodenanlage
- Fahrzeug
- Leitsystem
- Visualisierung (optional)

Für die **Bodenanlage** gibt es verschiedene Ausführungsarten. Sie wird benötigt, damit das Fahrerlose Transportfahrzeug (FTF) dem Fahrkurs folgen kann. Dieser kann

- optisch (aufgemalte Fahrspur),
- passiv induktiv (Metallstreifen),
- aktiv induktiv (Kabel mit verschiedenen Frequenzen) oder
- virtuell (Lasernavigation über Reflektoren an den Wänden) sein.

Damit dem Fahrzeug auf dem Kurs eine Position zugewiesen werden kann, sind im Boden Identgeber (Responder) eingelassen, die mit einer Zahl codiert sind (Meldepunkte). Wenn ein FTF solch einen Responder überfährt liest es diesen mit Hilfe seines Responderlesegerätes aus und meldet die ausgelesene Ziffer dem Leitsystem (LS / OS). Im Bereich zwischen zwei Meldepunkten (MP) befindet sich eine Strecke.

Abbildung 3 : FTF – Interbrew Jupille (Belgien)

Das **Fahrzeug** selbst transportiert das Transportgut (Rohmaterialien, Halb- und Fertigwaren) zwischen den Bestimmungsorten (Stationen). Hierbei unterscheidet man zwischen Hole- und Bringezielen. Der Lastwechsel zwischen einer Station und dem FTF kann über verschieden Wege / Techniken ablaufen. Es gibt zum Beispiel Fahrzeuge die über eine Palettengabel verfügen. Die Fahrzeuge verfügen außerdem über Personenschutzeinrichtungen, wie Schaumstoffbumper oder Laserscanner, zur Bereichsüberwachung, damit kein Mensch zu Schaden kommen kann.

Da das **Leitsystem** (LS / OS) durch die Responder weiß wo sich ein FTF befindet, kann es eine Verkehrsregelung übernehmen, indem es bestimmte Strecken für die Durchfahrt sperrt bzw. freigibt.
Des Weiteren übernimmt das OS die Auftragsdisposition, das bedeutet, dass das OS jeweils dem Fahrzeug einen Transportauftrag zuteilt, welches sich am dichtesten am Holeziel befinden und noch keinen Auftrag hat.
Ein Auftrag kann über verschieden Wege ausgelöst werden. Dies ist durch Ruftaster, Belegtmelder an den Stationen oder auch durch einen übergeordneten Lagerverwaltungsrechner möglich. Außerdem können auch Aufträge durch das HMI (HMI – **H**uman **M**achine **I**nterface, ehemals MMI – Men Machine Interface) generiert werden.
Eine weitere Aufgabe des Leitsystems ist es, die Fahrzeuge über den Fahrkurs zu führen. Das OS beinhaltet eine Logik welche immer den kürzesten Weg zwischen den Zielen berechnet.
Nachdem sich ein FTF an einem Meldepunkt gemeldet hat berechnet das OS den Weg für das jeweilige FTF. Nach erfolgreicher Berechnung sendet das OS per Telegramm einen Befehl für die aktuelle Strecke, welcher die Fahrgeschwindigkeit (in Kurven langsam), die aktuelle Frequenz der gefolgt werden soll sowie einige andere Befehle wie Blinken, Hupen oder ähnliches beinhalten kann, an das FTF.

Um den ganzen Prozess zu überwachen und zu steuern wird häufig eine **Visualisierung** eingesetzt. Damit ist es möglich manuelle Aufträge zu generieren, auftretende Fehler sowie den Anlagenstatus anzuzeigen.

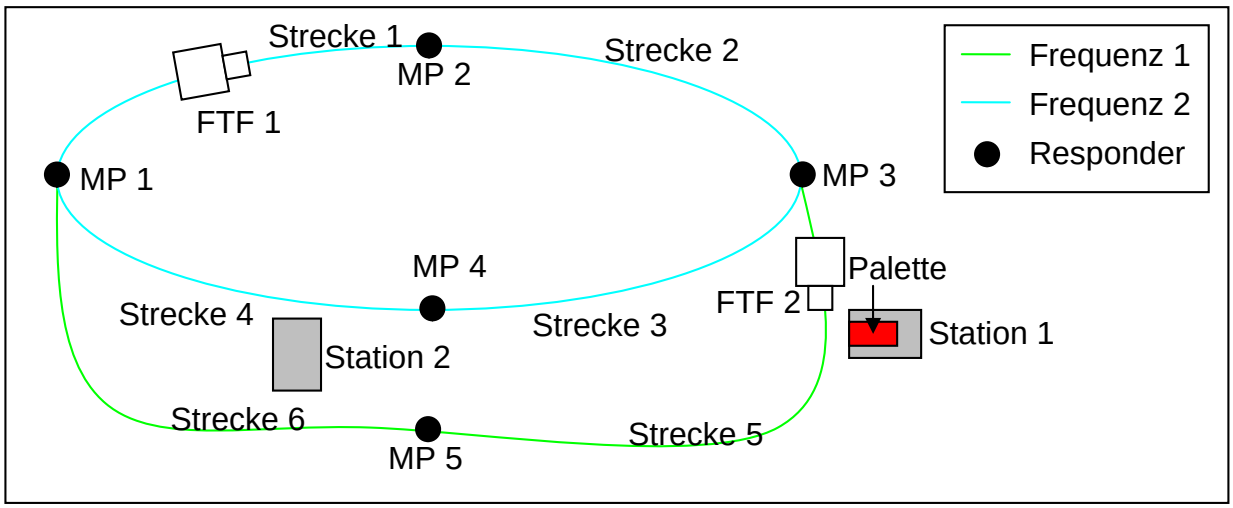


Abbildung 4 : Prinzip eines FTS

In Abbildung 4 kann man zwei Stationen erkennen, wobei die Station 1 zurzeit mit einer Palette beladen ist. Als Beispiel wird nun ein Auftrag generiert mit dem die Palette von Station 1 zu Station 2 gebracht werden soll.

Zu diesem Zeitpunkt würde FTF 2 den Auftrag vom OS bekommen an Station 1 (Holeziel) die Palette aufzuladen. Anschließend würde das den Befehl an das Fahrzeug schicken, über Meldepunkt 5, bis zum Meldepunkt 1 zu fahren, wenn Strecke 5,6 und 1 (immer auch eine im voraus) zurzeit nicht von einem anderem FTF befahren wird. Am *MP 1* würde das FTF außerdem den Befehl bekommen die Frequenz zu wechseln, also auf *Frequenz 2*. Danach würde es Befehle bekommen, bis zum MP4 zu fahren. Am MP 4 würde das OS schließlich den Befehl zur Lastabgabe an Station 2 geben. Damit das FTF jedoch soweit fahren kann, ist es nötig, dass das FTF 1 nicht mehr im Wege steht.

2.2.1. Testanlage E&K

Um Fahrzeuge vor der Inbetriebnahme testen zu können hat E&K eine Test-Bodenanlage. Diese wurde für die BuB gewählt, damit sie später für diesen Kurs benutzt werden kann. Dieser Testaufbau hat

- 16 Meldepunkte,
- 21 Strecken,
- 3 Stationen, davon
    - 2 Universalstationen (Holen und Bringen) (Erni / Bert),
    - 1 Warteplatz (Idle),
- 2 Fahrzeuge des selben Typs und
- 5 Frequenzen.

In Abbildung 5 sieht man den Hallenumriss (rosa), den Fahrkurs (türkis), die Stationen (gelb) und die einzelnen Meldepunkte (rot) der Testanlage von E&K-R.

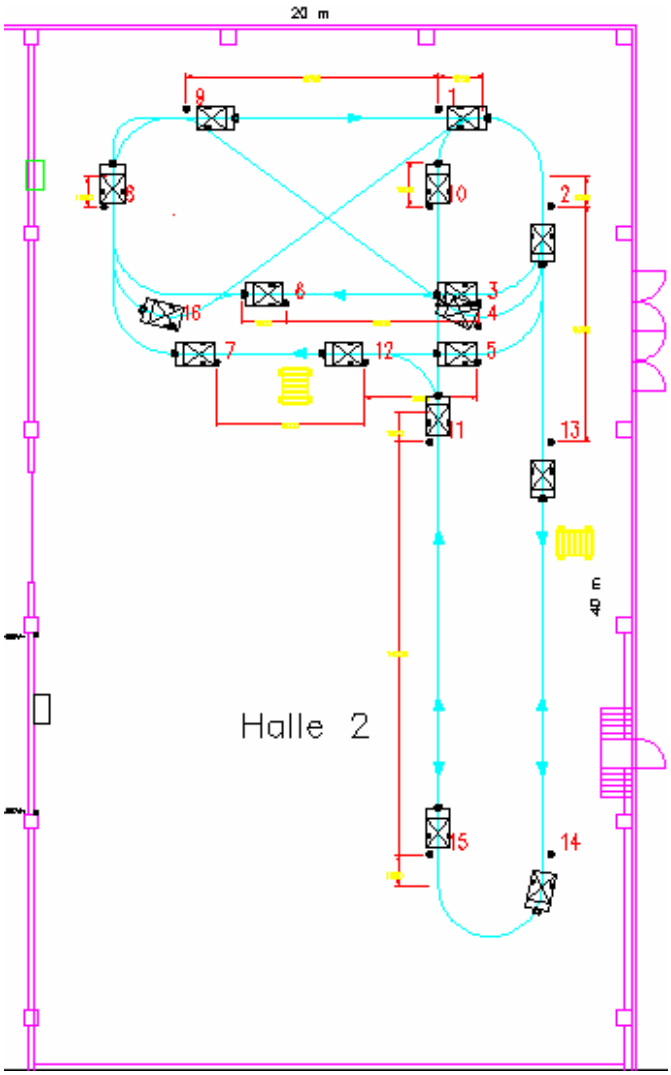


Abbildung 5 : Bodenanlage – Testanlage E&K

2.3. Speicherprogrammierbare Steuerungen (SPS)

Das Unternehmen E&K hat es sich zur Aufgabe gesetzt die Steuerungen in den FTF und dem OS möglichst mit SPSen zu realisieren. Hierfür werden hauptsächlich Steuerungen von Siemens benutzt.

Eine SPS ist ein modular aufgebautes Automatisierungsgerät, welches Eingänge, Ausgänge und innere Zustände bearbeitet und mit diesen Ergebnissen Ausgänge ansteuert. Eine SPS arbeitet zyklisch (siehe Abbildung 6), das heißt, dass am Anfang eines Zyklusses die Eingänge und Ausgänge in ein Prozessabbild eingelesen werden. Danach erfolgt die Abarbeitung des Programms. Darauf folgend wird das Prozessabbild auf die Ausgänge geschrieben. Am Ende werden interne Funktionen ausgeführt (Aktualisierung der Timer, etc.). Der Zyklus kann durch Interrupts unterbrochen werden. Hierbei gibt es sowohl Eingänge, die einen Interrupt auslösen können, als auch zeitgesteuerte Interrupts, z.B. für Regelungen alle 10ms.

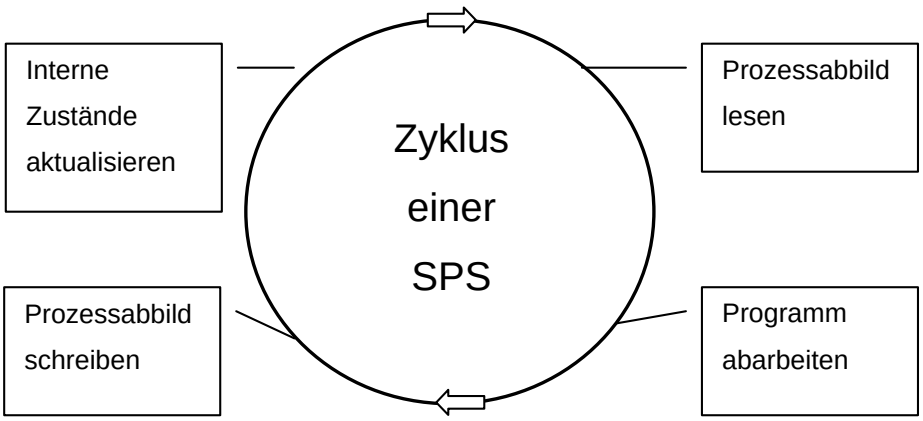


Abbildung 6 : Zyklus einer SPS

Zum Programmieren einer SPS gibt es verschiedene Programmiersprachen, welche der IEC 1131-3 Norm entsprechen. Einen Überblick über die verschiedenen Programmiersprachen bietet die Tabelle 2. Im Unternehmen E&K wird derzeit nur AWL eingesetzt. Um Programme besser strukturieren zu können gibt es verschiedene Arten von Bausteinen:

Operationsbausteine (OB): OB 1 wird zyklisch abgearbeitet. Aus ihm werden die eigentlichen Funktionen gestartet

Funktionen (FC): beinhalten den auszuführenden Programm-Code

Datenbausteine (DB): Datenspeicher in der SPS, der strukturiert und mit Symbolik versorgt werden kann

Abbildung 7 : S7-314C-2PTP[3]

Text basierende Programmierung	
Strukturierter Text (ST) Structured Text (ST)	Anweisungsliste (AWL) Instruction List (IL)

Grafische Programmierung		
Kontaktplan (KOP) Ladder Diagram (LD)	Funktionsbaustein Sprache (FBS) Function Block Diagram (FBD)	Ablaufsprache (AS) Sequential Function Chart (SFC)

Tabelle 2 : Programmiersprachen nach IEC 1131-3[4]

[3] www.e-technik.fh-kiel.de/ regiue/XSPS.html; 20.05.2005; 12:30 Uhr

[4] IEC 61131 – Wozu?, Ingo Rolle (Hrsg.),1998

- **Strukturierter Tex**t ist eine Pascal - ähnliche Hochsprache. Hierbei ist es möglich Schleifen (FOR, WHILE) zu bilden und Abfragen (IF, CASE) zu programmieren. Beispiel: A1 = (E1 U /E2) ∩ E3
  A1 := (E1 OR NOT E2) AND E3

- **AWL** ist eine Assembler - ähnliche Sprache. Viele Funktionen, z.B. Schleifen müssen mit Zählervariablen, Abfragen und Sprüngen selber programmiert (entwickelt) werden. Beispiel: A1 = (E1 U /E2) ∩ E3

```
U(
O   E1          // ODER (hier nur zum laden des Wertes in den AKKU 1)
ON    E2        // ODER_NICHT (AKKU1 = AKKU 1 ODER /AKKU 2)
)
U   E3          // UND
=   A1
```

- **Kontaktplan** ist eine Art Stromlaufplan.
  Beispiel: A1 = (E1 U /E2) ∩ E3

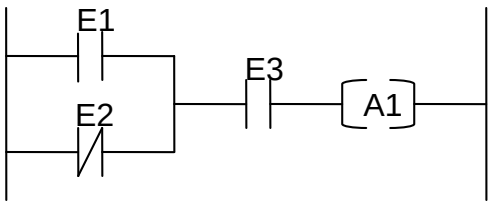


Abbildung 8 : Beispiel Kontaktplan

- **Funktionsplan**
  Beispiel: A1 = (E1 U /E2) ∩ E3

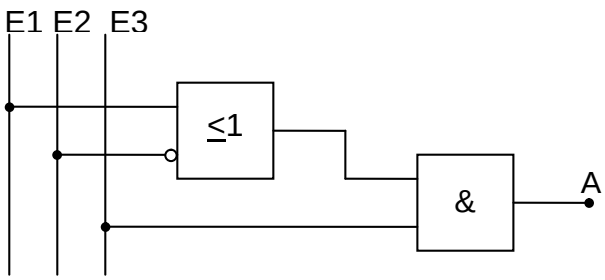


Abbildung 9 : Beispiel Funktionsplan

2.4. Das Beobachten und Bedienen System

Das Beobachten und Bedienen – System ist das Visualisierungstool des OS800 von INDUMAT. Dieses wurde speziell für das Beobachten und Bedienen von FTS – Anlagen erstellt.

Das BuB – System besteht aus den Komponenten *TCPCOM* (sorgt für die Kommunikation mit dem Leitsystem), *Störmeldesystem* (Protokollieren von Fehler- und Störmeldungen) und dem *Beobachten und Bedienen* (Visualisierung und Steuerung der Anlage). Die einzelnen Komponenten kommunizieren untereinander über Memory Mapped Files.

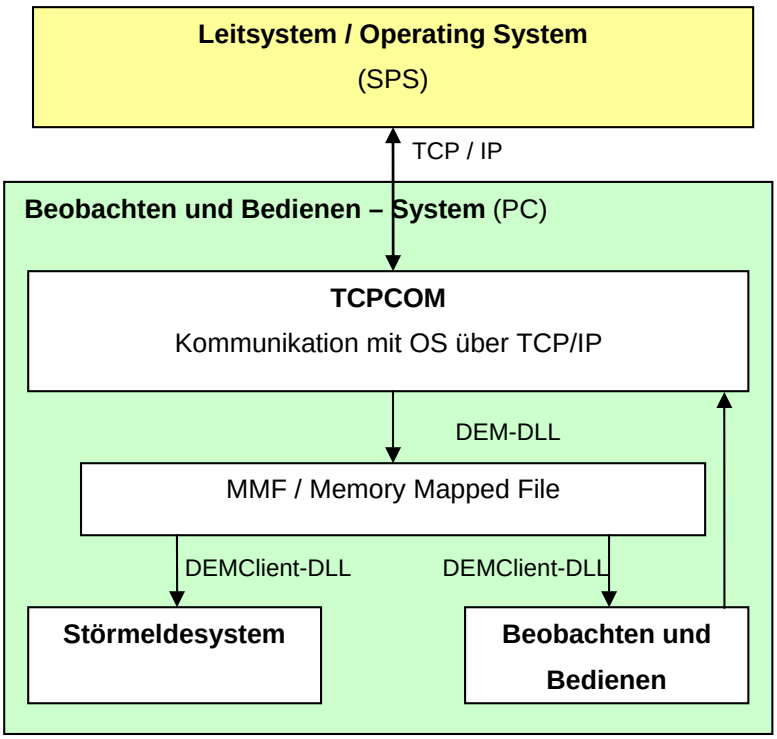


Abbildung 10 : BuB - System – Gesamtübersicht

2.4.1. Beobachten und Bedienen (BuB)

Die **B**eobachten **u**nd **B**edienen – Komponente (BuB) bietet die Möglichkeit, sich verschiedene Tabellen und Ansichten anzeigen zu lassen. Mit dem BuB – System kann sowohl beobachtet, als auch bedient werden.

Die BuB erhält ihre Daten aus der Komponente TCPCOM. Die Art und Ablage der Daten wird in der Beschreibung zu diesem Modul näher erläutert.

In Abbildung 11 kann man das Hauptfenster des BuB erkennen, aus dem die einzelnen Tabellen und Ansichten gestartet werden können

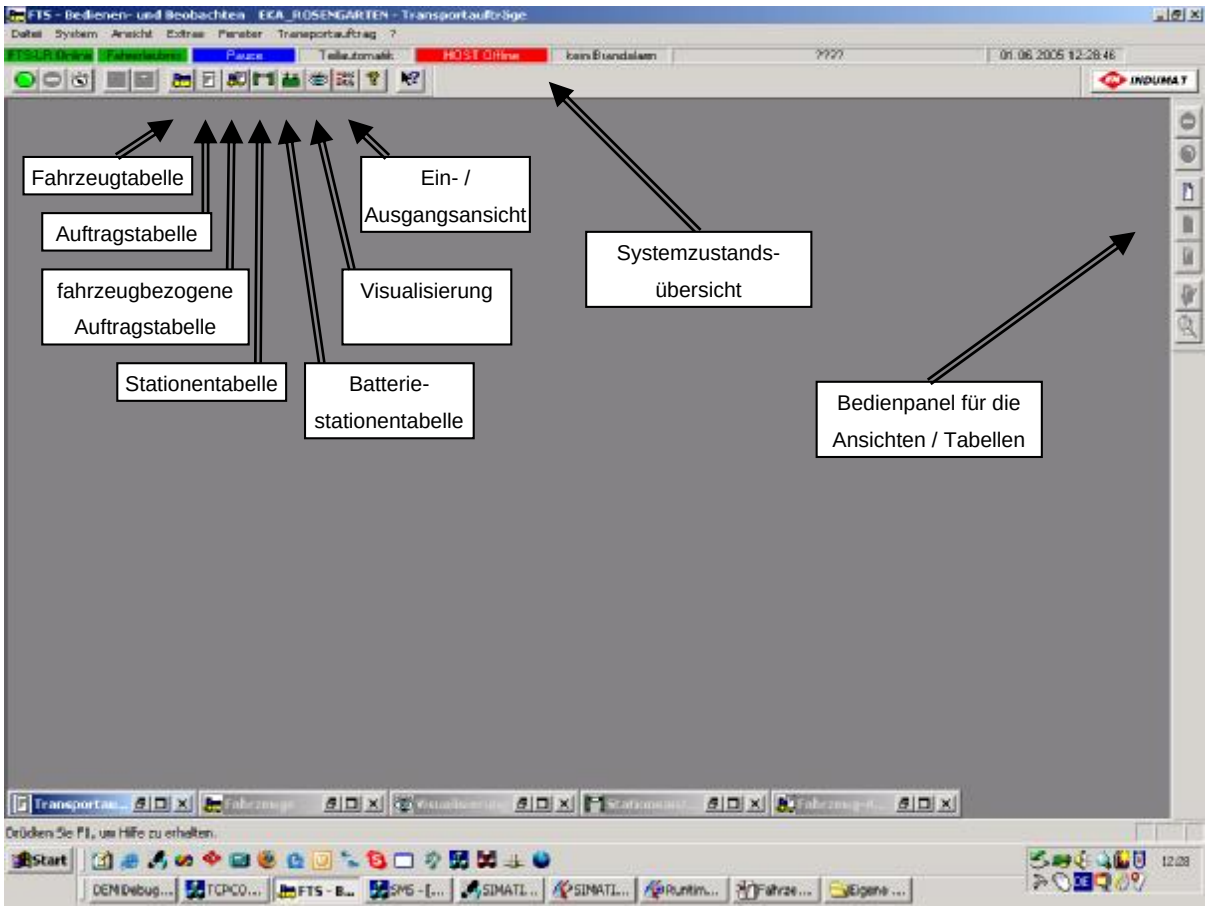


Abbildung 11 : Hauptfenster der BuB

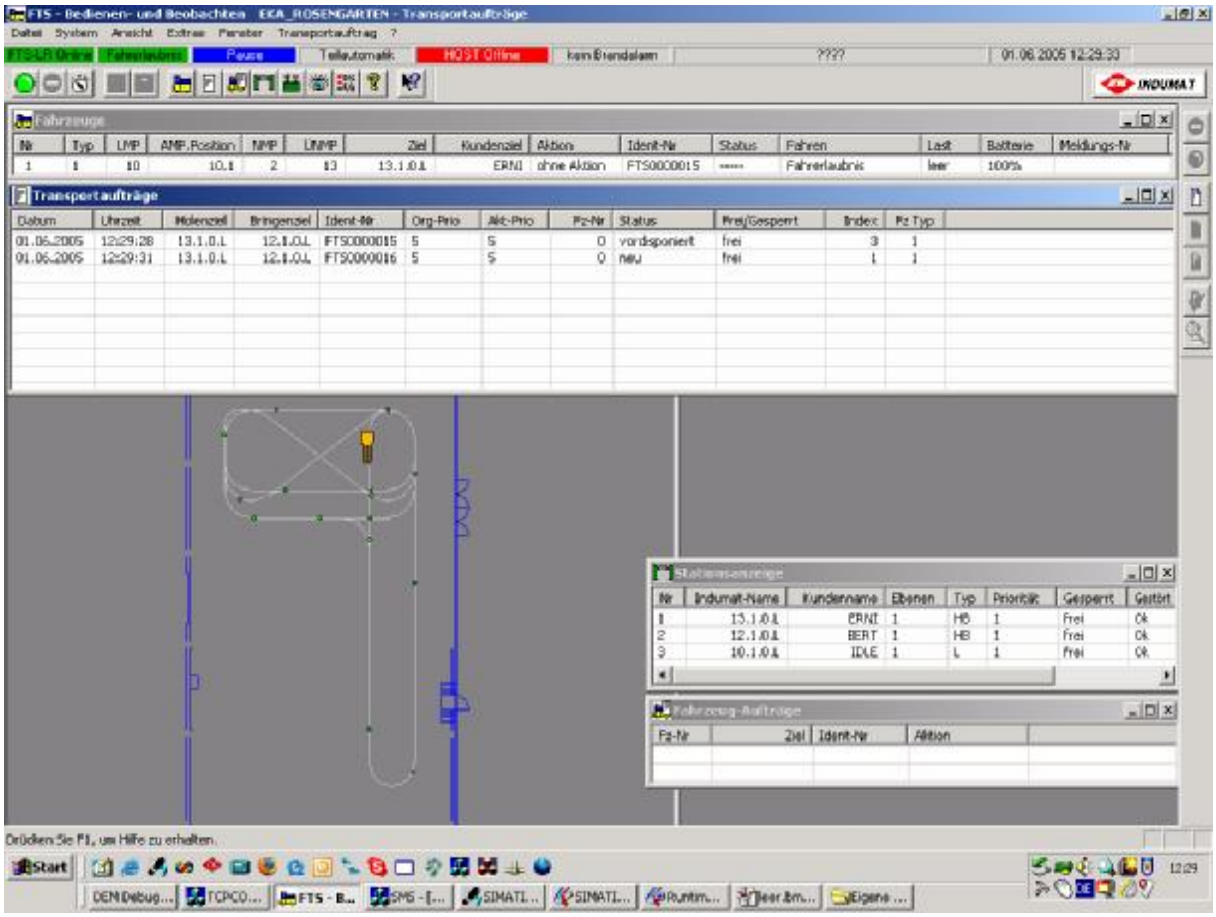


Abbildung 12 : Beobachten und Bedienen - Gesamtübersicht

In Abbildung 12 kann man das Hauptfenster, mit einigen geöffneten Tabellen / Ansichten, erkennen. In den Tabellen werden die Daten der Anlage (zum Beispiel die Transportaufträge) angezeigt. Außerdem sind wichtige Anlageninformationen in der „Systemszustandsübersicht“ (siehe Abbildung 11) immer sichtbar:

- FTS-LR (Online / Offline)
- Fahrzustand (Fahrerlaubnis / -verbot)
- Systemzustand (Pause / Betriebsbereit / Synchronisation)
- Automatikmodus (Teil- / Vollautomatik)
- Hostmodus (Online / Offline)
- Brandalarmstatus (kein Brandalarm / Brandalarm)
- Sammelalarmübersicht
- Uhrzeit / Datum

2.4.1.1. Die Fahrzeug - Tabelle

Am rechten Rand befindet sich immer das Bedienpanel (Abbildung 14), mit dem man die Daten der entsprechenden Liste ändern kann. In der Fahrzeug – Tabelle (Abbildung 13) werden die aktuellen Daten der Fahrzeuge angezeigt:

Nr:	Fahrzeugnummer
Typ:	Fahrzeugtyp
LMP:	Letzter Meldepunkt
AMP.Position:	Aktueller Meldepunkt & Position am Meldepunkt
NMP:	Nächster Meldepunkt
ÜNMP:	Übernächster Meldepunkt
Ziel:	Akt. Fahrziel: Meldepunkt.Position.Ebene.Lage
Kundenziel:	Kundenname des Zieles
Aktion:	Derzeitige Aktion des Fahrzeuges (holen, bringen, etc.)
Ident-Nr:	Auftragsnummer des aktuellen Auftrages
Status:	Batterie voll, nicht im System, etc.
Fahren:	Fahrzustand (Fahrerlaubnis, -verbot)
Last:	Laststatus (leer, beladen)
Batterie:	Ladestatus der Batterie in Prozent
Meldungsnummer:	Nummer einer Fehlermeldung

Abbildung 13 : Fahrzeuge – Tabelle

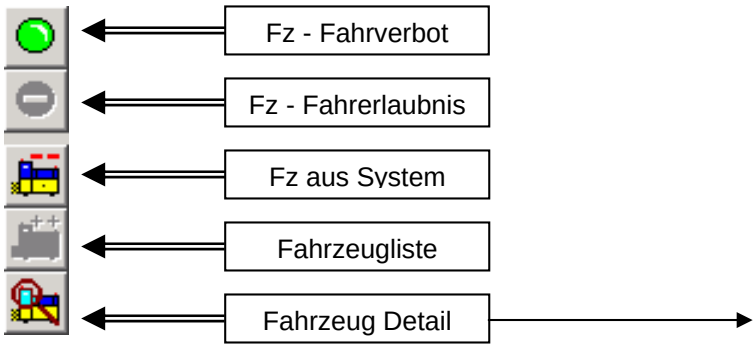


Abbildung 14 : Fahrzeug – Tabelle - Bedienpanel

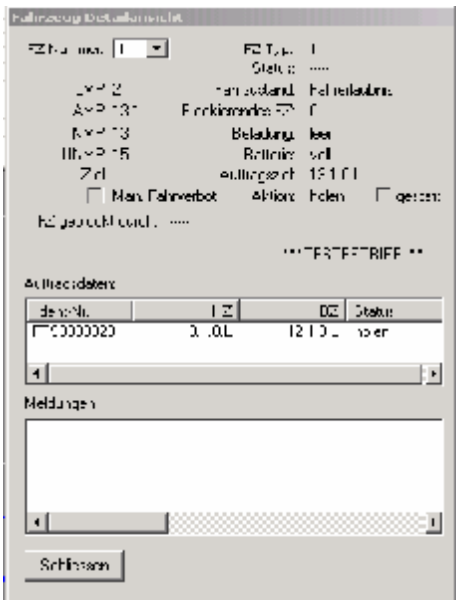


Abbildung 15 : Fahrzeug – Tabelle - Detailansicht

2.4.1.2. Die Transportaufträge - Tabelle

Als Auftrag wird in der BuB jeder Auftrag mit Hole- und Bringeziel angezeigt.

Datum:	Datum, an dem der Auftrag ausgelöst wurde
Uhrzeit:	Uhrzeit, an der der Auftrag ausgelöst wurde
Holenziel:	Station, an der die Last geholt werden soll
Bringenziel:	Station, zu der die Last gebracht werden soll
Ident-Nr:	Auftragsnummer des Auftrages
Org-Prio:	Originalpriorität des Auftrages
Akt-Prio:	Aktuelle Priorität des Auftrages
Fz-Nr:	Fahrzeug, welches den Auftrag bearbeitet
Status:	Status des Auftrages (vor-, disponiert, neu)
Frei / Gesperrt:	Zustand des Auftrages (gesperrt / frei)
Index:	Indexnummer des Auftrages
Fahrzeugtyp:	Für welchen Fahrzeugtyp der Auftrag ist

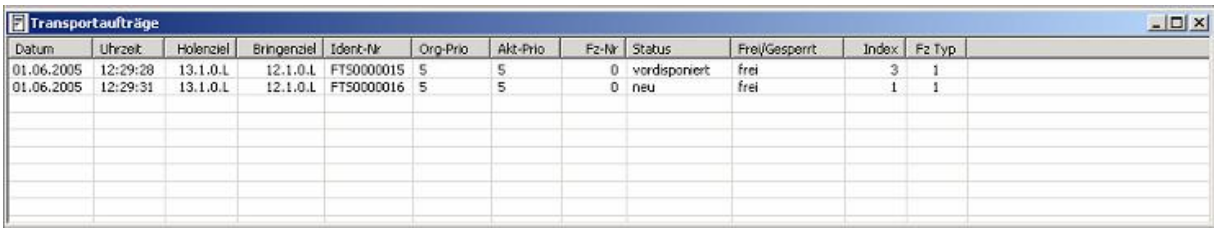


Abbildung 16 : Transportaufträge

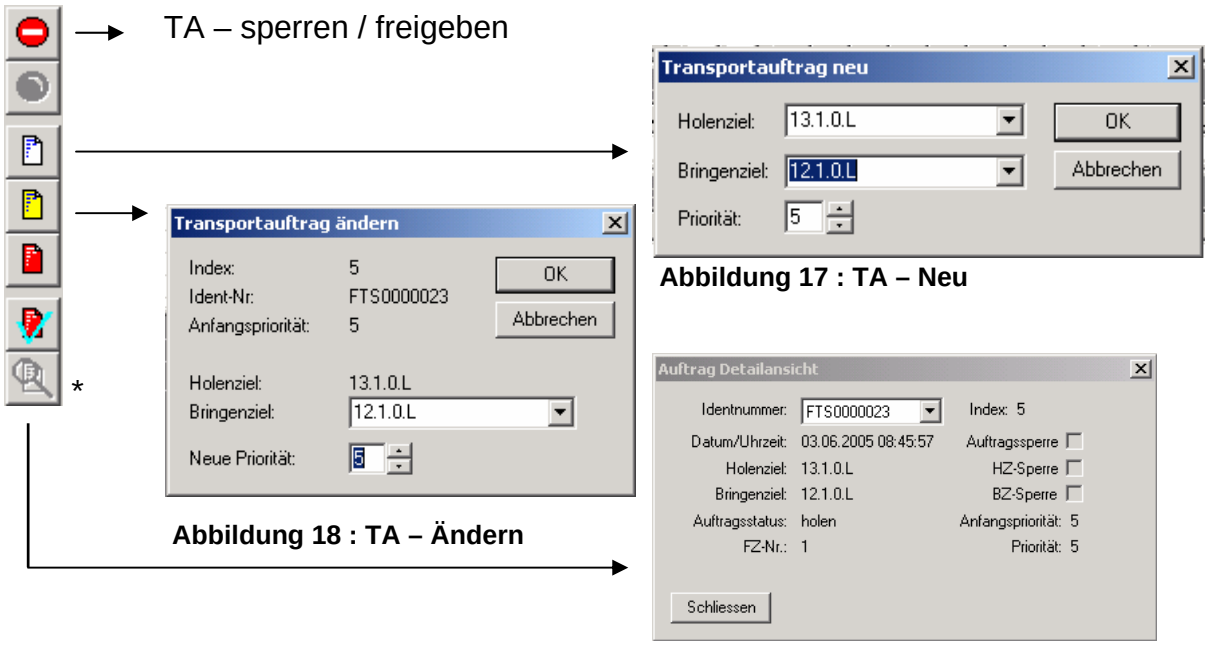


Abbildung 17 : TA – Neu

Abbildung 18 : TA – Ändern

Abbildung 19 : TA – Details

*Abbildung 20 : TA – Bedienpanel**

2.4.1.3. Die Fahrzeug – Aufträge - Tabelle

Alle Fahrten des Fahrzeuges die kein Hole- und Bringeziel haben (Testfahrt, Warteplatz, Batterie laden, etc.), werden in der „Fahrzeug-Aufträge“ Tabelle (Abbildung 21) angezeigt. Jedes FTF kann nur einen Fahrzeugauftrag zurzeit haben. Es können auch „Fahrzeug-Aufträge“ per Hand geändert, neu erstellt oder gelöscht werden (Abbildung 22).

Fz-Nr: Nummer des Fahrzeuges
Ziel: Ziel des Auftrages
Ident-Nr: Auftragsnummer des Auftrages (MAN00xxxxx)
Aktion: Aktion, die am Ziel ausgeführt werden soll (Lastaufnahme, -abgabe)

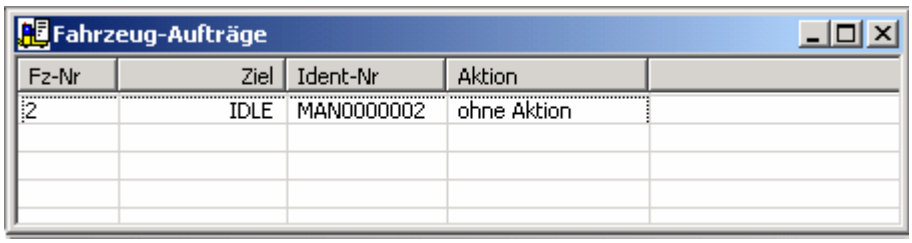


Abbildung 21 : BuB – Fahrzeug-Aufträge

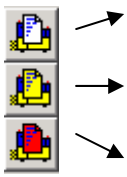


Abbildung 22 : BuB – Fahrzeug – Aufträge – Bedienpanel

2.4.1.4. Die Stationen - Tabelle

In der Stationsanzeige (Abbildung 23) werden alle Stationen der Anlage angezeigt. Dazu gehören die Hole-, Bringe-, Universal- und Batterieladestationen.

Nr:	Stationsnummer
Indumat-Name:	MP.POS.EBENE.LAGE der Station
Kundenname:	Kundenname für die Station
Ebenen:	Anzahl der Ebenen, die das Ziel hat
Typ:	Art der Station (H = Holeziel, B= Bringeziel, L = Ladestation)
Priorität:	Anfangspriorität, wenn ein Auftrag an der Station ausgelöst wird
Gesperrt:	Zustand der Station (frei / gesperrt)
Gestört:	Fehlerstatus der Station (OK / gestört)

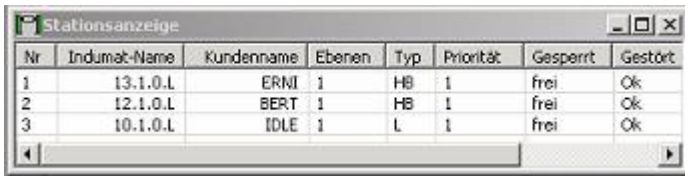


Abbildung 23 : BuB – Stationsanzeige

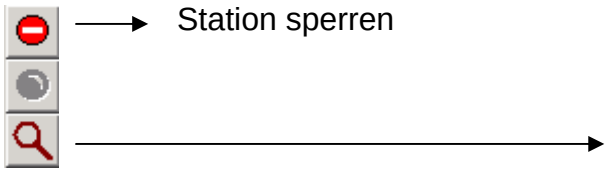


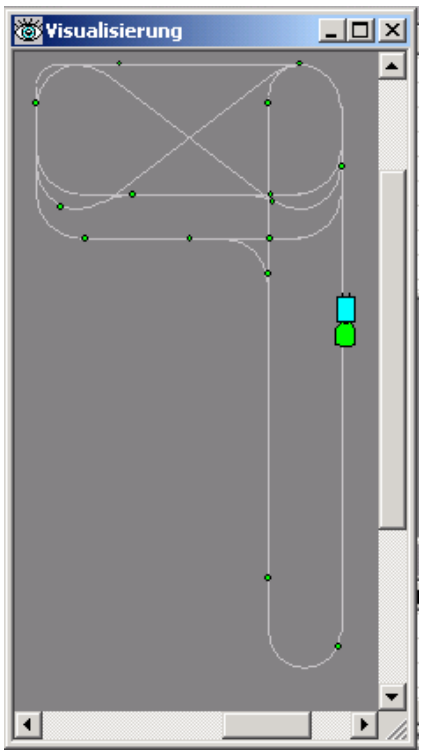
Abbildung 24 : Station – Bedienpanel

Abbildung 25 : Station – Detailansicht

Mit Hilfe des Bedienpanels (Abbildung 24) kann man Stationen sperren und freigeben. Außerdem kann man die Detailansicht (Abbildung 25) der jeweiligen Station öffnen, um erweiterte Informationen über die Station zu bekommen.

2.4.1.5. Die Visualisierung

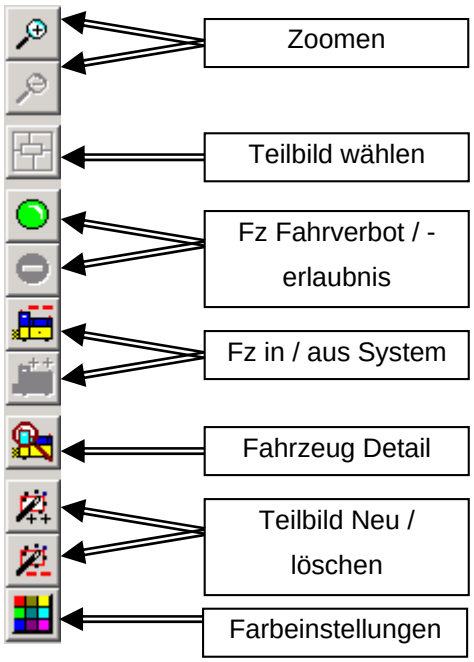
Die Visualisierung dient zum Anzeigen der Fahrzeugposition sowie zum Anzeigen des Fahrzeug- und Stationsstatus. Beim Anklicken der Fahrzeuge öffnet sich ein Detail – Dialog – Fenster (Abbildung 15). Bei der Visualisierung der Fahrzeuge gibt es zwei unterschiedliche Arten:



- Kontinuierlich: Die aktuelle Position des Fahrzeuges auf dem Kurs wird mit Hilfe der Streckendaten (Länge, Geschwindigkeit) interpoliert, so dass sich das Fahrzeug auf dem visualisierten Kurs kontinuierlich bewegt. Dies ist jedoch nur bei Anlagen möglich, deren Daten mit Hilfe von *Kontrast 2* erstellt wurden (keine Induktiv geführten Anlagen (Testkurs E&K)).

- Diskret: Das Fahrzeug springt von Meldepunkt zu Meldepunkt, da keine detaillierten Streckendaten (Kontrast 1) vorliegen.

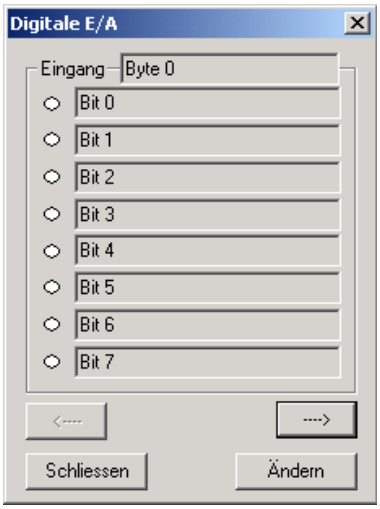
Abbildung 26 : BuB – Visualisierung



Mit Hilfe des Bedienpanels (Abbildung 27) kann man einige Funktionen der Anlage steuern.
Es ist möglich, in das Bild herein oder aus ihm herauszuzoomen. Man kann Teilbilder erstellen, löschen und auswählen, um schnell zu wichtigen Punkten der Anlage zu springen. Außerdem kann man die Funktionen der Fahrzeuge bedienen.

Abbildung 27 : BuB – Visualisierung – Bedienpanel

2.4.1.6. Die Digitale E/A - Anzeige



Im „Digitale E/A“ – Fester werden die aktuellen Zustände der digitalen Ein- und Ausgänge angezeigt.

Abbildung 28 : BuB – Digitale E/A

2.4.2. Störmeldesystem (SMS)

„Die SMS Komponente dient der Anzeige von Anlagenmeldungen (Fehler- und Statusmeldungen) und deren Verwaltung in Form von Log – Dateien“ [5]. Diese Log – Dateien können dann in „EK – Report“ (Service und Analysetool) importiert und angezeigt werden. Alle Meldungen haben eine Modul- und eine Meldungsnummer. Die Kombination aus beidem ergibt eine eindeutige Meldung. Diese Meldungen werden zurzeit vereinheitlicht, so dass alle Standorte dieselben Meldungen benutzen.

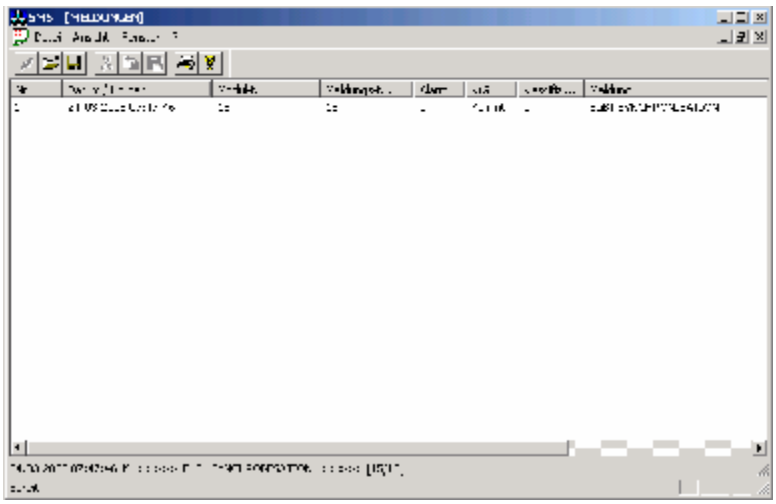


Abbildung 29 : SMS – Störmeldesystem

[5] Systemdokumentation : „BuB – Beobachten und Bedienen“

2.4.3. Kommunikationskomponente (TCPCOM)

Das Modul TCPCOM sorgt für den Datenaustausch zwischen dem BuB – System und dem Leitsystem. Die Kommunikation zwischen BuB und Leitsystem findet mittels TCP / IP - Protokoll statt. Hierbei ist das Leitsystem der Server und TCPCOM der Client, welcher die Verbindung aufbaut. Zurzeit können am Leitsystem OS300 (SPS) 2 BuB – Systeme parallel laufen. Im Rahmen der Optimierungsphase soll diese Anzahl erhöht werden.

Um eine Verbindung aufbauen zu können, muss im Konfigurationsmenü (Abbildung 30) von TCPCOM sowohl IP-Adresse des Servers, als auch einer der konfigurierten Ports eingestellt werden. Als letztes muss jeder BuB noch eine PC – Nr. zugewiesen werden. Hierbei ist die Nummer 1 der PC, welcher für die Synchronisation der Uhrzeit sorgt.

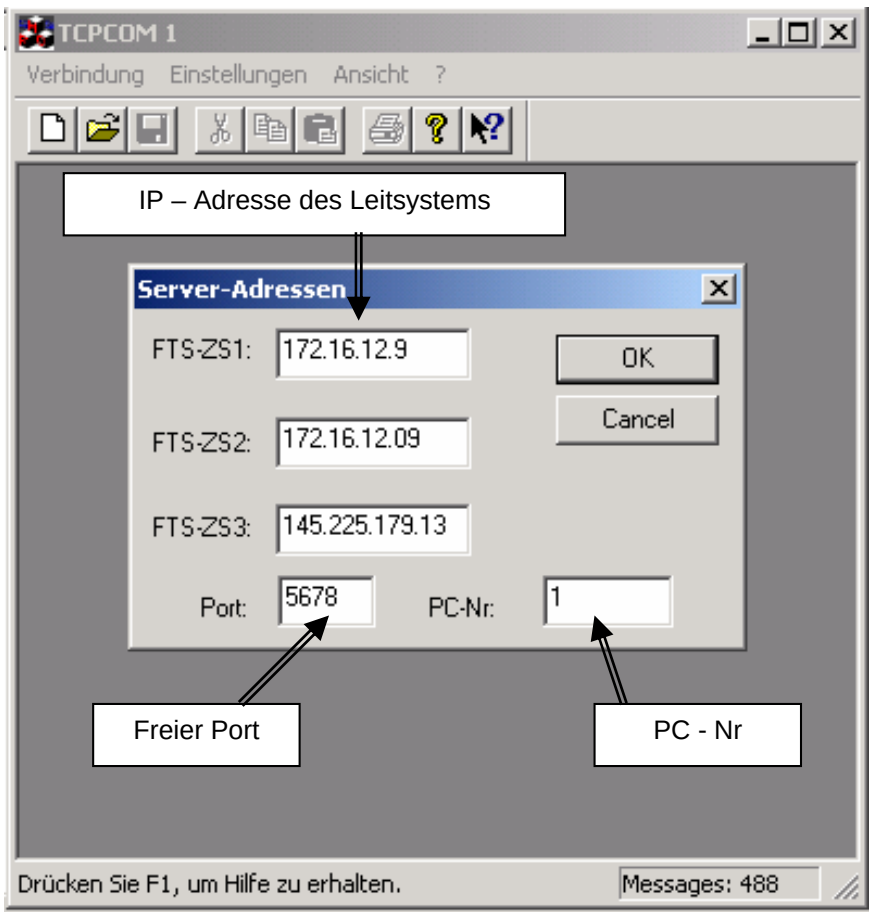


Abbildung 30 : TCPCOM – Konfiguration

Nachdem alles richtig eingestellt wurde, sollte TCPCOM eine Verbindung zum Leitsystem aufnehmen. Wenn eine Verbindung aufgebaut ist, ist keine weitere Konfiguration mehr möglich.

Um die Konfiguration zu ändern, muss man das Log-Fenster schließen und bis zur Wiederherstellung der Verbindung seine Veränderungen tätigen.

Bei erfolgreicher Verbindung erscheint, wie in Abbildung 31 gezeigt, ein Log-Fenster, in dem man die aktuellen Telegramme erkennen kann.

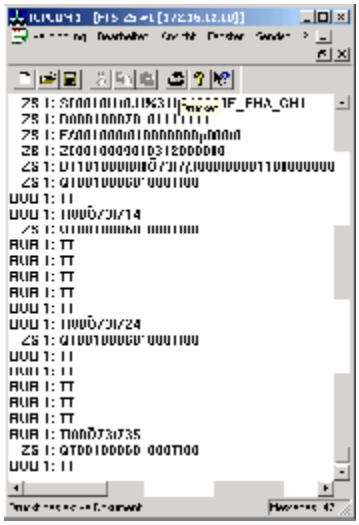


Abbildung 31 : TCPCOM – Log – Fenster

Die empfangenen Daten werden in ein MMF (**M**emory **M**apped **F**ile - dynamisch angeforderte Shared Memory Bereiche) abgelegt, in dem sich die einzelnen Fenster (Komponenten) anmelden, damit sie die enthaltenen Nutzdaten auslesen können. TCPCOM selbst hat keine Kenntnis vom Inhalt und der Struktur der Nutzdaten, die vom und zum OS geschickt werden.

Die Nutzdaten werden von TCPCOM in Steuerungsdaten (Tabelle 3) eingerahmt.

Symbol	Variablen-Typ	Erklärung
nTeleLaenge	unsigned short	Länge der gültigen Nutzdaten (in Element nData)
nRetStatus	unsigned short	wird nicht verwendet
nOnOff	unsigned short	wird nicht verwendet
nData	unsigned char [255]	Nutzdaten (feste Länge)
nPcNr	unsigned char	Nummer des PCs, nur beim Versenden relevant

Tabelle 3 : TCPCOM : Steuerungsdaten[6]

TCPCOM hat zusätzlich noch die Aufgabe, den Datenabgleich mit dem Leitsystem anzustoßen (SY – Telegramm). Dies ist die einzige Abhängigkeit zum Leitsystem, ansonsten könnte TCPCOM mit jedem TCP/IP-Server, der die o.g. Telegrammstruktur einhält, eine Kommunikation aufbauen.

[6] Systemdokumentation : „BuB – Beobachten und Bedienen“

2.4.3.1. Die MMF

„Zum Datenaustausch zwischen den Komponenten wird die Interprozesskommunikations-Komponente (IPC-Komponente) benutzt. Diese beinhaltet ein DLL-„Paar“. Dieses schreibt und liest aus **M**emory **M**apped **F**iles. TCPCOM bedient sich der DEM-DLL, die restlichen BuB-Komponenten benutzen die DEMClient-DLL um auf die Daten des MMF zuzugreifen. Es findet also kein direkter Datenaustausch zwischen den Komponenten statt. (siehe Abbildung 10)

Die DEM-DLL hat aus der Sicht von TCPCOM folgende Aufgaben:

1. Ablegen von Datenelementen in eine MMF. Diese Elemente können dann über die DEMClient-DLL ausgelesen werden.
2. Sammeln von ausgehenden Datenelementen und Bereitstellung einer Zugriffsfunktion auf diese.
3. Versenden von Update-Nachrichten an die BuB - Komponenten bei Änderungen an den MMFs.

Jedem Datenelement, das von der DEM-DLL in einem MMF abgelegt werden soll, muss ein Header vorangestellt werden. Dieser Header steuert die Position des Datenelementes. Die Identifikation der generierten MMFs erfolgt durch ein Kürzel (ID) mit einer Länge von 2 Byte. Diese ID wird als Name für das MMF verwendet. Sollte ein Datenelement in seinem Header eine ID tragen, die noch nicht in Form eines MMFs auf dem System existiert, dann wird dieses MMF zunächst generiert und anschließend die Daten eingetragen. Ein gezieltes Löschen eines MMFs ist momentan nicht möglich – beim Beenden der DEM-DLL werden aber alle generierten MMFs entfernt.
Eine Besonderheit stellt das MMF mit der ID „00“ dar. Dieses MMF wird nicht durch ein Telegramm der Anlage generiert, sondern intern von TCPCOM selbst angelegt. In diesem MMF wird die Information abgelegt, ob TCPCOM online zur Anlage ist. Die anderen Komponenten können diesen Zustand abfragen und den Status der Verbindung zwischen TCPCOM und der Anlage in ihrer Statuszeile anzeigen.“[7]

[7] Systemdokumentation : „BuB – Beobachten und Bedienen“

Name	Typ	Beschreibung
usId	unsigned short	ID des MMF, in der dieses Datenelement abgelegt werden soll.
usType	unsigned short	0 tabellarisch organisiertes MMF 1 als Ringpuffer organisiertes MMF
usValid	unsigned short	immer 1, momentan nicht verwendet
usIndex	unsigned short	null – basierender Index des Elements bei tabellarischer Anordnung
usSize	unsigned short	Größe des Datenelements in Bytes (ohne Header)
usMaxItems	unsigned short	Maximale Anzahl von Elementen (Größe der Tabelle / des Ringpuffers)
es folgen die Nutzdaten selbst		

Tabelle 4 : MMF – Struktur des Headers[8]

Beim Start der DEM-DLL wird ein Informationsfenster geöffnet, aus dem man ablesen kann, welche MMFs mit welchen Einstellungen angelegt sind und welche Fenster an den einzelnen MMFs registriert sind. In Abbildung 32 sieht man dieses Fenster. Exemplarisch wurde hier das MMF „S0“ (Stationen) gewählt. In dem *Header Data* sieht man nun unter anderem die *MaxItems* und die *ItemSize*. In *Registered Windows* tragen sich die Fenster ein, die auf die Daten des gewählten MMFs zugreifen, bzw. die bei Änderungen benachrichtigt werden müssen.

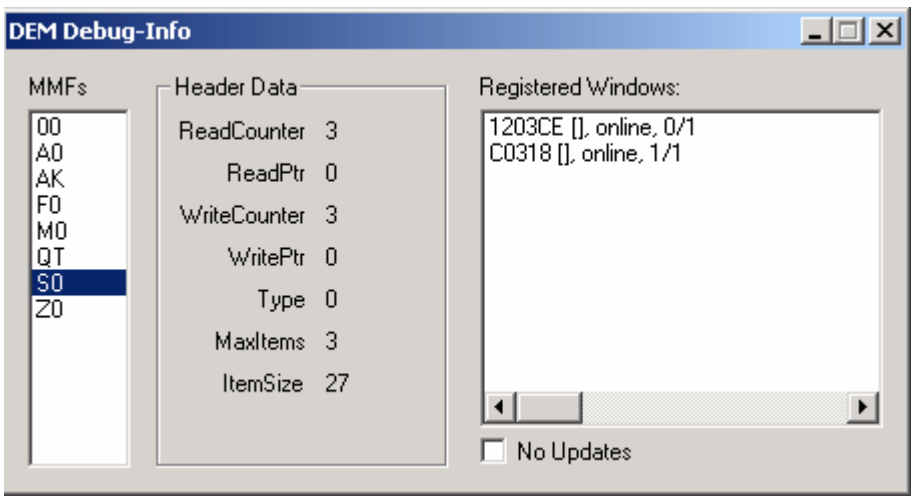


Abbildung 32 : DEM Debug-Info – Fenster

Die DEMClient-DLL hat ausschließlich lesenden Zugriff auf die MMF. Sie stellt die Zugriffsfunktionen auf die MMF bereit. Außerdem registriert sie die Fenster an den MMFs.[9]

[8] Systemdokumentation : „BuB – Beobachten und Bedienen“

3. Methodisches Vorgehen

Um ein methodisches Vorgehen zu gewährleisten, wurde eine Projektmatrix erstellt. Diese gliedert sich in Methode (Methode, die zum Einsatz kommt, um das gewünschte Ziel zu erreichen), Bereich (der grobe Bereich, in dem die Methode zum Einsatz kommt), Ansatz (die Einzelbereiche die untersucht werden sollen) und Instrument (Instrumente, um an die gewünschten Informationen zu gelangen). Die in der Tabelle 5 vorkommenden Punkte wurden anschließend mit Hilfe von MS Projekt™ in einen Zeitplan (siehe Anhang) gebracht, welcher während der Projektbearbeitung kontrolliert wurde.

Methode	Bereich	Ansatz	Instrument
IST – Analyse	Schnittstelle RMOS ß à BuB	Struktur	Recherche VC++ Befragung J. Lachmund (Indumat R.) Sichtung von Dokumentationen
		Inhalt / Interpretation	
		Interaktion	
Vergleich	Funktionalität OS300 ß à OS800	Verarbeitung der Daten	Befragung D. Sagewitz
Festlegungen	Änderungen	Funktionalität OS300	Gespräch D. Sagewitz
		Austauschschnittstelle der relevanten Daten OS300 ß à Schnittstelle	Gespräch D. Sagewitz
Programmierung	OS300 (Schnittstelle)	Send / Receive	AWL evtl. SCL
Benchmarking / Vergleich	Simulation: OS800 ß à BuB vs. OS300 ß à BuB	Funktion	Telegrammverkehr Verhalten des Systems
	Realtest: OS800 ß à BuB vs. OS300 ß à BuB	Funktion	Telegrammverkehr Verhalten des Systems

Tabelle 5 : Projektmatrix

[9] Systemdokumentation : „BuB – Beobachten und Bedienen“

Während der Bearbeitungszeit der Diplomarbeit musste als erstes der bestehende Telegrammverkehr analysiert werden. Hierzu stand die Software des BuB - Systems und das TCPCOM – Log Fenster (Abbildung 31) zur Verfügung. Aus der Software konnte die Struktur und der Inhalt der einzelnen Variablen abgelesen werden. Mit Hilfe des TCPCOM – Log Fensters konnte der Telegrammverkehr eines bestehenden Systems aufgezeichnet und anschließend interpretiert werden. Dadurch konnten Informationen über die Interaktion des Systems gesammelt werden. Eine Übersicht über die Telegramme befindet sich im Anhang (CD).

Danach folgte eine Abstimmung mit Herrn Sagewitz, welche Daten der Telegramme das OS300 bereitstellen kann und welche nicht zur Verfügung stehen. Dadurch wurden einige Festlegungen getroffen. Zum Beispiel kann das OS300 nur einen Fahrzeugtyp bedienen. Deswegen wurde beschlossen, dass der Fahrzeugtyp immer 1 ist.

Nach diesen Festlegungen erfolgte das Erstellen eines Konzeptes für die Schnittstelle. Diese wurde dann im folgenden Schritt programmiert und getestet.

Bisher hat nur ein Test mit einem simulierten System stattgefunden. Ein Realtest folgt sobald die Testanlage voll zur Verfügung steht.

Methode	Bereich	Zeit	Erfüllungsgrad
IST – Analyse	Schnittstelle RMOS ß à BuB	- 06. Mai 2005	100%
Vergleich	Funktionalität OS300 ß à OS800	- 13. Mai 2005	100%
Festlegungen	Änderungen	- 13. Mai 2005	100%
Programmierung	OS300 (Schnittstelle)	- 27. Mai2005 Optimierung	90% (Optimierung)
Benchmarking / Vergleich	Simulation:	- 01. August 2005	100%
	Realtest:	optional	0%

Tabelle 6 : Zeitplan / Erfüllungsgrad

4. Visualisierungsschnittstelle (OS300 – BuB)

Die Schnittstelle, zwischen BuB und OS300, ist für den Austausch von Telegrammen zuständig. Wenn auf Seiten der BuB ein Befehl ausgelöst wird (z.B. Station sperren (Abbildung 33)) soll dieser an das OS300 geschickt werden. Das OS300 soll dann die Station sperren und eine Bestätigung an alle BuB schicken.

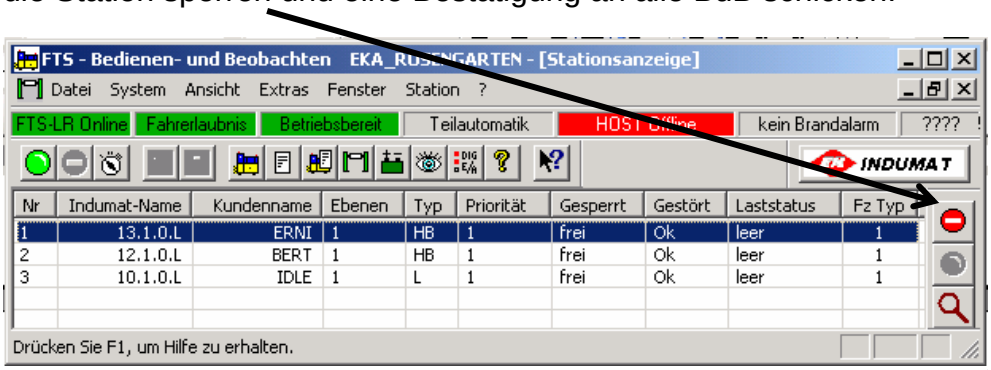


Abbildung 33 : BuB – Station freigegeben

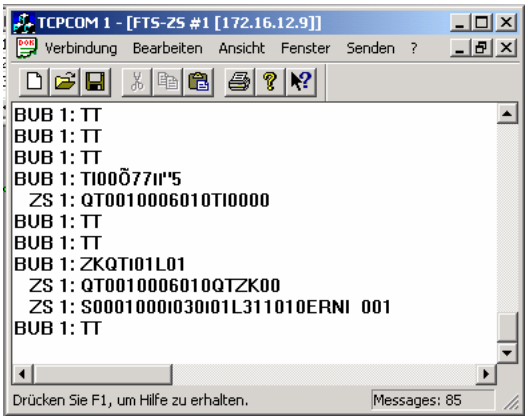


Abbildung 34 : TCPCOM – Station sperren

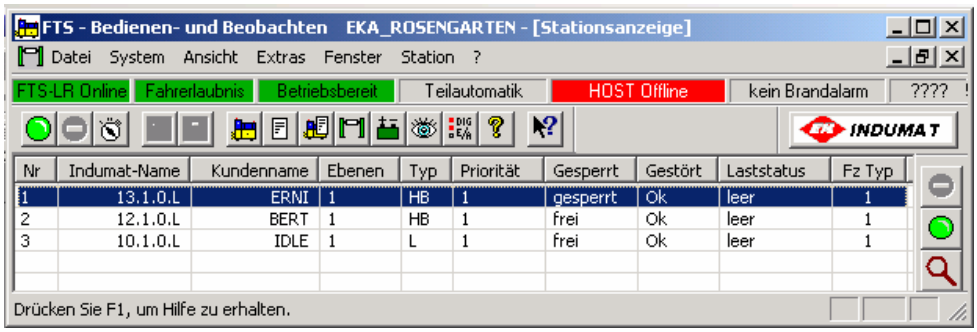


Abbildung 35 : BuB – Station gesperrt

In Abbildung 33 bis Abbildung 35 kann man exemplarisch das Sperren einer Station, mit dem dadurch ausgelösten Telegrammverkehr (Abbildung 34), erkennen. Dieses Beispiel wird im Absatz 4.4 nochmals näher betrachtet.

Wie bereits aus der Aufgabenstellung ersichtlicht soll der Telegrammverkehr, welcher zwischen OS800 und der BuB stattfindet, analysiert und nachgebildet werden, so dass die OS300 auch mit der BuB kommunizieren kann. Das System besteht derzeit aus folgenden Komponenten:

OS300

- CPU 416-2 DP
- CP 443 – 1
- PS 407 4A
- Rack

BuB - System

- PC inkl. Netzwerkkarte
- Monitor

Bedingt durch die Aufgabenstellung soll die Schnittstelle so programmiert werden, dass sie modular, das heißt nicht mit dem OS verzahnt, aufgebaut ist.
Hierfür wurde in der SPS der Funktions- und Datenbausteinsbereich 500-599 gewählt.

Der FC 500 (FC_ COM_BuB) wird aus dem OB 1 (Cycle Execution) aufgerufen. Aus ihm werden dann die Funktionen der Schnittstelle gestartet. Wenn keine BuB zum Einsatz kommen soll, muss dieser Aufruf des FC 500 aus dem OB 1 entfernt werden, damit es nicht zu einem Fehler in der SPS kommt.
Des Weiteren muss noch im DB 1 (Global_DB) im Datenwort (DBW) 4 (HMI-Typ) eine „0“ für OP/TP oder eine „1“ für WinCC stehen. Eine „2“ würde HMI_Typ BuB bedeuten.
Dies sind die einzigen Stellen an denen das OS und die Schnittstelle verzahnt sind.

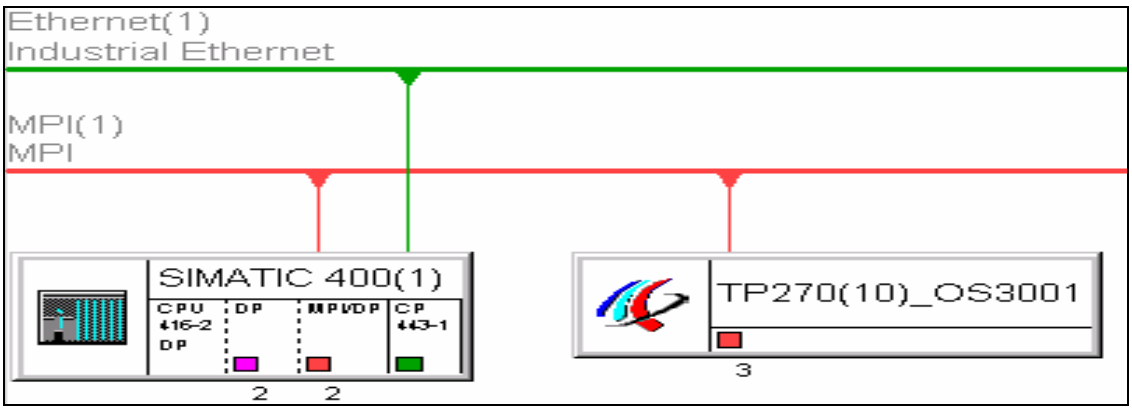


Abbildung 36 : Konfiguration S7

4.1. Notwendige Daten

Damit die BuB arbeiten kann, muss sie mit grundsätzlichen Anlagendaten versorgt werden. Hierzu sind Informationen über den Fahrkurs nötig. Diese werden mit Hilfe des Programms *„Kontrast“* in einer Datenbank abgelegt und dann in DB-Form für die SPS exportiert. In diesen Tabellen befinden sich die spezifischen Daten für jede Strecke, für jede Station und über die Anlagenkonstanten. Da die Streckendaten bereits für das OS erstellt werden mussten, müssen diese nicht gesondert für die BuB erstellt werden.
Außerdem müssen noch die Anlagenzeichnungen für die BuB erstellt werden. Diese umfassen die Hintergrundzeichnung (enthält die Hallenumrisse, Ausgänge, etc.), die Fahrkurszeichnung (enthält den Fahrkurs) und die Simulationszeichnung (enthält die Position jedes FTF an jedem Meldepunkt), die nötig ist, um das FTF an den richtigen Stellen anzeigen zu können.
Dies sind die Daten, die im Vorwege bereitgestellt werden müssen. Die aktuellen Daten werden dann in Form von Telegrammen in die BuB geladen.

In Abbildung 37 erkennt man die Tools, mit denen die notwendigen Daten erstellt werden und welches System (BUB, OS) diese benötigt.

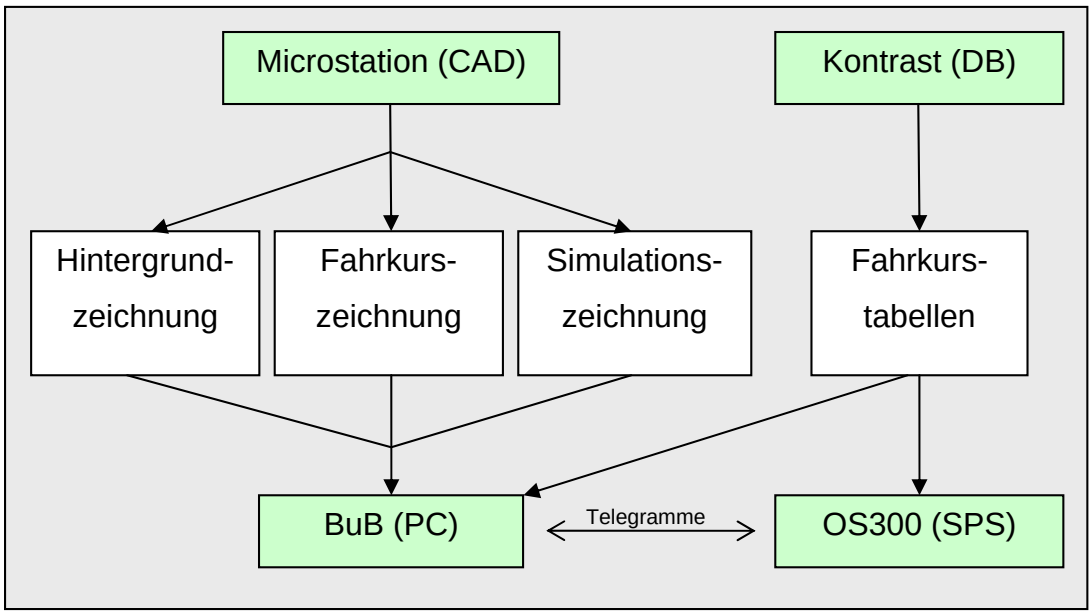


Abbildung 37 : Toolkette

4.2. Struktur der Schnittstelle

Um die Funktionsweise der Software näher zu erläutern, wird als Beispiel das Stationstelegramm näher betrachtet.

Man nehme an, dass von der BuB 1 ein Befehl gesendet werden soll, die „Station 1“ zu sperren. Dieser Befehl wird über die TCP/IP – Schnittstelle an das OS300 geschickt. Dieses empfängt das Telegramm und schreibt es in den *recieve buffer* für die BuB 1. Die BuB 1 verlangt eine Quittierung des Telegramms. Deswegen wird ein QT – Telegramm erzeugt, in den *send buffer* der BuB 1 geschrieben und schließlich an die BuB 1 gesendet. Außerdem wird das Befehlstelegramm noch in den *intern recieve buffer* geschrieben, welcher als Schnittstelle zur OS300 dient. Da die Quittierung nur an die BuB gesendet werden soll, welche den Befehl ausgelöst hat, wurde die Struktur so gewählt, dass es für jede BuB ein *recieve* und *send buffer* Paar gibt.

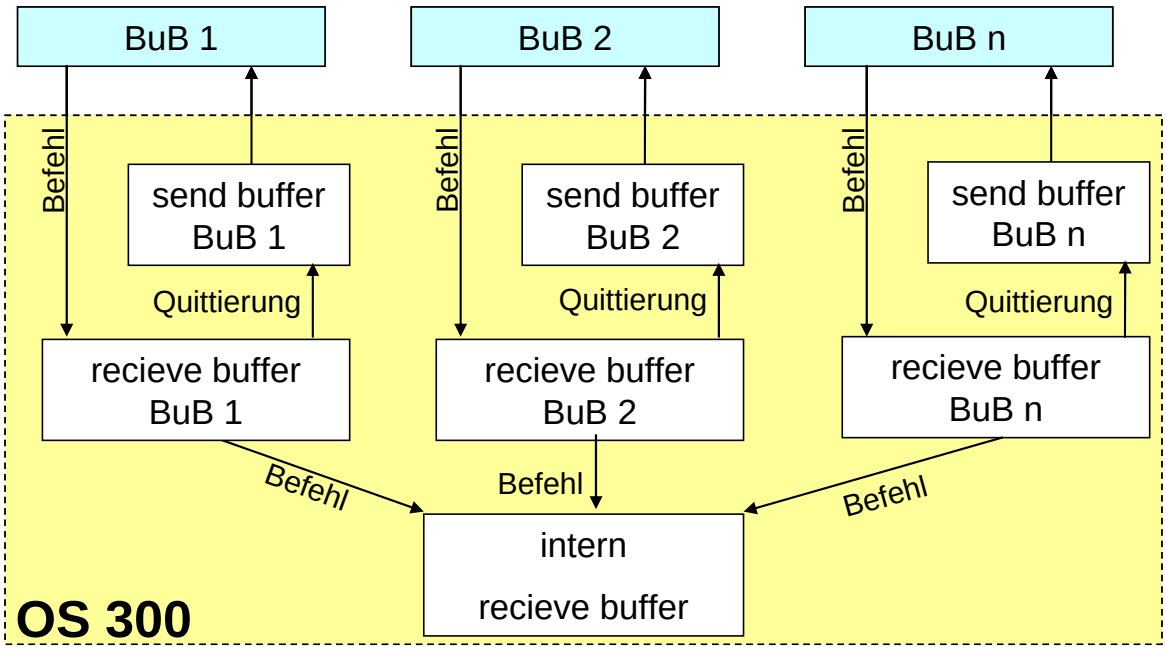


Abbildung 38 : Struktur der Schnittstelle (1)

Das OS300 überwacht den *intern recieve buffer* und verarbeitet die Befehle, die in ihm enthalten sind. Durch die Abarbeitung des Befehles kann es zu einer Änderung des Anlagenzustandes (im Beispiel: Station gesperrt) kommen. Das OS300 vergleicht immer den alten und den aktuellen Zustand der Anlage, so dass die Änderung erkannt wird.

Da diese Änderung den einzelnen BuB mitgeteilt werden muss, schreibt das OS300 eine Anforderung, ein Stationstelegramm zu senden, in den *intern send buffer*. Dieser Buffer wird von der Schnittstelle überwacht, die Anforderung erkannt, das jeweilige Telegramm zusammengestellt und in die *send buffer* der einzelnen BuB (alle die Online sind) geschrieben. Anschließend wird das Telegramm via TCP / IP an jede BuB gesendet.

Da sich der Anlagenstatus des OS300 auch ohne einwirken der BuB ändern kann (z.B. durch Brandalarm), werden auch auf diesem Wege Telegramme gesendet.

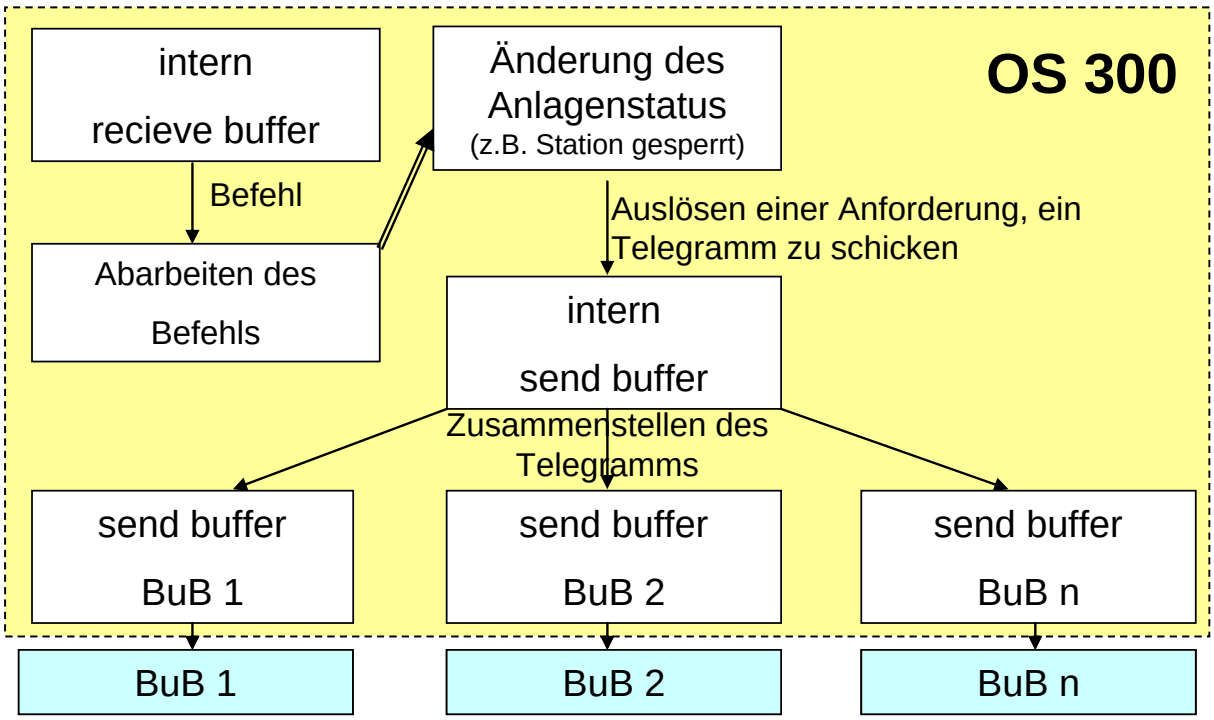
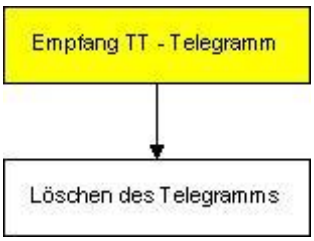


Abbildung 39 : Struktur der Schnittstelle (2)

4.2.1. Reaktionen auf Telegramme

Auf die einzelnen Telegramme reagiert das OS auf unterschiedliche Weise.

4.2.1.1. TT – Telegramm (Alive)



Die BuB sendet alle 10 Sekunden ein TT – Telegramm an das OS. Dieses Telegramm dient als Alive - Signal. Es findet auf Seiten des OS keine Reaktion auf dieses Telegramm statt. Das Telegramm wird nur aus dem *recieve buffer* gelöscht.

Abbildung 40 : Abarbeitung – TT – Telegramm

4.2.1.2. TI – Telegramm (Zeit setzen)

Alle 60 Sekunden wird statt eines TT – Telegramms ein TI – Telegramm gesendet (nur von der BuB mit der PC Nr. 1). Die BuB sendet im TI -Telegramm ihre aktuelle Systemzeit an das OS, welche dann ihre interne Uhr auf diese Zeit einstellt.

Das TI – Telegramm erfordert im Gegensatz zum TT – Telegramm eine Reaktion des OS.

Nach Empfang des TI – Telegramms wird dieses abgearbeitet. Dadurch wird der FC 528 (Empfang TI – Telegramm) aufgerufen. Dieser schreibt ein Quittungs – Telegramm (FC 553) in den *send buffer* und setzt die CPU – Zeit (Datum / Uhrzeit im Telegramm). Anschließend wird das TI – Telegramm aus dem *recieve buffer* gelöscht. Nachdem das QT – Telegramm aus dem FC 501 gesendet wurde, wird dieses aus dem *send buffer* gelöscht.

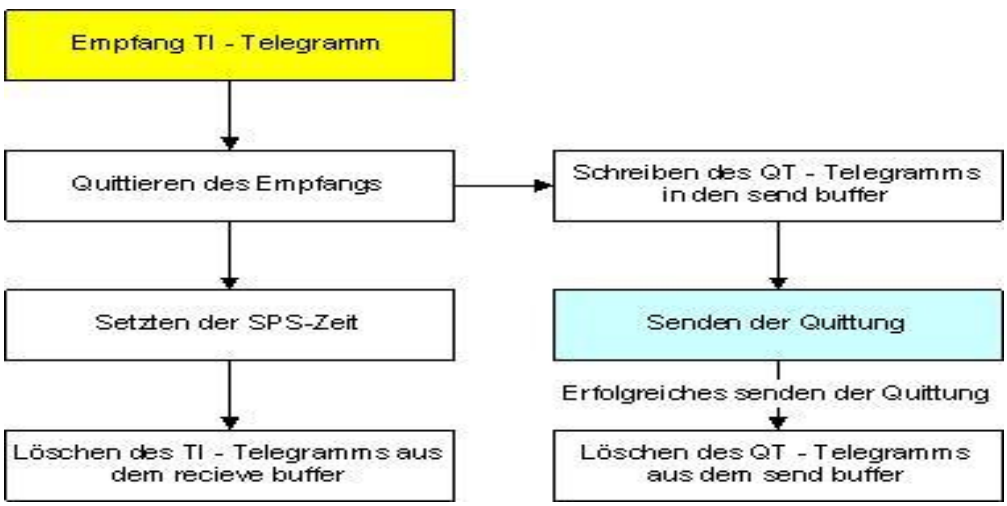


Abbildung 41 : Abarbeitung – TI – Telegramm

4.2.1.3. Sonstige Telegramme

Bei den restlichen Telegrammen erfolgt die Abarbeitung wie bereits unter Absatz 4.2 beschrieben. In Abbildung 42 kann man nochmals den generellen Ablauf erkennen.

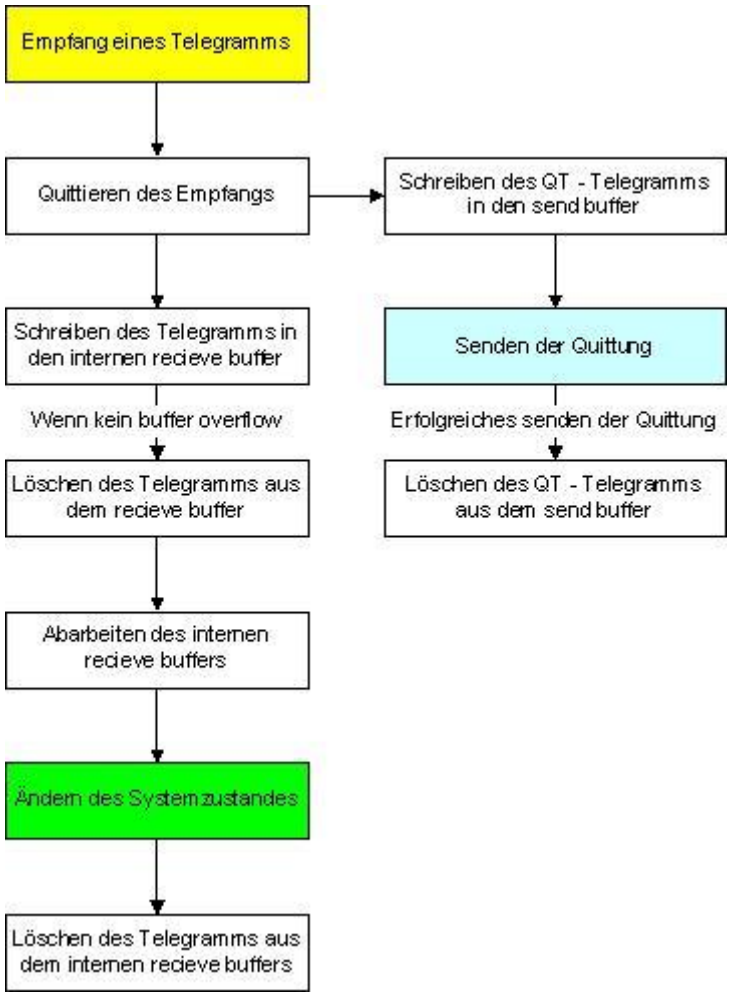


Abbildung 42 : Abarbeitung – Sonstige Telegramme

Folgende Telegramme können von dem BuB ausgelöst werden:

SK – Telegramm:	Beinhaltet Befehle zum Ändern der **Betriebsart**
AT – Telegramm:	Erstellt, ändert oder löscht **Transportaufträge** Transportauftrag **à** Auftrag mit Hole- und Bringeziel
MA – Telegramm:	Erstellt, ändert oder löscht **fahrzeuggebundene Aufträge** fahrzeuggebundene Aufträge **à** Alle die nicht Transportaufträge sind
FK – Telegramm:	Enthält Befehle, die das **Fahrzeug** betreffen. (Fahrverbot)
ZK – Telegramm:	Enthält Befehle, die **Stationen** (Ziele) betreffen (sperren / freigeben)
BK – Telegramm:	Enthält Befehle, die **Batterieladestationen** betreffen (sperren / freigeben)
EK – Telegramm:	Enthält Änderungen für die **Digitalen Ein- und Ausgänge**
TT – Telegramm:	Alive – Signal
TI – Telegramm:	Dient zum Setzten der SPS-Zeit

Tabelle 7 : Übersicht Telegramme BuB à OS

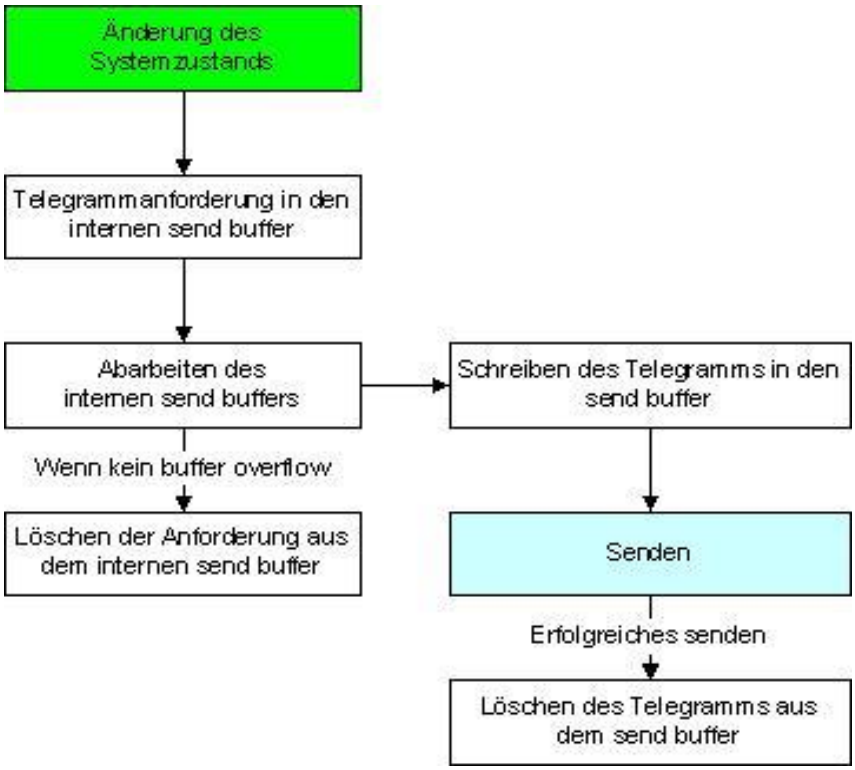


Abbildung 43 : Abarbeitung – Sonstige Telegramme (2)

Folgende Telegramme können von dem OS300 ausgelöst werden:

F0 – Telegramm:	Der **Fahrzeug**datensatz wird an die BuB gesendet (Laststatus, Aktueller Meldepunkt, Zustand der Batterie, etc.)
A0 – Telegramm:	Im A0 – Telegramm steht der **Transportauftrag**s-Datensatz. (Hole-, Bringeziel, Startzeit, etc.)
M0 – Telegramm:	Enthält die Information über eine **fahrzeuggebundenen Auftrag** (Ziel, Aktion am Ziel, Fahrzeugnummer)
S0 – Telegramm:	Enthält den Status einer **Station** (gestört / OK, frei / belegt)
B0 – Telegramm:	Enthält den Status einer **Batterieladestation** (gestört / OK, frei / belegt)
Z0 – Telegramm:	Enthält den aktuellen **Anlagenzustand** (Fahrverbot / -freigabe, Brandalarm,etc)
DT – Telegramm:	Enthält Informationen für das **Störmeldesystem** (FTF-Nr, AT-Nr., etc)
EA – Telegramm:	Enthält den aktuellen Zustand der **Digitalen Ein- und Ausgänge**
QT – Telegramm:	**Quittung**stelegramm
AK – Telegramm:	Enthält die **Anlagenkonstanten** (Anzahl der Fahrzeuge, Stationen, Meldepunkte)

Tabelle 8 : Übersicht Telegramme OS à BuB

Das OS sendet aufgrund eines erhaltenen Telegramms (Änderung des Systemstatus), in den meisten Fälle ein Telegramm an die BuB.

Empfangenes Telegramm	Antworttelegramm(e)	Grund
SK – Telegramm	QT – Telegramm Z0 – Telegramm	Quittierung des Empfangs Änderung des Anlagenzustands
AT – Telegramm	QT – Telegramm A0 – Telegramm evtl. F0 - Telegramm	Quittierung des Empfangs Auftragsänderung neuer Auftrag für Fahrzeug x
MA – Telegramm	QT – Telegramm M0 – Telegramm	Quittierung des Empfangs neuer fahrzeuggebundener Auftrag
FK – Telegramm	QT – Telegramm F0 – Telegramm	Quittierung des Empfangs Änderung des Fahrzeugstatus
ZK - Telegramm	QT – Telegramm S0 – Telegramm	Quittierung des Empfangs Änderung des Stationsstatus
BK – Telegramm	QT – Telegramm B0 – Telegramm	Quittierung des Empfangs Änderung des Batterieladestationsstatus
EK – Telegramm	QT – Telegramm EA – Telegramm	Quittierung des Empfangs Änderung der E/A
TI – Telegramm	QT – Telegramm	Quittierung des Empfangs
TT – Telegramm	keine Reaktion	

Tabelle 9 : Reaktionen auf Telegramme

Es kann vorkommen, dass auf manche Telegramme unterschiedlich geantwortet wird. Es wir zum Beispiel, wenn eine Station gesperrt wird, auf jeden Fall mit einem S0 – Telegramm geantwortet. Wenn aber gerade ein Auftrag bearbeitet wird, der als Ziel diese Station anfahren soll, ändert sich auch der Status dieses Auftrages (Ziel gesperrt). Dadurch wird in diesem Fall auch ein A0 – Telegramm geschickt. In der Tabelle 9 sind nur die grundsätzlichen Reaktionen aufgezeigt.

4.2.1.4. SY – Telegramm (Synchronisation)

Das SY – Telegramm wird nicht von der BuB, sondern vom Modul TCPCOM gesendet. Dieses geschieht, wenn TCPCOM eine Verbindung zum OS hergestellt hat. Durch das SY – Telegramm wird das OS in den Status „Synchronisation“ gesetzt. Hierbei werden alle Anlagendaten an die BuB gesendet. Dabei empfängt nur die BuB, welche gerade im Status „Synchronisation“ ist, die Telegramme. Alle anderen BuB´s erhalten im Verlauf der Synchronisation nur das DT- und die Z0 – Telegramme. Folgend die Reihenfolge der Telegramme bei der Synchronisation:

1. **Z0 – Telegramm**: Status auf Synchronisation
2. **AK – Telegramm**: Anlagenkonstanten
3. Fahrzeugdaten:
    a. **F0 – Telegramm**: Fahrzeugdaten
    b. **M0 – Telegramm**: fahrzeuggebundene Aufträge
4. **A0 – Telegramm**: Transportaufträge
5. **B0 – Telegramm**: Batterieladestationen
6. **S0 – Telegramm**: Stationen
7. **DT – Telegramm**: SMS
8. **EA – Telegramm**: Ein- / Ausgänge
9. **Z0 – Telegramm**: Alter Anlagenstatus (z.B. Betriebsbereit)

Von den Z0-, AK-, DT- und EA – Telegrammen wird nur eins in jedem Schritt geschickt. Da aber eine Anlage mehrere Fahrzeuge (F0- und M0 – Telegramm), mehrere Aufträge (AT – Telegramm) und mehrere Stationen (S0 – Telegramm) haben kann, können von diesen Telegrammen auch mehrere geschickt werden. Deren Anzahl richtet sich nach Größe der Auftragspuffer und nach anlagenspezifischen Daten, wie Fahrzeuganzahl und Anzahl der Stationen. In einem Schritt werden bei den Fahrzeugdaten immer das F0 – Telegramm und das M0 – Telegramm für ein Fahrzeug gesendet. Danach folgen die Daten für das nächste FTF.

Während der Synchronisation findet für die jeweilige BuB kein Abarbeiten des *recieve buffers* und des *intern send buffers* statt. Es werden aber dennoch die Telegramme empfangen und nach der Synchronisation abgearbeitet, so dass keine Informationen verloren gehen.

4.2.2. Funktionsweise der Buffer

In der Software werden Buffer benutzt, um Daten zwischen der BuB und der Schnittstelle, sowie der Schnittstelle und dem OS300, auszutauschen. Diese Buffer haben alle die gleiche Struktur. Im DBW 0 bis DBW 6 stehen die Informationen über den nächsten freien Platz, die aktuelle Anzahl der Telegramme und die Größe des Buffers. Danach folgen die Telegrammdaten. Immer wenn ein Telegramm in den Buffer eingetragen werden soll, muss überwacht werden, ob es noch in den Buffer passt.

$$next\_free\_pointer_{+1} = next\_free\_pointer + length\_of\_telegram$$

[Formel 1 : Berechnung – next_free_pointer]

Wenn next_free_pointer > max_pointer ist, kann das Telegramm nicht mehr in den Buffer eingetragen werden (buffer overflow). Im nächsten Zyklus der SPS muss dies erneut versucht werden. Wenn es gelingt, ist es noch notwendig den „next_free_pointer“ (DBW 2) [Formel 1] zu berechnen. Außerdem muss noch die aktuelle Anzahl der Telegramme („total_tele“ (DBW 4)) um eins erhöht werden. Wenn ein Telegramm aus dem Buffer gelöscht werden soll, wird der FC 512 (FC_DelTeleFromBuffer) aufgerufen. Dieser löscht das erste Telegramm, verschiebt die restlichen Telegramme nach vorne, berechnet den nächsten freien Pointer und verringert „total_tele“ um eins.

Der Aufbau der Buffer ist in Abbildung 44 zu erkennen.

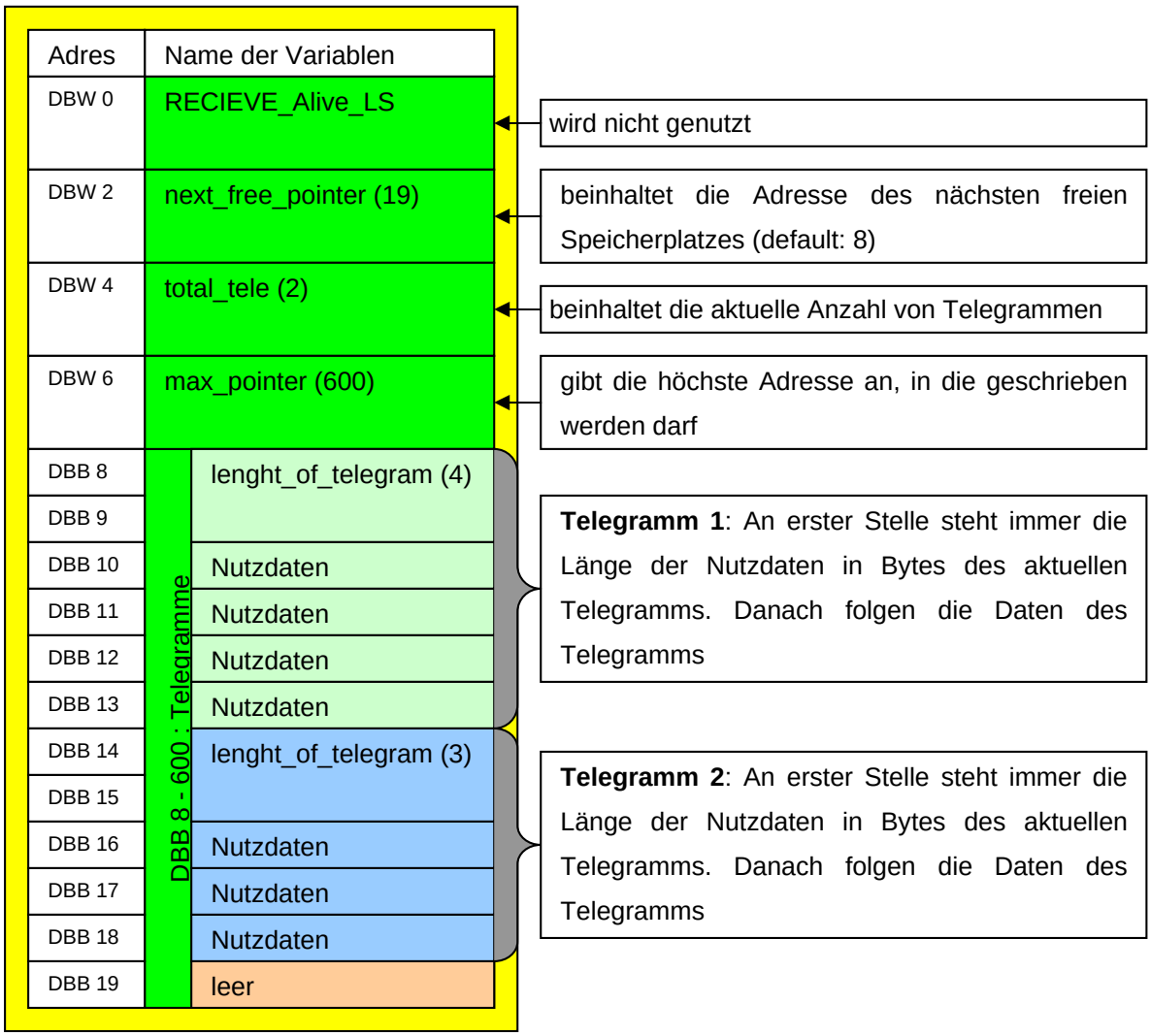


Abbildung 44 : Struktur der Buffer

Insgesamt kommen 4 unterschiedliche Buffer zum Einsatz. Es gibt für jede BuB einen *recieve* und einen *send buffer*. Zurzeit ist das OS auf den Betrieb von 2 BuB gleichzeitig ausgelegt. Das heißt, dass es insgesamt 2 *recieve* und 2 *send buffer* gibt. Hinzu kommt noch ein *intern recieve* und ein *intern send buffer*. Somit sind es insgesamt 6 Buffer.

In die ***send buffer*** werden die Telegramme der BuB nach deren Erhalt eingetragen. Dieser wird dann ausgelesen und die Telegramme werden interpretiert.

Der Inhalt der Telegramme wird dann, falls nötig, in den ***intern recieve buffer*** geschrieben. Da das Dateiformat bei der SPS und dem PC unterschiedlich ist, müssen bei WORD- und DWORD-Variablen noch die Bytes getauscht werden.

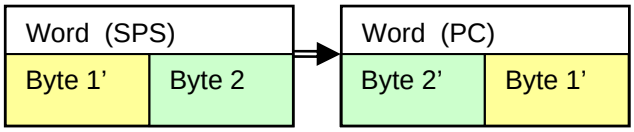


Abbildung 45 : BYTE – Tausch bei WORDs

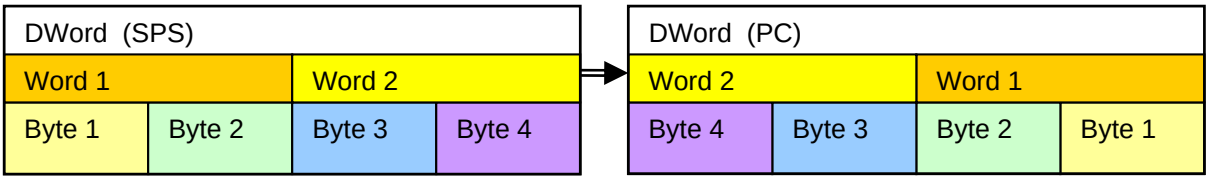


Abbildung 46 : BYTE – Tausch bei DWORDs

Das OS schreibt, nach einer Änderung des Systemstatus, in den ***intern send buffer*** nur die Anforderung, dass ein Telegramm gesendet werden soll.

Aufgrund einer Anforderung werden die nötigen Daten für das jeweilige Telegramm durch das OS zusammengestellt und in den ***send buffer*** geschrieben, aus dem das Telegramm später gesendet wird.

4.3. Die Bausteine

In diesem Kapitel werden die einzelnen Funktionen näher erläutert. Hierzu werden Ablaufdiagramme benutzt. Da es in der Programmiersprache Step 7 – AWL nötig ist Schleifen und Bedingungen mit Sprüngen selber zu erstellen, wurde diese Art von Diagrammen gewählt, da Sprünge mit dieser Technik relativ einfach darzustellen sind. In AWL kann man, um die Funktionen zu strukturieren, einzelne Netzwerke anlegen, in denen sich Funktionsteile befinden, die zusammen gehören. Um die einzelnen Funktionsteile möglichst schnell im Quelltext wieder zu finden sind die Diagramme farblich hinterlegt. Außerdem befindet sich in der oberen rechten Ecke die Netzwerknummer und der Netzwerkname. In den einzelnen Abschnitten (außer FC 500 und FC 560, da Aufruf ohne Parameter) wird ein Beispiel für den Aufruf der jeweiligen Funktion gegeben. Zusätzlich folgt noch eine Übersicht und Erklärung der Ein- und Ausgänge der Funktion.

Über dem eigentlichen Programmcode (Abbildung 47) befindet sich noch eine Versionsverwaltung und eine Beschreibung der möglichen Fehlercodes, die der Baustein generieren kann (RET_VAL).

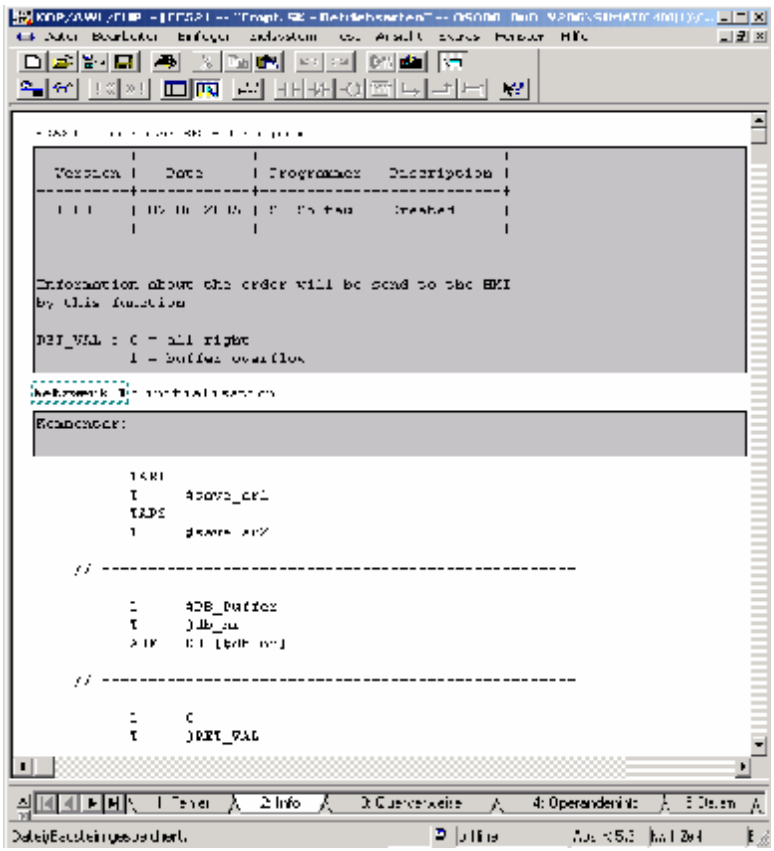


Abbildung 47 : AWL – Programmierfenster (recieve)

4.3.1. FC 500 – FC_COM_BuB

Wie bereits beschrieben erfolgt der Aufruf der einzelnen Funktionen aus dem FC 500.

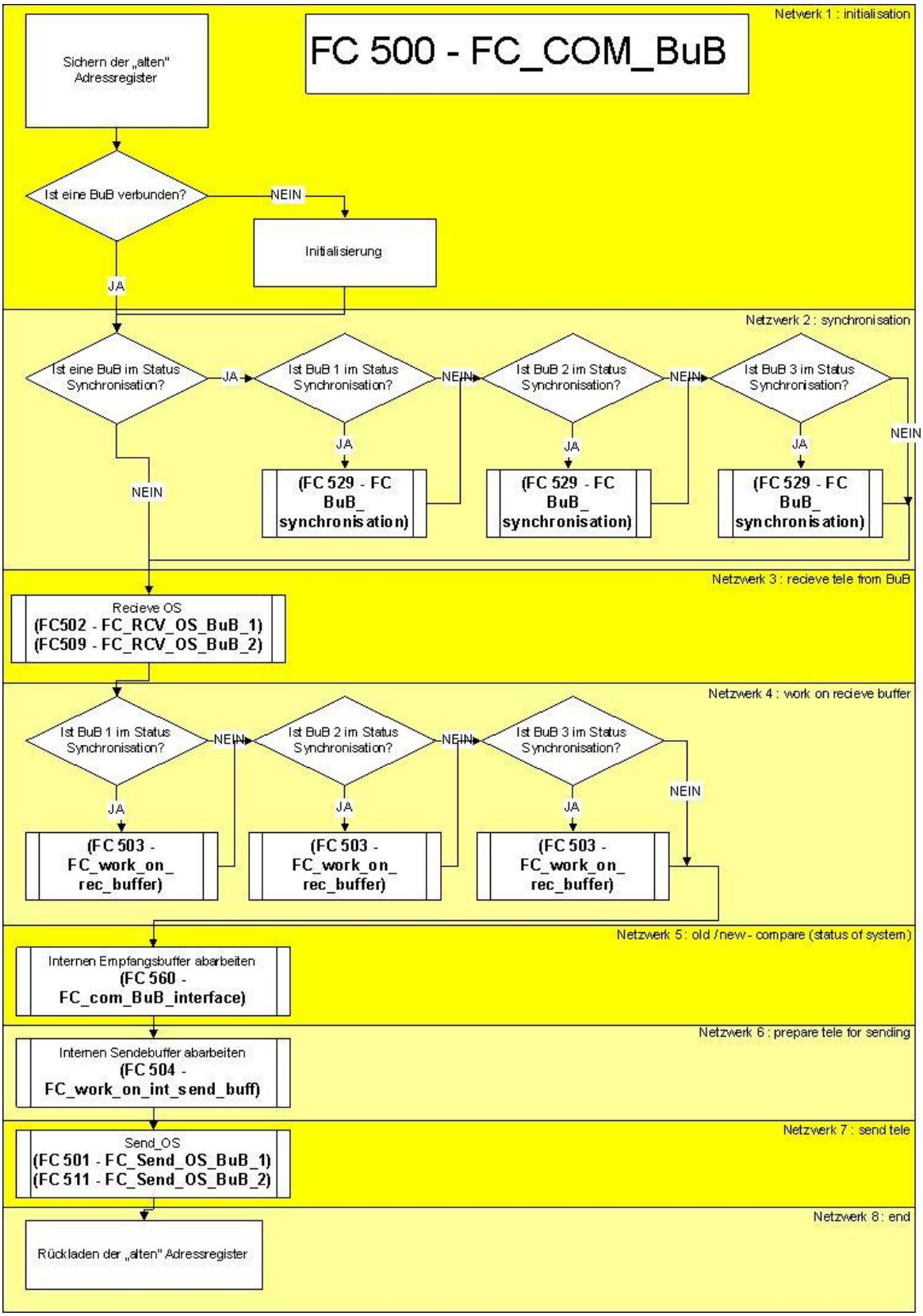


Abbildung 48 : Struktur FC 500 – FC_COM_BuB

4.3.1.1. FC 529 – FC_BuB_synchronisation

Wenn sich eine BuB im Synchronisationsmodus befindet wird der FC 529 aus dem FC 500 aufgerufen. Der FC 529 übernimmt die Synchronisation des OS300 mit den BuB. Alle anderen BuB bekommen außer der Systemstatusänderung und dem DT – Telegramm (Absatz 4.2.1.4) nichts von der Synchronisation mit. Es werden die Synchronisationsdaten an die BuB gesendet, welche zurzeit im Status Synchronisation ist. Zum FC 529 gehört noch der DB 501 (DBwork Synchro). In ihm befinden sich die Counter für die Schrittkette (Step_counter) und für die Fahrzeuge, Stationen und Aufträge.

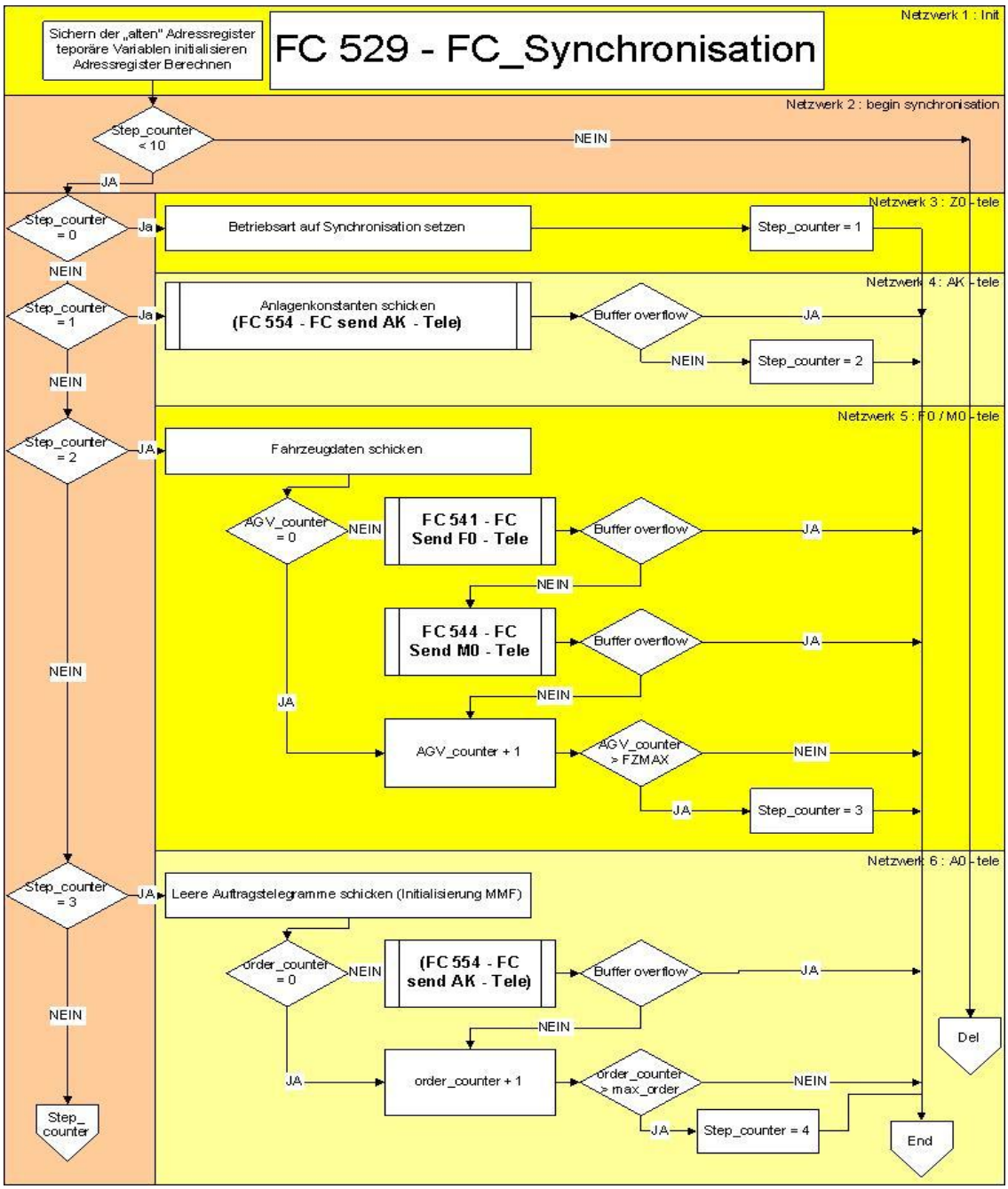


Abbildung 49 : Struktur - FC 529 – BuB_synchronisation (1)

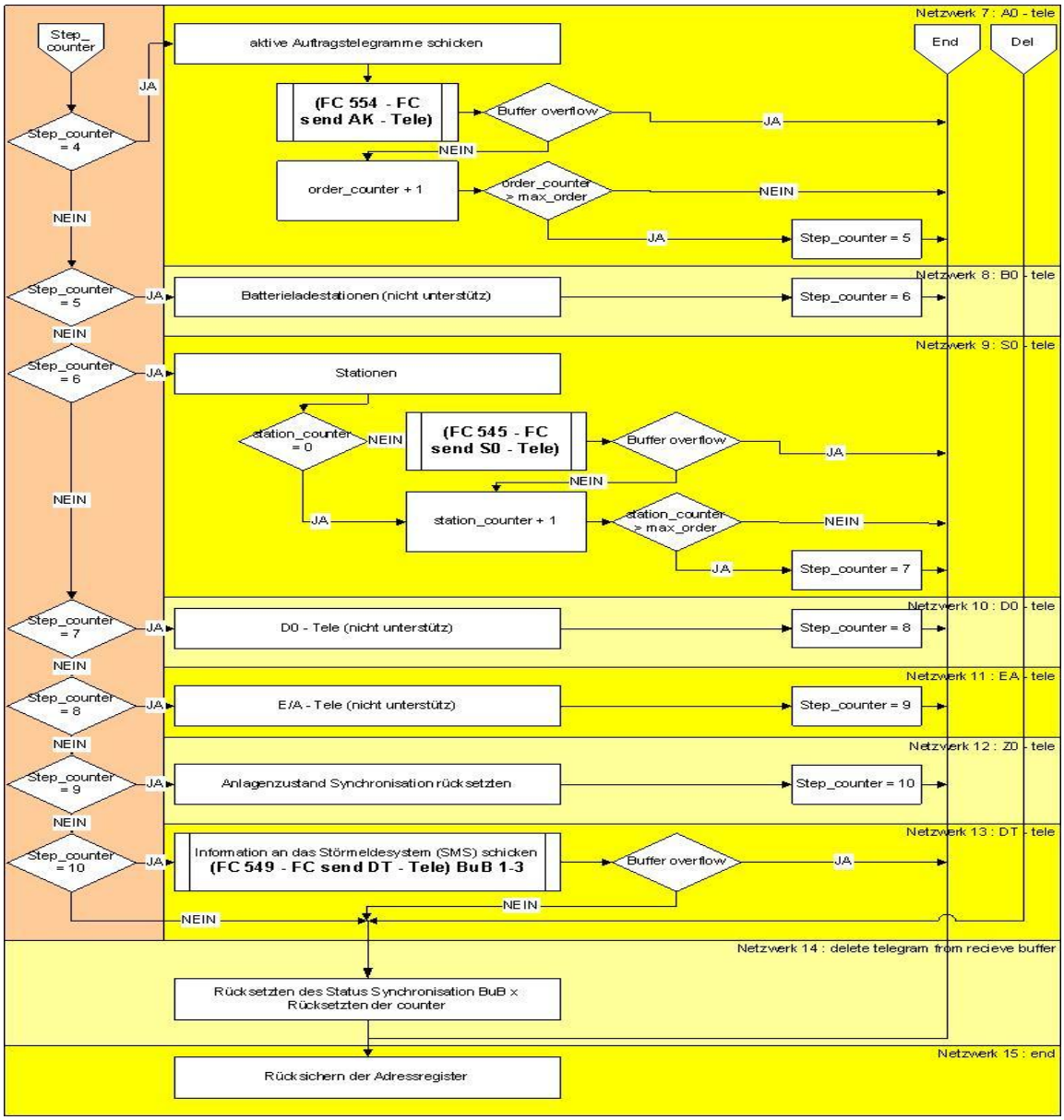


Abbildung 50 : Struktur - FC 529 – BuB_synchronisation (2)

Im Netzwerk 2 wird ausgewertet, in welchem Schritt sich die Synchronisation befindet. In den folgenden Netzwerken werden dann die jeweiligen Telegramme gesendet. Hierbei erfolgt eine Überwachung des *send buffer*s in Hinsicht auf eine buffer overflow. Nähere Informationen zum Status Synchronisation befinden sich im Abschnitt 4.2.1.4.

Der Aufruf aus dem FC 500 sieht wie folgt aus: (Beispiel für BuB 1)

```
CALL  "BuB_synchronisation"          // FC 529
 BuB_id_nr          :=1              // IN    // communication ID - BuB 1
 db_send_buffer     :=515            // OUT // DB number of send buffer
 RET_VAL            :=#return_DW     // IN     // return value - 0 = all right
```

Beim Aufruf müssen folgende Parameter versorgt werden:

Art	Name	Datentyp	Erklärung
IN	BuB_ID_nr	INT	Für jede BuB wird der FC 529 aufgerufen. Dabei hat jede eine „eigene“ BuB ID. Diese entspricht der ID der Verbindung
IN	db_send_buffer	INT	Jede Verbindung hat einen eigenen *send buffer*.
OUT	Ret_Val	DWORD	Rückgabewert der Funktion (immer 0)

Tabelle 10 : FC 529 – Ein- / Ausgänge

4.3.1.2. FC 502 – FC_recieve_OS_BuB_1

Es handelt sich beim FC 502 um einen Standardbaustein von E&K. Dieser ruft den FC 60 AGLrecieve auf, welcher für den Empfang der Telegramme zuständig ist. Der FC 502 entfernt die Rahmendaten und schreibt das Telegramm in den dazugehörigen *recieve buffer*.

```
CALL  "RCV_OS_BuB_1"
 id_nr              :=1
 DB_rec_work        :=514
 DB_rec_buffer      :=516
```

Zurzeit gibt es für jede Verbindung einen separaten work_db in dem die Telegrammdaten zwischengespeichert werden. Dies wird im Zuge der Optimierung der Schnittstelle noch verändert, um die Erweiterung der Schnittstelle zu vereinfachen. Dies war jedoch im Rahmen der Diplomarbeit zeitlich nicht mehr möglich.

Art	Name	Datentyp	Erklärung
IN	id_nr	INT	ID-Nummer der Verbindung
IN	DB_rec_work	INT	Zwischenspeicher DB
IN	DB_rec_buffer	INT	Buffer in dem die empfangenden Telegramme stehen.

Tabelle 11 : FC 502 – Ein- / Ausgänge

4.3.1.3. FC 501 – FC_send_OS_BuB_1

Der FC 501 ist, wie der FC 502, ein Standardbaustein von E&K. Er sendet die Daten die im *send buffer* stehen, mit Hilfe des FC 50 AGLsend, an die BuB. Hierfür fasst er vorher die Nutzdaten in die Rahmendaten ein, kopiert das Telegramm in einen Work_DB (Eingang der Funktion) und sendet dieses von hier aus.

```
CALL  "SEND_OS_BuB_1"
 DB_Send_buffer_OS  :="DB_SENDBUFFER_OS"
 id_nr              :=1
 DB_work_send_os    :="DB_Send_Work_OS"
```

Art	Name	Datentyp	Erklärung
IN	DB_Send_buffer_OS	INT	Buffer, in dem die zu sendenden Daten stehen
IN	id_nr	INT	ID-Nr der Verbindung
IN	DB_work_send_os	INT	work DB zum zwischenspeichern des Telegrammes

Tabelle 12 : FC 501 – Ein- / Ausgänge

4.3.1.4. FC 560 – FC_com_BuB_interface

Der FC 560 bearbeitet den *intern receive buffer*. Dieser Baustein wurde von Herrn Sagewitz geschrieben, da er sich mit der Funktionalität des OS300 am besten auskennt. Die im Buffer enthaltenen Telegramme werden abgearbeitet (der Befehl wird ausgeführt). Außerdem erfolgt im FC ein Alt – Neu Vergleich des Systemzustandes, wodurch bei Änderungen eine Telegrammanforderung ausgelöst wird.

```
CALL  "FC_com_BuB_interface"
```

Es werden keine Parameter benötigt.

4.3.1.5. FC 503 – FC_work_on_recieve_buffer

Der FC 503 wertet den *receive buffer* der einzelnen BuB aus. Hierfür wird er für jede BuB einzeln, mit anderen Parametern, aufgerufen.

Im Netzwerk 2 wird entschieden, was für ein Telegramm an der ersten Stelle des buffers steht. Je nach Telegramm wird dann in eines der folgenden Netzwerke gesprungen (z.B. bei ZK - Telegramm in Netzwerk 9), in denen der dazugehörige recieve FC (4.3.1.6 - Die „recieve“ Bausteine) aufgerufen wird.

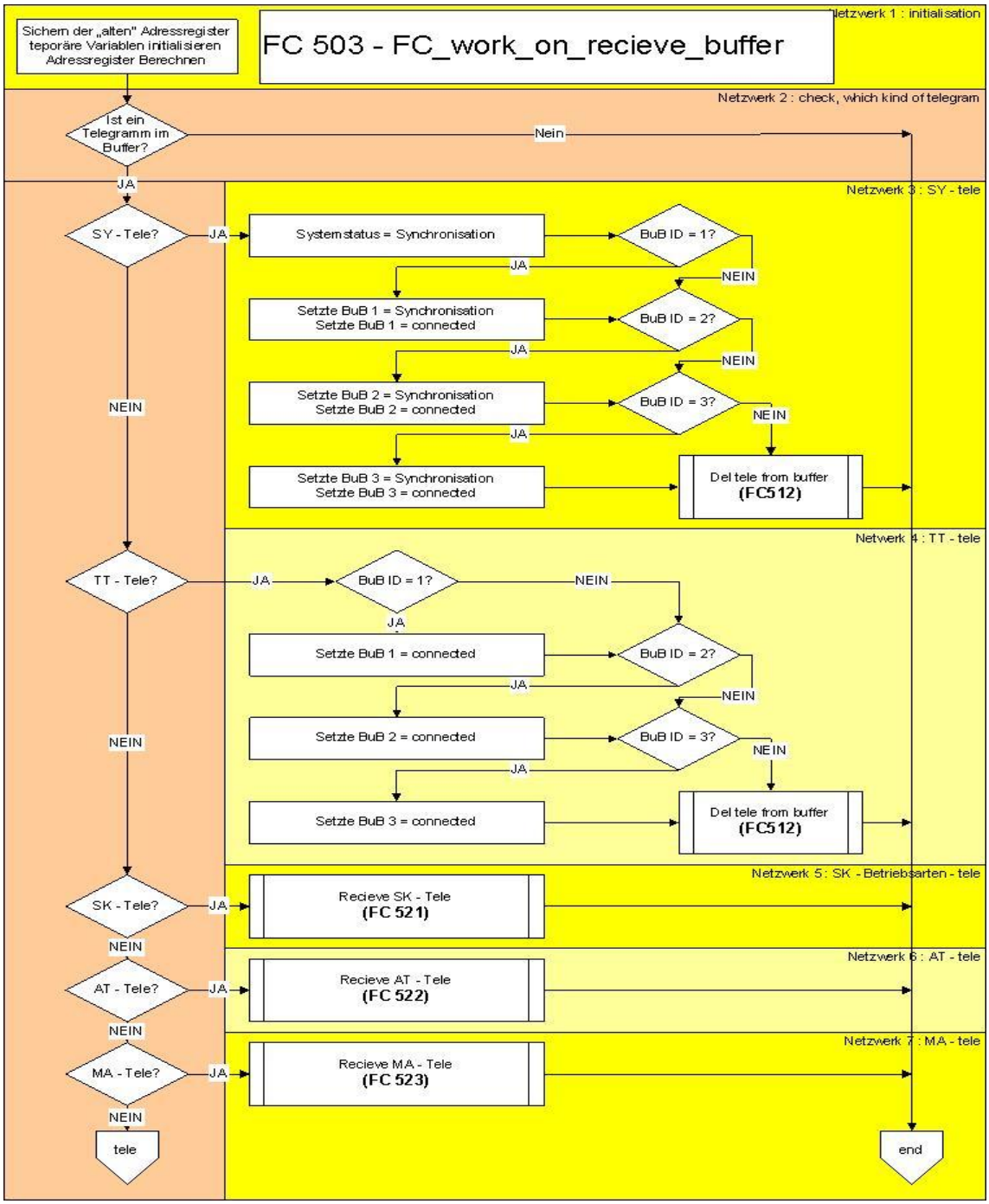


Abbildung 51 : Struktur - FC 503 – Work_on_recieve_buffer (1)

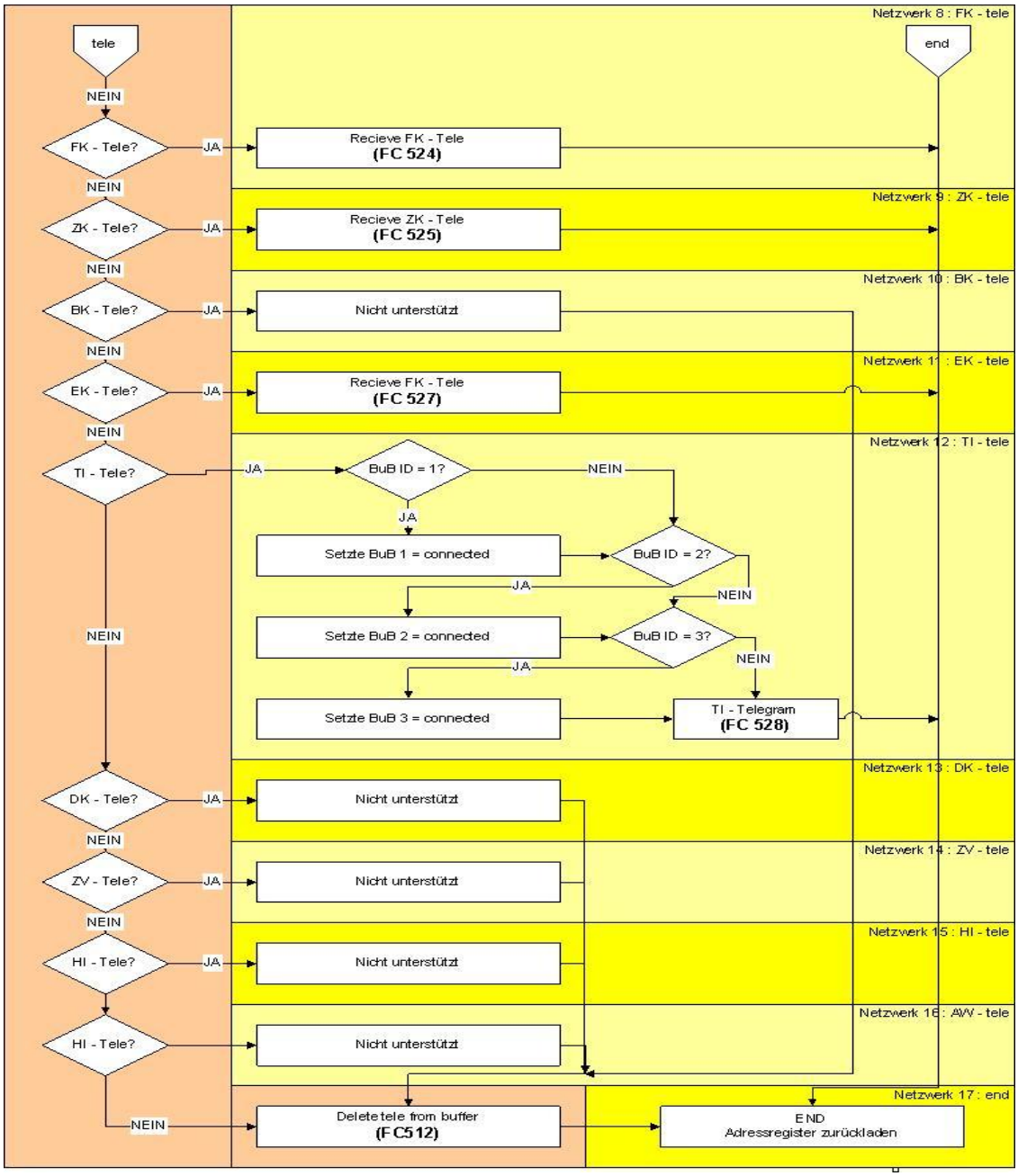


Abbildung 52 : Struktur - FC 503 – Work_on_recieve_buffer (2)

Beim Empfang eines TT, SY oder TI – Telegramms wird außerdem noch der Status der BuB, welche dieses Telegramm empfangen hat, auf *connected* gesetzt. Beim SY – Telegramm wird zusätzlich der Status der jeweiligen BuB auf Synchronisation gesetzt.

```
CALL  "FC_work_on_rec_buffer"
DB_RCV_BUFFER_OS   :="DB_RCV_BUFFER_OS"
DB_SEND_BUFFER_OS :="DB_SENDBUFFER_OS"
BuB_id_nr                  :=1
```

Art	Name	Datentyp	Erklärung
IN	DB_RCV_BUFFER_OS	Block_DB	DB Nummer des recieve buffer
IN	DB_SEND_BUFFER_OS	Block_DB	DB Nummer des send buffer
IN	BuB_id_nr	INT	ID Nummer der Verbindung (BuB)
OUT	Ret_Val	DWORD	Rückgabewert der Funktion (immer 0)

Tabelle 13 : FC 503 – Ein- / Ausgänge

4.3.1.6. Die „recieve“ Bausteine

Müssen Änderungen im Telegramminhalt vorgenommen werden, ist nur eine Änderung der Recieve- und Sendbausteine nötig. Es kann zu deren Veränderungen kommen, wenn der Kunde zusätzliche Anlagendaten wünscht, die angezeigt werden sollen.
Um diese Änderungen so leicht wie möglich zu gestalten, wurde für die recieve Bausteine eine einheitliche Struktur gewählt.

Wenn ein Telegramm im *recieve buffer* der BuB steht, wird die dazugehörige Funktion aufgerufen.
Um das ganze einfacher zu verstehen, wird als Beispiel der Empfang eines ZK – Telegramms (Stationen) genommen. Wenn also ein ZK – Telegramm im *recieve buffer* steht, wird der FC 525 aufgerufen. Dieser kopiert nach der Initialisierung der temporären Variablen, den Inhalt des Telegramms erst in einen *work_DB*, danach wird der Erhalt des Telegramms an die BuB quittiert. Anschließend werden die Daten aus dem *work_DB* in den *internen recieve buffer* kopiert. Nachdem das Telegramm erfolgreich in den *internen send buffer* eingetragen wurde, wird das von der BuB erhaltene Telegramm aus dem *recieve buffer* gelöscht.

Für Änderungen im Inhalt der Telegramme müssen nur noch die „neuen“ Daten im Netzwerk 2 in den *work_DB* kopiert werden. Die Länge des Telegramms steht im Telegramm selbst. Es müssen dadurch in den anderen Kopierfunktionen keine Änderungen vorgenommen werden.

Da der FC 560 – FC_com_BuB_interface die Daten aus dem *intern recieve buffer* auswertet, müssen zusätzlich Änderungen im jeweiligen Netzwerk dieses FC´s durchgeführt werden.

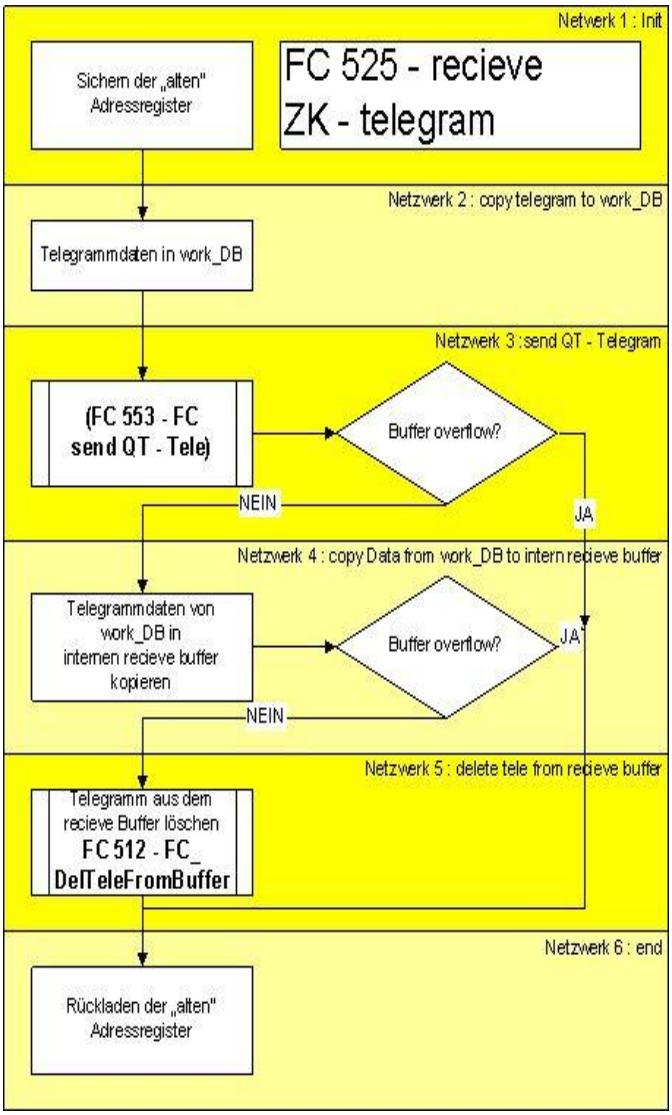


Abbildung 53 : Abarbeitung – ZK – Telegramm

Beispiel eines Aufrufes

CALL "FC recieve ZK - tele"
DB_recieve_buffer:=#db_no_recieve_buffer
DB_send_buffer :=#db_no_send_buffer // für QT – Telegramm

Art	Name	Datentyp	Erklärung
IN	DB_recieve_buffer	INT	aktueller receive buffer, aus dem gelesen wird, je BuB einer (BuB 1 **à** DB 516)
IN	DB_send_buffer	INT	aktueller send buffer, in den die Quittierung geschrieben werden soll, je BuB einer (BuB 1 **à** DB 515)
OUT	Ret_Val	DWORD	Rückgabewert der Funktion (immer 0)

Tabelle 14 : FC 525 – Ein- / Ausgänge

4.3.1.7. FC 504 – FC_work_on_int_send_buff

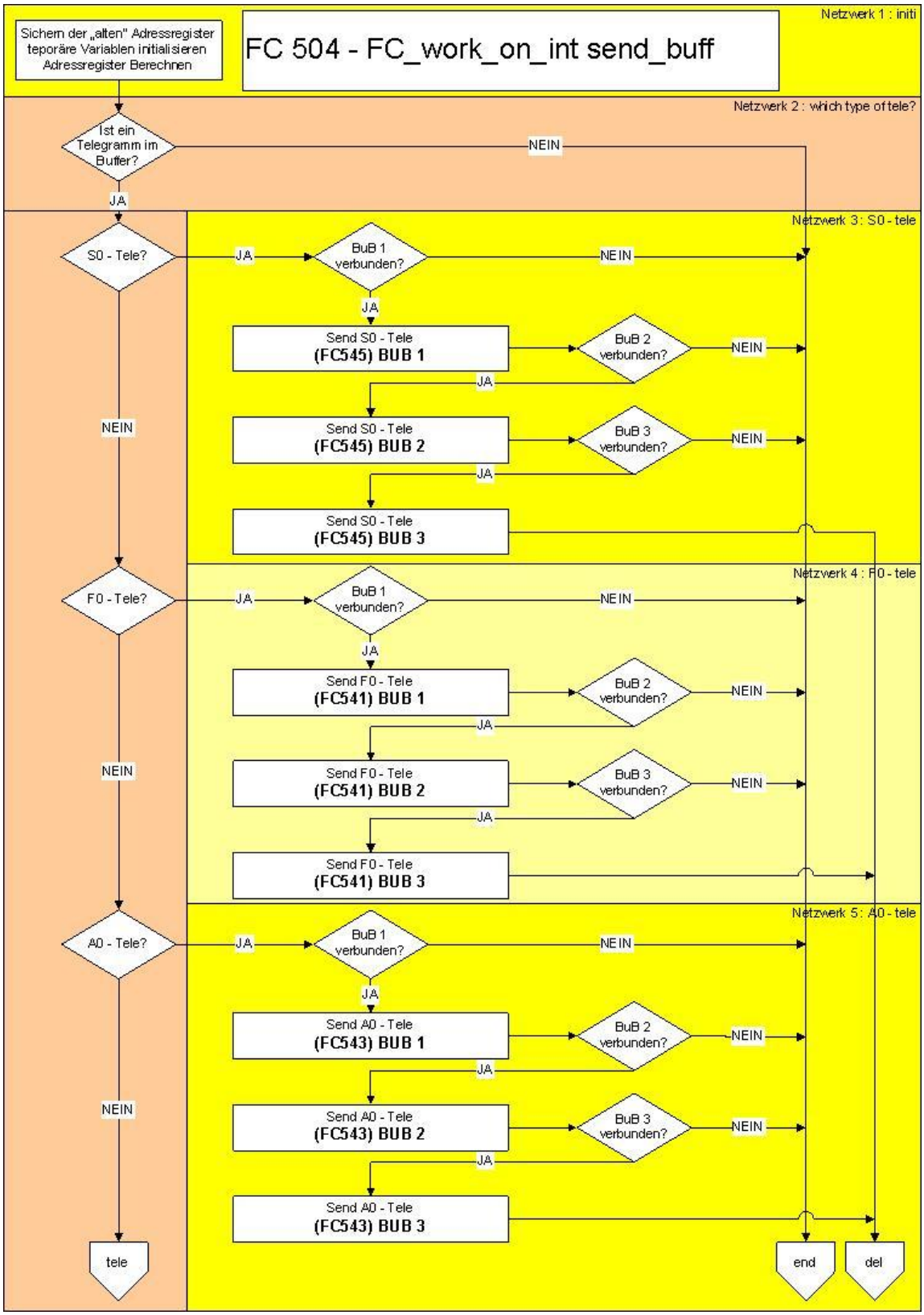


Abbildung 54 : FC 504 – FC_work_on_intern_send_buff (1)

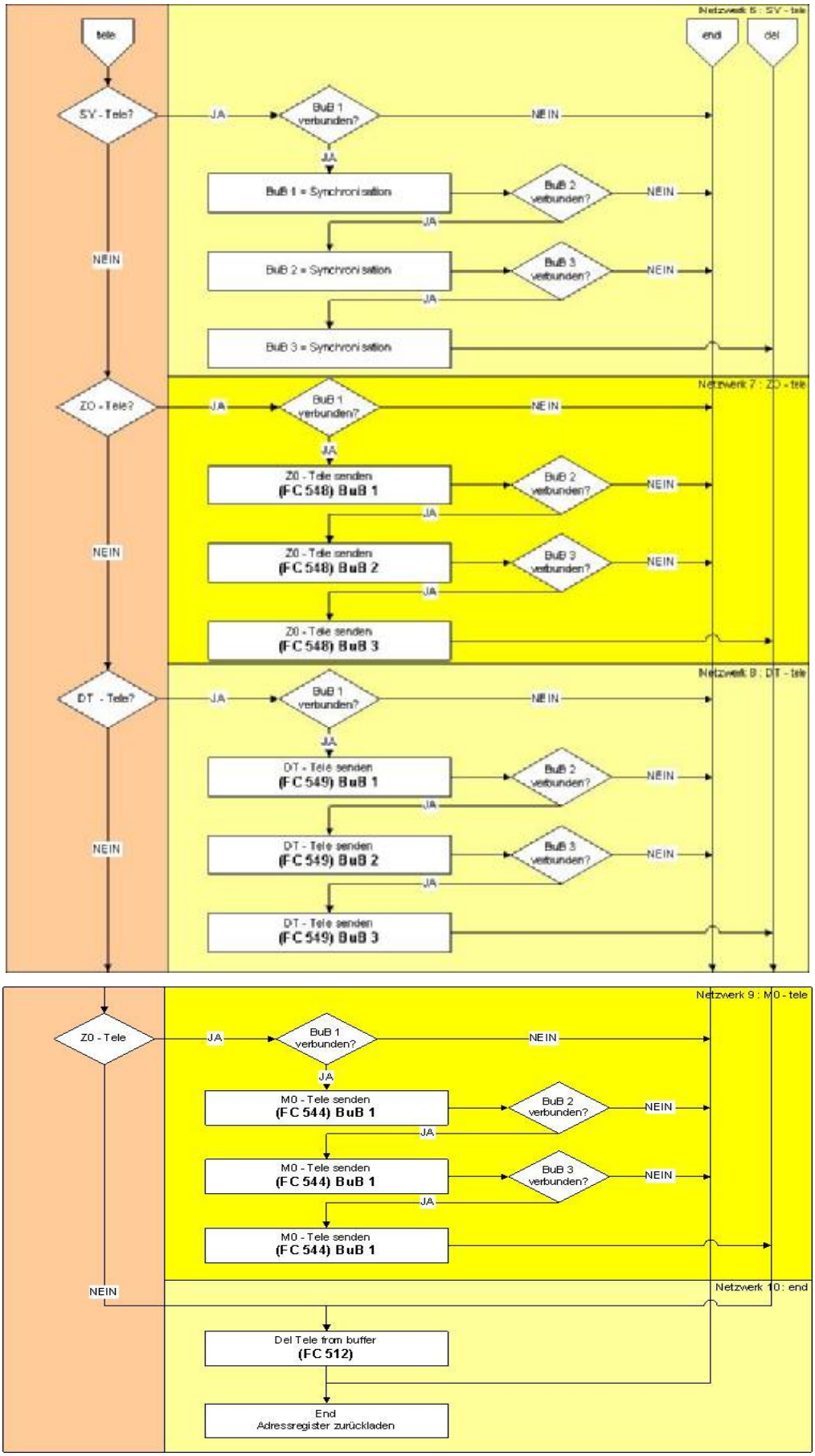


Abbildung 55 : FC 504 – FC_work_on_intern_send_buff (3)

Der FC 504 wertet den *intern send buffer* aus. Das heißt, dass er prüft ob eine Telegrammanforderung im Buffer steht. Anschließend wird im Netzwerk 2 zum jeweiligem Netzwerk des Telegramms (z.B. S0 – Telegramm ins Netzwerk 3) gesprungen. Aus diesem wird der send FC aufgerufen (4.3.1.8 - Die „send“ Bausteine), welcher das jeweilige Telegramm zusammenstellt und in den *send buffer* schreibt.

CALL "FC_work_on_int_send_buff"
DB_RCV_BUFFER_intern :="DB_Send_intern"
DB_SENDBUFFER_intern :="DB_Rec_intern"

Art	Name	Datentyp	Erklärung
IN	DB_RCV_BUFFER_intern	Block_DB	
IN	DB_SENDBUFFER_intern	Block_DB	
OUT	Ret_Val	DWORD	Rückgabewert der Funktion (immer 0)

Tabelle 15 : FC 504 – Ein- / Ausgänge

4.3.1.8. Die „send“ Bausteine

Die „send“ Bausteine werden aufgerufen, wenn eine Anforderung, ein Telegramm zu senden, im *intern send buffer* steht oder wenn ein QT – Telegramm (send QT – Tele) gesendet werden soll. Im Beispiel wird die Telegramm-Anforderung im *intern send buffer* der FC 545 (send S0 – Tele) gestartet. Dieser stellt das Telegramm zusammen und schreibt es in den *send buffer*. Eventuelle Änderungen im Inhalt des Telegramms müssen im Netzwerk 3 vorgenommen werden. Hier müssen die „neuen“ Daten in den zugehörigen DB geschrieben werden. Es muss aber vorher im DB eine Struktur für die neuen Daten angelegt werden. Diese wird im Netzwerk 3 mit Daten versorgt. Außerdem muss im DBW 0 des DB die Länge des Telegramms aktualisiert werden (Länge des DB – 2). Dies sind alle Änderungen, die nötig sind.

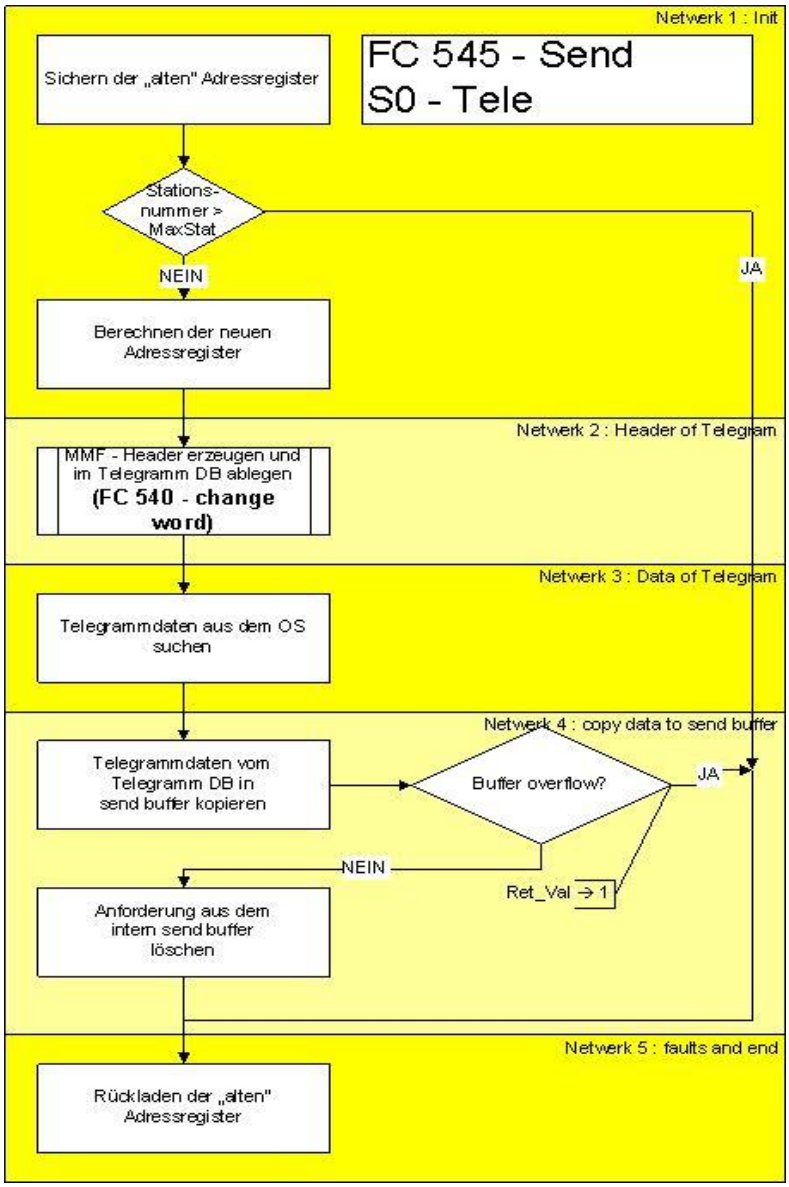


Abbildung 56 : Abarbeitung – S0 – Telegramm

Beispiel eines Aufrufes:

```
CALL  "FC send S0 - Tele"     // Stationsnummer, dessen Daten gesendet
stationnumber :=#station      // werden sollen
db_send_buffer:=515           // send buffer
RET_VAL       :=#return_DW    // Fehlerwort: Wenn ungleich 0 à buffer overflow
```

Art	Name	Datentyp	Erklärung
IN	stationnumber	INT	Stationsnummer, für die die Daten gesendet werden sollen
IN	DB_send_buffer	INT	aktueller send buffer, für die Quittierung
OUT	Ret_Val	DWORD	0 = Alles in Ordnung / 1 = buffer overflow

Tabelle 16 : FC 545 – Ein- / Ausgänge

Wie bereits beschrieben, gibt es für jeden FC einen DB, in dem die Daten des Telegramms zwischengespeichert werden. Dieser DB hat folgende Struktur:

Adr	Name	Typ	Anfangs-wert	Beschreibung
0.0	laenge_tele	WORD	W#16#1F	Telegrammlänge = DB-Länge – 2
2.0	ucID	2 CHAR	'S0'	siehe 2.4.3.1 - Die MMF
4.0	usType	2*BYTE	B#16#00	siehe 2.4.3.1 - Die MMF
6.0	usValid	2*BYTE	B#16#10	siehe 2.4.3.1 - Die MMF
8.0	usIndex	2*BYTE	B#16#00	siehe 2.4.3.1 - Die MMF
10.0	usSize	2*BYTE	B#16#00	siehe 2.4.3.1 - Die MMF
12.0	usMaxItems	2*BYTE	B#16#10	siehe 2.4.3.1 - Die MMF
14.0	mp	2*BYTE	B#16#00	Meldepunkt, an dem sich die Station befindet
16.0	pos	CHAR	' '	Position der Station ab Meldepunkt
17.0	lage	CHAR	' '	Lage (links / rechts)
18.0	typ	CHAR	' '	(Hole- / Bringeziel; Universal, etc.)
19.0	fz_typ	CHAR	' '	Fahrzeugtypen, die zur Station dürfen
20.0	gesp	CHAR	' '	1 = Station gesperrt; 0 = frei
21.0	gest	CHAR	' '	1 = Station gestört; 0 = fehlerfrei
22.0	prio	CHAR	' '	Anfangspriorität für Aufträge
23.0	last	CHAR	' '	1 = Station beladen; 0 = Station frei
24.0	KundenNr	ARRAY	' '	Kundenname der Station
32.0	anz_ebenen	CHAR	' '	Anzahl der Ebenen, die die Station haben kann.

Tabelle 17 : Struktur – DB 545

4.4. Beispiel für den Telegramm-Ablauf

In den vorhergegangenen Kapiteln wurde des öfteren das Sperren einer Station als Beispiel verwendet. Es folgt eine komplette Abarbeitung, vom Bedienen im BuB über den Telegrammverkehr bis zur Anzeige im BuB.

4.4.1. Auslösen des Telegramms

Um eine Station der Anlage zu sperren, muss die Station in der Stationsliste aktiviert sein. Anschließend kann man mit Hilfe des Bedienpanels an der rechten Seite die Station sperren.

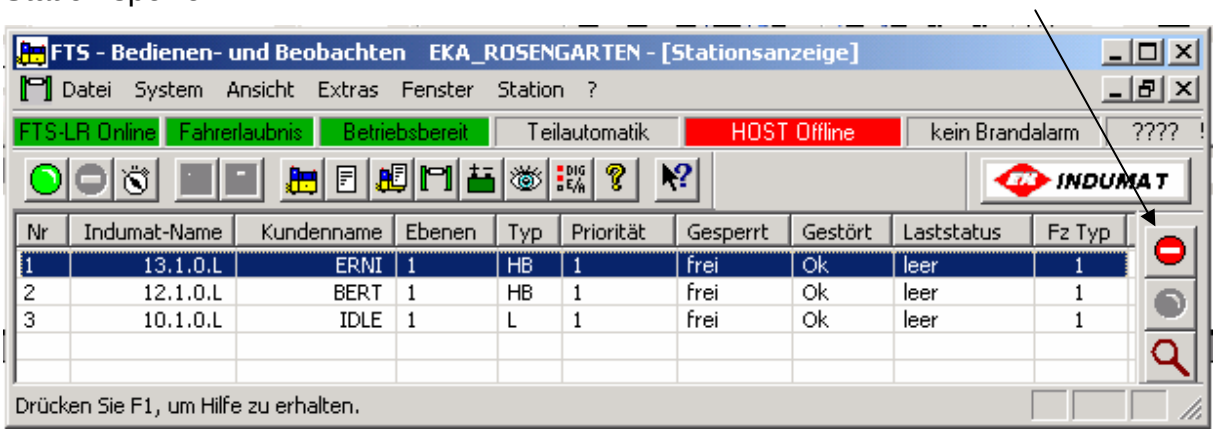


Abbildung 57 : BuB – Station freigegeben

Durch das Anklicken des „*Sperren*“ Buttons wird ein ZK – Telegramm ausgelöst. Dieses wird durch TCPCOM an das OS300 geschickt und dort empfangen.

Das Log-Fenster von TCPCOM wandelt die Datenformate für die Anzeige. Im Fenster wird für mp.nr hier zum Beispiel ein CHAR ’**o**’ angezeigt, welches der Dezimalzahl 13 entspricht. Damit die Telegramme leichter zu verstehen sind werden hier alle CHAR, die als Zahl interpretiert werden müssen, in Dezimalzahlen angegeben. Im Absatz 4.4.7 werden die eigentlichen Telegramme in Originalform angezeigt.

BuB => OS				
Name des Telegramms	Name	Variablenart	Inhalt	Beispiel
Stationen	kenn	unsigned char[2]	ID des MMF	**ZK** (CHAR)
	qkenn	unsigned char[2]	Quittungskennung	**QT** (CHAR)
	mp.nr	unsigned short	Nummer des Meldepunktes, an dem sich die Station befindet	**13** (Dez)
	mp.pos	unsigned char	Position der Station (von Meldepunkt aus)	**1** (DEZ)
	mp.lage	unsigned char	L = links / R = rechts	**L** (CHAR)
	mp.ebene	unsigned char	Ebene der Station (Hochregallager)	**0** (DEZ)
	ziel_stat	unsigned char	0 freigeben 1 sperren	**1** (DEZ)

Tabelle 18 : Inhalt – ZK – Telegramm

In der Tabelle 18 : Inhalt – ZK – Telegramm kann man das Telegramm sehen welches, in diesem Beispiel, an das OS300 gesendet wurde. Die BuB sendet die nötigen Daten an das OS300, damit dieses die jeweilige Station sperren kann. Die Station selber wird durch die Meldepunktdaten (Nummer, Position, Lage und Ebene) eindeutig bestimmt. Abschließend folgt der Befehl, was mit der Station geschehen soll (hier sperren).

4.4.2. Abarbeiten des recieve buffers

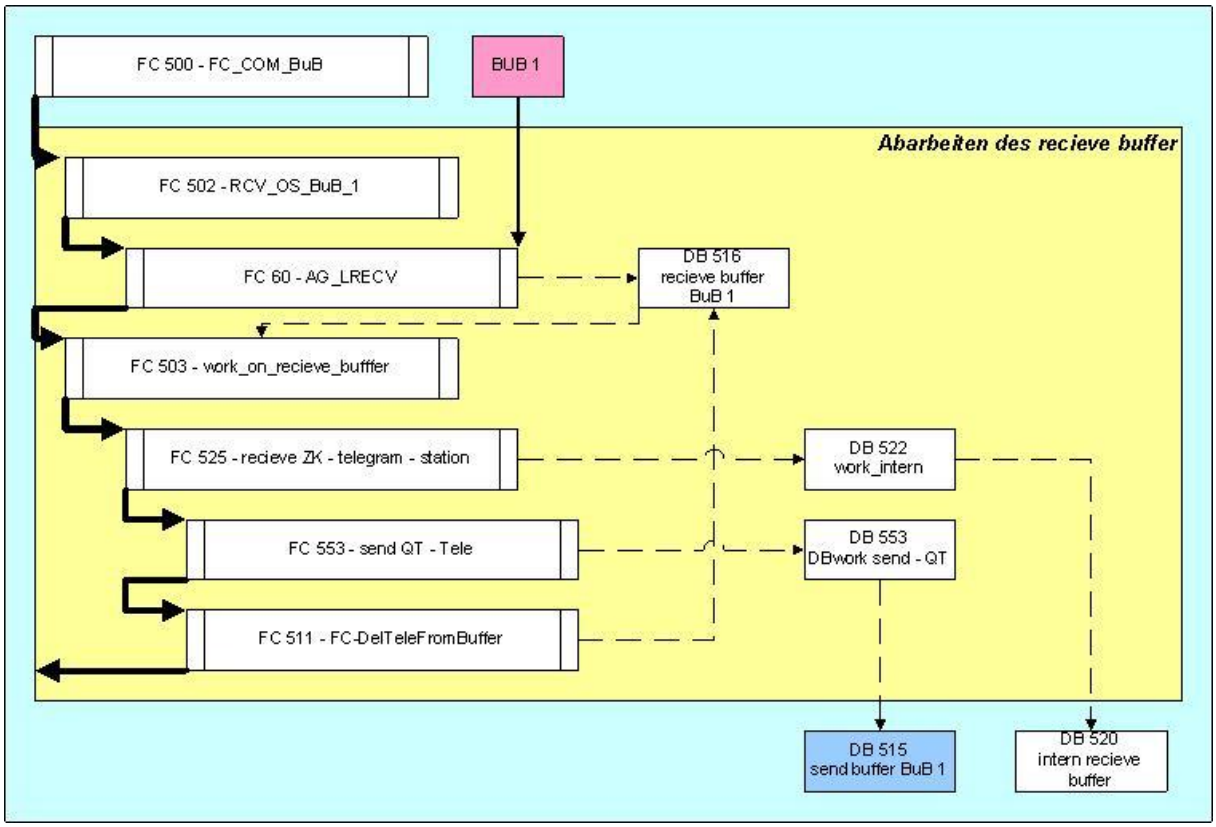


Abbildung 58 : Beispiel : S0 – Telegram – recieve

Wenn das Telegramm von der BuB empfangen wurde (FC 60), befindet es sich noch in den Rahmendaten, welche TCPCOM um die Nutzdaten legt. Diese Rahmendaten wurden bereits im Absatz 2.4.3 / Tabelle 3) näher erläutert.

Der FC 502 entfernt diese Rahmendaten und schreibt die Nutzdaten in den *recieve buffer* (DB 516) der BuB 1. Hier liegen die Daten im Nutzdatenbereich (Abbildung 44 : Struktur der Buffer) des Buffers, wie in Tabelle 18 angegeben, vor. Anschließend wird aus dem FC 500 der FC 503 aufgerufen, welcher immer das erste Telegramm des Buffers ausliest. Der FC 503 erkennt, dass es sich um ein ZK – Telegramm handelt und startet den FC 525. Dieser kopiert die Telegrammdaten in den DB 522, aus dem sie dann in den *intern recieve buffer* eingetragen werden. Ist das Eintragen erfolgreich, wird das ZK – Telegramm aus dem *recieve buffer* mittels des FC 511 gelöscht. Außerdem ruft der FC 525 wiederum den FC 553 auf, welcher im DB 553 ein Quittungstelegramm zusammenstellt und dieses in den *send buffer* (DB 520) der BuB 1 kopiert, aus dem dieses später gesendet wird (Absatz 4.4.5 - Abarbeiten des send buffers).

OS => BuB				
Name des Telegramms	Name	Variablenart	Inhalt	Beispiel
QT – Quittungs-tele	ucID[2]	unsigned char	ID des MMF	**QT** (CHAR)
	usType	unsigned short	Ringpuffer / tabelarisch	**00** (DEZ)
	usValid	unsigned short	immer 1	**01** (DEZ)
	usIndex	unsigned short	immer 0, da nur ein QT-Tele	**00** (DEZ)
	usSize	unsigned short	Länge der Nutzdaten	**06** (DEZ)
	usMaxItems	unsigned short	immer 1, da nur ein QT-Tele	**01** (DEZ)
	kenn	unsigned short	Quittungskennung des empfangenen Telegramms	**QT** (CHAR)
	qkenn	unsigned short	Kennung des empfangenen Telegramms	**ZK** (CHAR)
	error_code	unsigned short	error code	**00** (DEZ)

Tabelle 19 : Inhalt - QT - Telegramm

4.4.3. Auswertung des intern recieve buffers

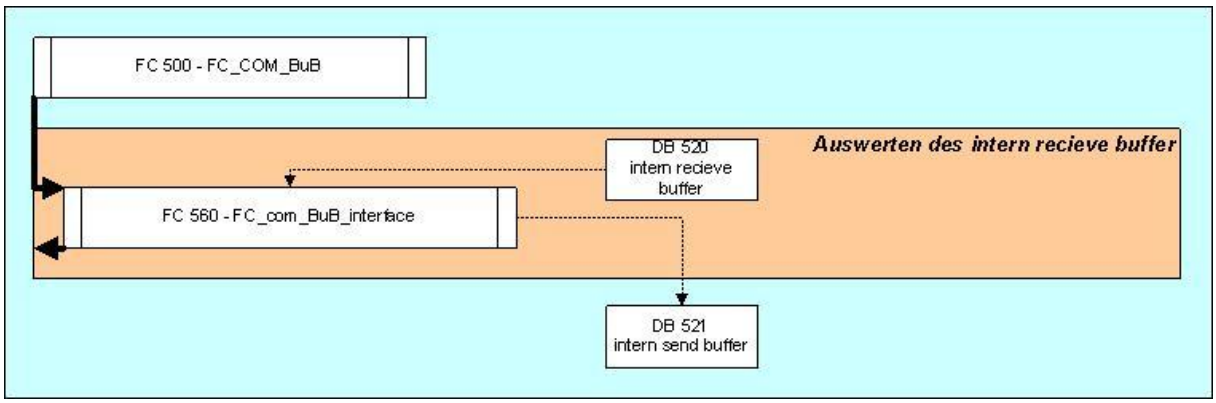


Abbildung 59 : Beispiel : S0 – Telegram – intern recieve

Nach dem Empfang von Telegrammen erfolgt die Abarbeitung des *intern recieve buffer*. Dabei wird immer das erste Telegramm des Buffers ausgelesen und der in ihm enthaltene Befehl ausgeführt. Im Beispiel wäre es das Sperren der Station Nummer 1. Nach dessen Sperrung wird auch das Telegramm aus dem *intern recieve buffer* gelöscht.

Da in dem OS300 ein Vergleich des alten mit dem aktuellen Systemzustand stattfindet, erkennt das OS, dass die Station 1 gesperrt wurde und schreibt schließlich eine S0 – Telegrammanforderung in den *intern send buffer*.

4.4.4. Abarbeiten des intern send buffers

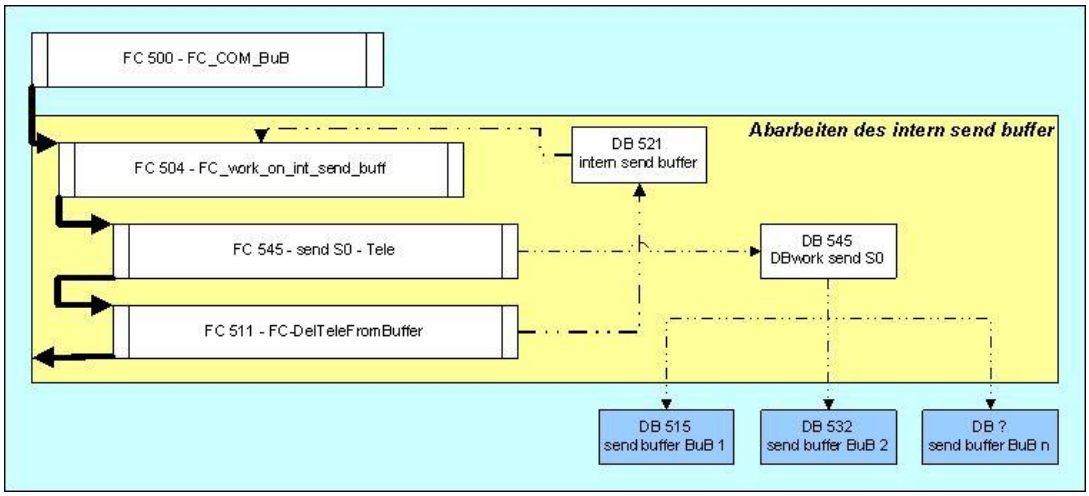


Abbildung 60 : Beispiel : S0 – Telegram – intern send

Die Anforderung im *intern send buffer* wird durch den FC 504 erkannt. Dadurch wird der FC 545 gestartet, welcher als Eingangsparameter die Stationsnummer, der zu sendenden Daten der Station, erhält. Der FC 545 stellt die Telegrammdaten im DB 545 zusammen und kopiert diese dann in die *send buffer* aller BuB. Wenn dies erfolgreich war, wird die Anforderung aus dem *intern send buffer* gelöscht.

OS => BuB				
	Symbol	Variablenart	Variablenart	Beispiel
S0 - Station	ucId[2]	unsigned char	ID des MMF	**S0** (CHAR)
	usType	unsigned short	Ringpuffer / tabelarisch	**00** (DEZ)
	usValid	unsigned short	immer 1	**10** (DEZ)
	usIndex	unsigned short	0, da Stationsnummer - 1	**00** (DEZ)
	usSize	unsigned short	Länge der Nutzdaten	**19** (DEZ)
	usMaxItems	unsigned short	3, da die Anlage 3 Stationen hat	**30** (DEZ)
	mp	unsigned short	Meldepunkt	**13** (DEZ)
	pos	unsigned char	Position	**01** (DEZ)
	lage	unsigned char	L = links / R = rechts	**L** (CHAR)
	typ	unsigned char	Typ (3 = Universal station)	**3** (DEZ)
	fz_typ	unsigned char	Fahrzeugtyp	**1** (DEZ)
	gesp	unsigned char	0 = frei / 1 = gesperrt	**1** (DEZ)
	gest	unsigned char	0 = OK / 1 = gestört	**0** (DEZ)
	prio	unsigned char	Priorität	**1** (DEZ)
	last	unsigned char	0 = frei / 1 = belegt	**0** (DEZ)
	KundenNr	unsigned char	Kundenname	**ERNI 00**(CHAR)
	anz_ebenen	unsigned char	Anzahl der Ebenen, die die Station hat	**1** (DEZ)

Tabelle 20 : Inhalt – S0 – Telegramm

4.4.5. Abarbeiten des send buffers

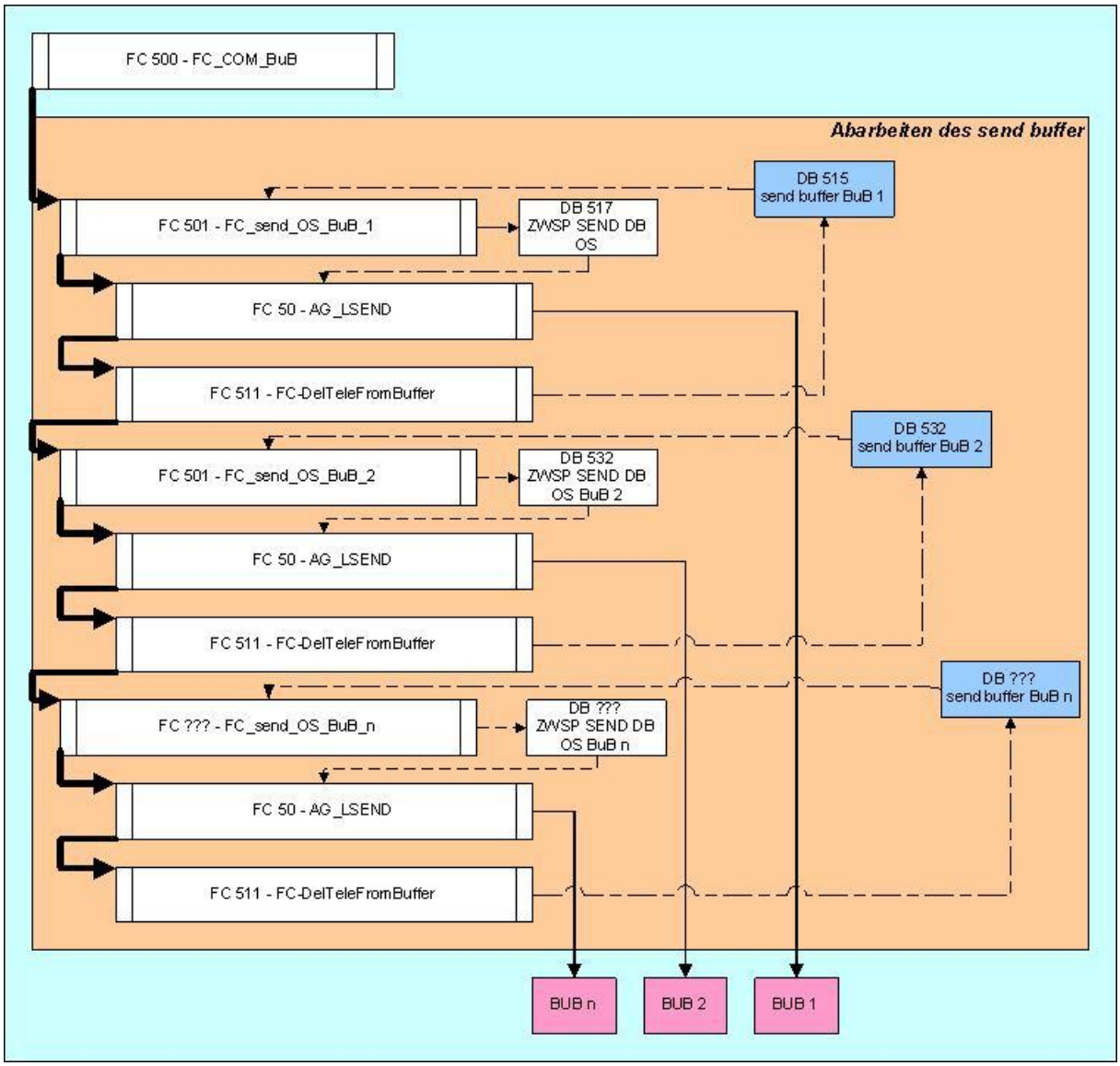


Abbildung 61 : Beispiel : S0 – Telegram –send

Schließlich folgt noch das Senden der Telegramme. Hierzu werden nacheinander die Telegramme aus den einzelnen *send buffer*n in einen Zwischenspeicher kopiert. Dabei werden wieder die Rahmendaten (Tabelle 3 : TCPCOM : Steuerungsdaten) um das Telegramm gelegt. Anschließend wird es mit Hilfe des FC 50 an die BuB gesendet. Danach wird das Telegramm mittels FC 511 aus dem *send buffer* gelöscht. Damit endet die Abarbeitung des Befehles.

Die Telegrammanforderungen haben am Anfang immer dieselbe Struktur. Nur der Nutzdatenteil ändert sich je nach angefordertem Telegramm.

OS => BuB				
	Offset auf buffer	**Art**	**Beschreibung**	**Beispiel**
Tele-gramm-anfor-derung	0.0	WORD	Länge der Anforderung	**08** (DEZ)
	2.0	WORD	Modulnummer (zur Zeit nicht verwendet)	**00** (DEZ)
	4.0	WORD	MMF ID des zu sendenden Telegramms	**'S0'** (CHAR)
	ab 6.0	WORD	Nutzdaten	**01** (DEZ)

Tabelle 21 : Struktur - Telegrammanforderungen

In den Nutzdaten des S0 – Telegramms steht nur die Nummer der Station, deren Daten an die BuB gesendet werden soll. Dies ist in diesem Beispiel die Stationsnummer 1.

Telegramm	**Offset**	**Art**	**Beschreibung**
S0 – Telegramm	6.0	WORD	Stationsnummer
F0 – Telegramm	6.0	WORD	FTF-Nummer
A0 – Telegramm	6.0	WORD	Auftragsnummer
	8.0	WORD	Indexnummer (wenn <> 32767 dann Auftrag löschen)
DT - Telegramm	6.0	WORD	Modulnummer
	8.0	WORD	Meldungsnummer
	10.0	WORD	kommt / geht
	12.0	WORD	Anzahl der Variablen
	14.0	WORD	Variable 1
	16.0	WORD	Variable 2
	18.0	WORD	Variable 3
	20.0	WORD	Variable 4
	22.0	WORD	Variable 5
	24.0	WORD	Variable 6
	ab 26	STRING	leer String
M0 – Telegramm	6.0	WORD	FTF-Nummer
Z0 – Telegramm	----	-----	keine Nutzdaten

Tabelle 22 : Nutzdaten der Telegrammanforderungen

4.4.6. Anzeige in der BuB

Nachdem TCPCOM das Telegramm erhalten hat wird dieses im dazugehörigen MMF abgelegt. Die BuB erhält dann eine Benachrichtigung, dass sich Daten geändert haben. Dadurch wird die Anzeige aktualisiert und es wird der aktuelle Status (Station gesperrt) angezeigt.

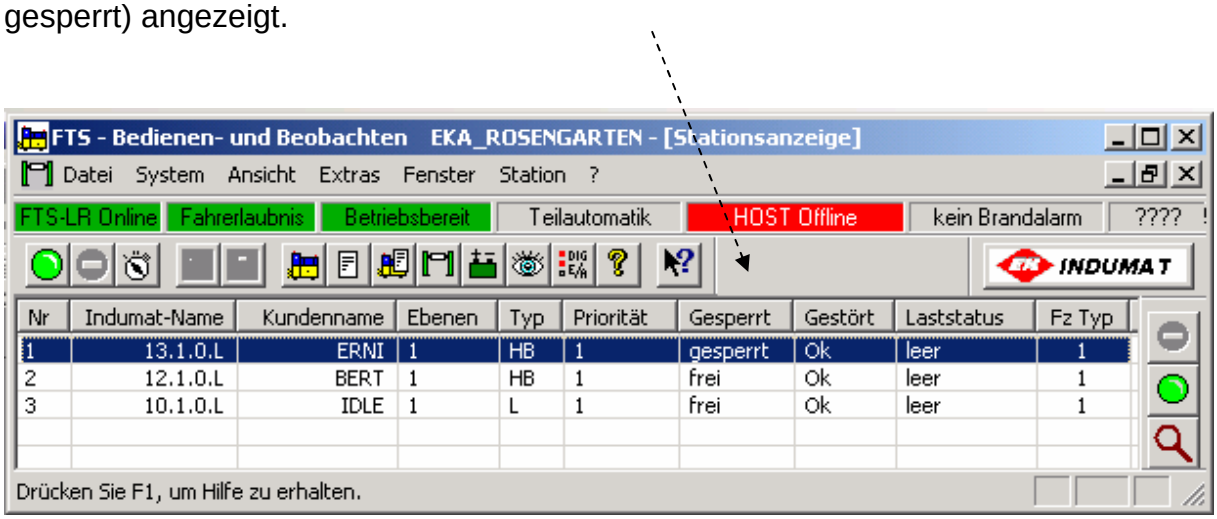


Abbildung 62 : BuB – Station gesperrt

Der gesamte Telegrammverkehr wird im TCPCOM Log Fenster angezeigt.

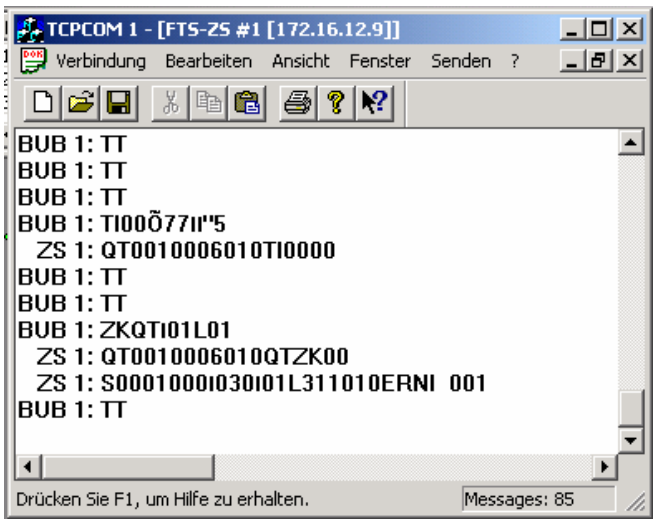


Abbildung 63 : TCPCOM – Log Fenster – Station sperren

4.4.7. Zusammenfassung

In den vorherigen Abschnitten wurde die Abarbeitung eines Befehles einmal ausführlich beschrieben. Es wurde davon ausgegangen, dass zu diesem Zeitpunkt keine anderen Befehle oder Anforderungen in den einzelnen Buffern standen. In diesem Fall wäre die Reaktionszeit (Aktion **ß à** Reaktion) sehr gering. Da aber während des Betriebes sehr viele Daten bearbeitet werden, kann es zu längeren Bearbeitungszeiten kommen, welches aber nicht relevant ist, da es sich um ein zeitunkritisches System handelt. Außerdem bewegt sich die Zykluszeit der SPS im ms Bereich. In Abbildung 65 sieht man nochmals die einzelnen Abbildungen der vorhergegangenen Abschnitten zusammengefügt.

Abbildung 64 : BuB – Station gesperrt - Detailansicht

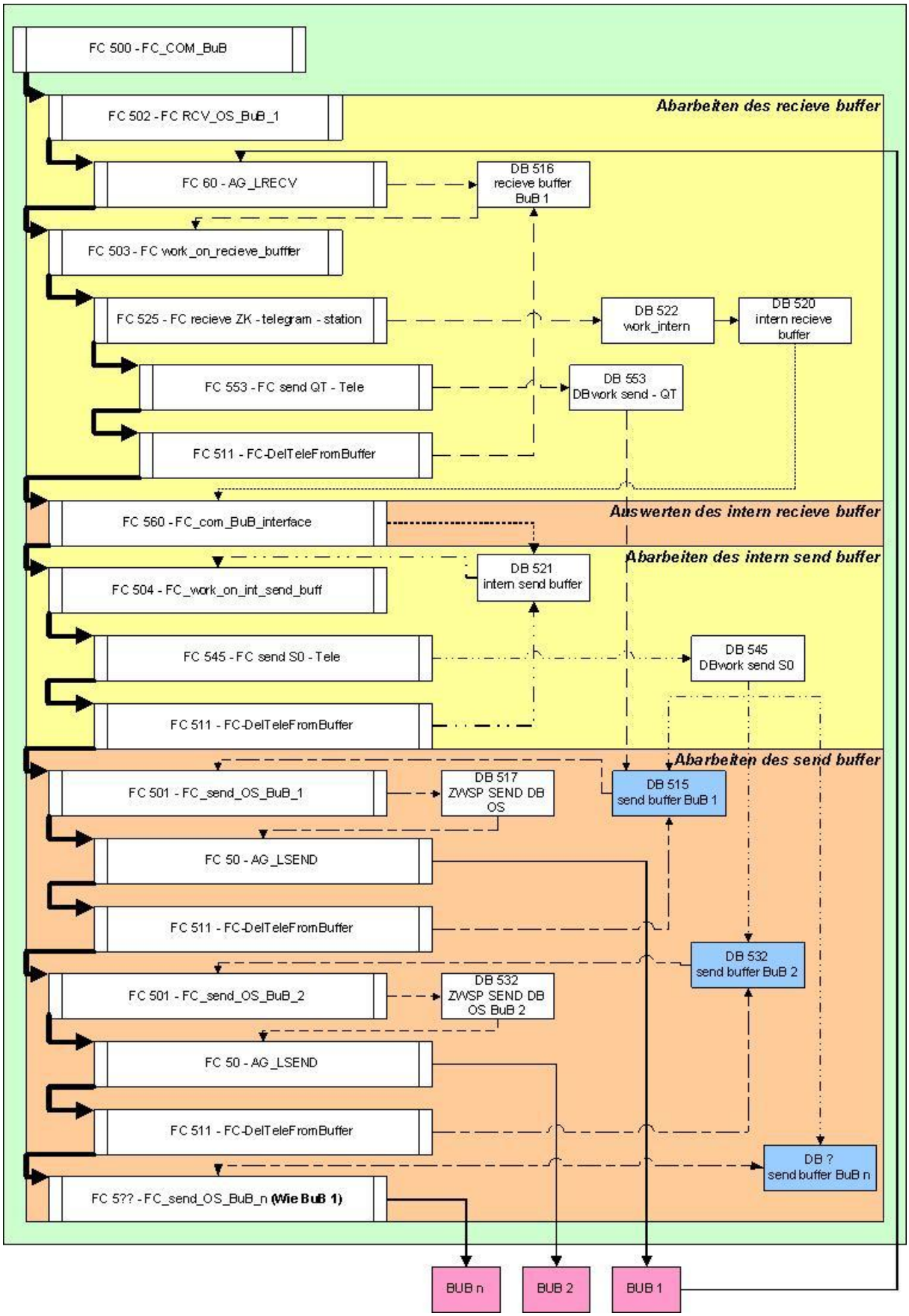


Abbildung 65 : Beispiel : S0 – Telegram – gesamt

5. Vergleich BuB / WinCC

Um den Nutzen der Arbeit darzustellen werden hier das „neue“ und das bisher benutzte System gegenübergestellt. Hierzu werden die Anschaffungskosten und der Entwicklungsaufwand für die Erstellung einer funktionierenden HMI näher betrachtet. Es wird von der Standard-Software ausgegangen, so dass im Entwicklungsaufwand nur auf die projektspezifischen Anpassungen eingegangen wird. Als Beispielanlage wird die CON-LOG AG benutzt, für die gerade die Software geschrieben wird. Es wird eine BuB in diesem Projekt eingesetzt.

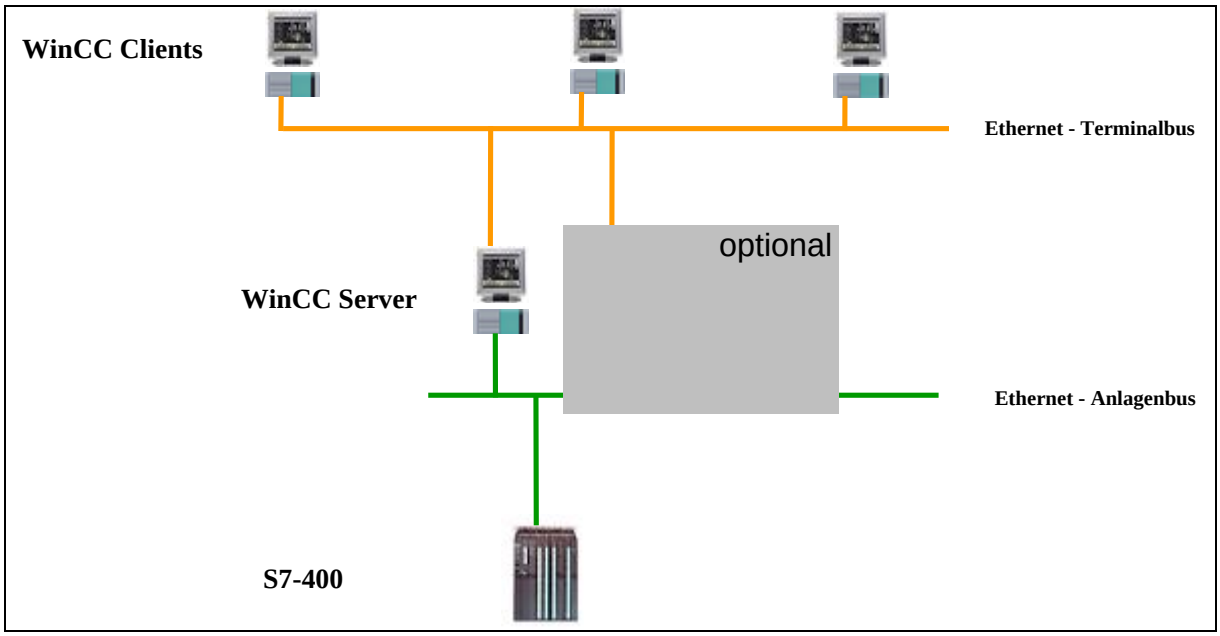


Abbildung 66 : Visualisierungskonzept WinCC[10]

Die Hardware – Konfiguration für das BuB – System ist ähnlich der des WinCC – Systems. Ansonsten wurde für die Vergleichbarkeit die gleiche Konfiguration gewählt.

- S7 – 400 mit CP 443 – 1 und PS 407 4A
- 4 PC für die Visualisierung

[10] Konzept.ppt; Siemens 22.07.2005

5.1. Anschaffungskosten

Die Vergleichsanlage verfügt über 4 HMI Plätze. Diese sollen mittels Ethernet mit der Steuerung verbunden sein. WinCC benötigt einen High-End Rechner, die BuB läuft hingegen schon auf einem 800 Mhz - Rechner.

Die Kosten des Entwicklungstools werden hier nicht betrachtet, da diese nur einmal angeschafft werden müssen.

Da der WinCC-Server direkt auf die Daten der SPS zugreift, wird ein CP benötigt, der für die Kommunikation zwischen PC und SPS sorgt.

Hardware	WinCC (€)[11]	BuB (€)
1. PC (4)	8000,00	4000,00
- inkl. WinXP		
- inkl. TFT		
2. CP (Communication Prozessor) 1612	120,00	
3. S7		
- CPU 416 – 2	4908,00	4908,00
- CP 443 – 1	1293,00	1293,00
- PS 407 4A	196,00	196,00
- Rack	282,00	282,00
Summe:	**14799,00**	**10679,00**

Tabelle 23 : Vergleich – Hardwarekosten (WinCC / BuB)

Außerdem wird bei WinCC mindestenz eine Server-Lizenz benötigt, da dieser die Daten aus der SPS lädt und diese dann für die Clients (greifen nicht direkt auf die SPS zu) bereitstellt. Wenn also der Server ausfällt, werden die Clients nicht mehr mit den Anlagendaten versorgt. Deswegen sollte ein zweiter Server (Redundanz) eingesetzt werden („optional“ in Abbildung 66 : Visualisierungskonzept WinCC) .

Software	WinCC (€)[12]	BuB (€)
1. WinCCServer	3000,00	
2. WinCC Runtime 1024 PT	4000,00	
3. WinCC Runtime 128 PT (3 * 2100,00€)	8400,00	
4. Lizenzgebühren BuB (INDUMAT)		0,00
Summe:	**15400,00**	**0,00**

Tabelle 24 : Vergleich – Softwarekosten (WinCC / BuB)

[11] Kalkulation_Soltau_Diplom.xls; Siemens 22.07.2005

[12] Kalkulation_Soltau_Diplom.xls; Siemens 22.07.2005

5.2. Entwicklungsaufwand

Um ein HMI für das jeweilige Projekt anzupassen muss ein Layout erstellt werden. Daraufhin muss jede Strecke des Layouts noch mit Daten versorgt (gekoppelt) werden. Bei WinCC muss diese Kopplung per Hand erfolgen, d.h., dass jeder Strecke in WinCC die dazugehörige Strecke in der SPS zugewiesen werden muss. Da die BuB über die Streckendaten verfügt, erfolgt diese Zuweisung automatisch. Es muss nur ein Tool gestartet werden, welches die Zuweisung erledigt.

Art der Arbeit	WinCC (Std)	BuB (Std)
1. Erstellen des Layouts WinCC - 1 Layout Gesamtzeichnung (meistens Kundenvorgabe in CAD) muss bearbeitet/bereinigt werden BuB - 1 Layoutzeichnung - 1 Hintergrundzeichnung - 1 Positionszeichnung der FTS (dadurch Punkt 2 nicht nötig)	4	6
2. Verknüpfen der Daten mit dem Layout	14	1
Summe:	**18**	**7**

Tabelle 25 : Vergleich – Entwicklungsaufwand

5.3. Funktionalität

In Hinsicht der Funktionalität der beiden Systeme gibt es nicht sehr viele Unterschiede. Ein Vorteil von dem BuB an sich ist es, dass bei lasergeführten Anlagen (nicht OS300) relativ einfach eine kontinuierliche Visualisierung erstellt werden kann.

Der Vorteil von WinCC ist, dass man nicht FTS spezifische Visualisierungteile (z.B. Prozessautomatisierung) realisieren kann, ohne das programmiert werden muss. Es müssen nur die anzuzeigenden Grafiken eingefügt und mit den Daten der SPS verknüpft werden. Dies ist auf Seiten der BuB nicht so einfach Möglich, da zuvor eine neue Telegrammart und deren Bearbeitung programmiert werden müsste.

5.4. Vor- / Nachteile

Art der Arbeit	WinCC	BuB
1. Hardwarekosten	14799,00 €	10679,00 €
2. Softwarekosten	15400,00 €	0,00 €
3. Entwicklungsaufwand	18 Stunde(n)	7 Stunde(n)
4. Anerkanntes HMI - Tool	+	-

Tabelle 26 : Vergleich – Vor- / Nachteile

Durch den Vergleich lässt sich sehr klar erkennen, dass das BuB-System einige Vorteile hat. Es fallen unter anderem keine Lizenzgebühren an. Außerdem ist der Entwicklungsaufwand für die projektspezifischen Anpassungen nicht sehr hoch. Jedoch ist WinCC ein Standart-Visualisierungstool, dessen Einsatz die Kunden des öfteren fordern. Dadurch wird WinCC sicherlich weiterhin eingesetzt werden.

6. Zusammenfassung / Ausblick

Zusammenfassend lässt sich sagen, dass die Abwicklung des „Projektes“ reibungslos stattfand. Sowohl von Seiten Eilers & Kirf Rosengarten, also auch von Indumat Reutlingen war die Betreuung durch die Mitarbeiter sehr gut. Da Herr Sagewitz des öfteren auf Inbetriebnahmen war, kam es manchmal zu Verzögerungen, weil ich auf Informationen von ihm angewiesen war. Diese Verzögerungen waren aber eingeplant.

Die Schnittstelle zwischen OS300 und BuB funktioniert so, wie es im Pflichtenheft gewünscht ist. Im Rahmen eines Tests wurde diese Funktionalität überprüft. Es werden alle gewünschten Telegramme gesendet, empfangen und ausgewertet. Wie im Absatz 4 bereits beschrieben, wurde die Schnittstelle so programmiert, dass sie nicht mit dem restlichen OS300 verzahnt ist.
Im Rahmen einer firmeninternen Präsentation wird dem Vertriebspersonal die Funktionalität und die Kostenkalkulation vorgelegt. Dadurch ist das Vertriebspersonal in Zukunft in der Lage dem Kunden das „neue“ Produkt anzubieten.

Das BuB – System wird demnächst in der E&K AUTOMATION Gruppe als Standardvisualisierung eingesetzt werden. Dadurch ergeben sich eine Menge Vorteile für das Unternehmen. Es können Servicemitarbeiter von jedem Standort mit dem Tool umgehen und eventuelle Veränderungen tätigen. Außerdem verringert sich die Zeit, die für projektspezifische Anpassungen notwendig ist. Dadurch kann E&K AUTOMATION seine Position auf dem Markt festigen, da das Unternehmen somit die FTS günstiger anbieten kann.

Durch das Projekt wurde ein weiterer Schritt in Richtung des Zusammenwachsens der verschiedenen Standorte gegangen.

Des Weiteren soll im Hause E&K eine Testfirma „EK CON-LOG AG“ aufgebaut werden, um dem Kunden eine ganzheitliche Lösung für die Prozessautomatisierung und den Materialtransport anbieten und präsentieren zu können. In dieser „Firma“ soll unter anderem das BuB - System für die Visualisierung der integrierten FTS - Anlage genutzt werden.

Neue Anwendungsmöglichkeiten des Systems:

- Das SMS – Modul für nicht FTS – Anlagen benutzen (Prozessautomatisierung)
- In der BuB auch Daten die nicht FTS spezifisch sind anzeigen (Ventile, Motoren, Digitale Signale (Füller aktiv/inaktiv))
- TCPCOM auf MPI (Bussystem) Funktionalität umsetzten
- Nutzen des Systems für Staplerleitsysteme
- Nutzen für reine Fördertechnik
- Anbindung an MES (Manufacturing Executiv System)

7. Quellen

7.1. Literaturverzeichnis

- E&K Firmenpräsentation; Stand 07.2005
- Pflichtenheft-Diplomarbeit Stefan Soltau V1.0.doc; 17.06.2005
- IEC 61131 – Wozu?, Ingo Rolle (Hrsg.),1998
- www.e-technik.fh-kiel.de/ regiue/XSPS.html; 20.05.2005; 12:30 Uhr
- Systemdokumentation : „BuB – Beobachten und Bedienen“
- Kalkulation_Soltau_Diplom.xls; Siemens 22.07.2005

7.2. Formelverzeichnis

7.3. Tabellenverzeichnis

7.4. Abbildungsverzeichnis

8. Erklärung zur Diplomarbeit

Name : Soltau
Vorname : Stefan
Matr.-Nr. : 153590
Studiengang : Angewandte Automatisierungstechnik

An den Prüfungsausschuss
des Fachbereichs Automatisierungstechnik
der Universität Lüneburg
Volgershall 1

21339 Lüneburg

Erklärung zur Diplomarbeit

Ich versichere, dass ich diese Diplomarbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe.

Lüneburg, den 3. August 2005

_____

Stefan Soltau

9. Anhang

Der komplette Anhang der Diplomarbeit befindet sich auf der beiliegenden CD.

9.1. Die Telegramme

...//9. Anhang/9.1 Die Telegramme.pdf

9.2. Quelltext

...//9. Anhang/9.2 Quelltext / *.AWL können im Editor geöffnet werden

9.3. Pflichtenheft

...//9. Anhang/9.3 Pflichtenheft.pdf

9.4. BuB Systemdokumentation

...//9. Anhang/9.4 BuB Systemdokumentation.pdf