BAKKALAUREATSARBEIT

BACnet/KNX Gateway für KNXcalibur

ausgeführt am

Institut für Rechnergestützte Automation
Arbeitsgruppe Automatisierungssysteme

der Technischen Universität Wien

unter der Anleitung von

ao. Univ.-Prof. Dipl.-Ing. Dr. Wolfgang Kastner
und
Dipl.-Ing. Wolfgang Granzer

durch

Jürgen Weidinger, MatrNr. 0325317
jweidinger@auto.tuwien.ac.at

Rottersdorf 59
3125 Statzdendorf

Wien, 31.10.2007

Abstract

Building Automation Systems (BAS) aim at improving control and management of mechanical and electrical systems in buildings. The core application area of BAS is environmental control with its traditional services heating, ventilation and air conditioning and lighting/shading. The main drivers for the employment of BAS are increased comfort at the one hand and considerable cost efficiency at the other. Additionally, these systems significantly contribute to energy saving and also facilitate the management of the building. Numerous different technologies and protocols for BAS exist. Among the openly available ones, KNXcalibur and KNX/EIB are most important.

The main aim of this work is the development of a BACnet/KNX gateway. It is based on a KNX hardware platform which was developed at the Institute (KNXcalibur). An existing software implementation for this platform is successively extended with the BACnet/KNX gateway functionality.

This document consist of two main parts. First, a short overview of the history and the main features of BACnet is given. Focus is put on aspects related to the BACnet/KNX gateway functionality. In the second part the actual implementation of the gateway is described. Its main feature is the access and manipulation of application objects both from the BACnet and KNX network side. As a proof-of-concept a binary application object for lighting has been implemented.

Kurzfassung

Systeme der Gebäudeautomation beschäftigen sich mit der automatischen Steuerung und Regelung von gebäudetechnischen Subsystemen. Dazu zählen primär Licht- und Beschattungssysteme sowie Anlagen zur Regelung von Heizung, Lüftung und Klima. Neben gesteigertem Komfort lassen sich durch automatische Steuerung der Gebäudeanlagen bedeutende Kostenvorteile erzielen. Einerseits können diese Systeme den Energieverbrauch signifikant reduzieren, andererseits helfen sie den Verwaltungsaufwand zu senken. Zur Realisierung solcher Automationssysteme stehen eine Vielzahl verschiedener Technologien zur Verfügung. Zwei wichtige Vertreter sind die offenen Kommunikationsstandards KNX/EIB und BACnet.

Ziel der vorliegenden Arbeit ist es ein BACnet/KNX Gateway zu implementieren. Als Grundlage dient eine bestehende KNX Hardwareplattform (KNXcalibur), welche am Institut entwickelt wurde. Dabei soll im Zuge dieser Arbeit die vorhandene KNX Softwareimplementierung mit einer entsprechender BACnet/KNX Gatewayfunktionlität erweitert werden.

Dieses Dokument gliedert sich in zwei Teile. Im ersten Teil wird eine kurze, allgemeine Einführung über BACnet gegeben. Neben einem geschichtlichen Überblick über die Entstehung von BACnet werden die Grundkonzepte von BACnet erläutert. Dieser Kurzüberblick sowie die gebrachten Beispiele sind vorallem auf die Implementierung der BACnet/KNX Gateway Funktionlität ausgerichtet. Im zweiten Teil wird die Implementierung der Gatewayfunktionalität selbst beschrieben. Diese Implementierung ermöglicht es, Applikationsobjekte sowohl vom KNXcalibur als auch vom BACnet Netzwerk aus zu steuern. Als Proof-of-Concept wurde ein binäres Applikationsobjekt implementiert, mit welchem eine Lichtquelle gesteuert werden kann.

Inhaltsverzeichnis

1 Übersicht **6**

1.1 Projektumfeld 6

1.2 Aufgabenstellung 6

2 BACnet **7**

2.1 BACnet Funktionsweise 8

2.1.1 Objekte 8

2.1.2 Eigenschaften (Properties) 9

2.1.3 Objektzugriff 9

2.2 Netzwerkkonzept vom BACnet 10

2.2.1 Schichtenmodell 10

2.2.2 Kommunikationsablauf 13

2.2.3 BACnet Ethernet Frame 15

3 Implementierung **15**

3.1 Übersicht Objekt-Implementierung 15

3.2 Binär Ausgabe-Objekt (Binary Output) 16

3.3 Objekt Access Service 17

3.4 Remote Device Management Service 18

3.5 Ethernet Implementierung 18

3.6 Beschreibung des Open Source Stacks 19

3.6.1 Ordnerstruktur 19

3.6.2 Wichtige Dateien 20

3.6.3 BACnet Stack Dateien 20

3.6.4 Beispiele für BACnet Objekte 21

3.6.5 Dateien für die Sicherungsschicht 21

3.6.6 Von der Implementierung der Sicherungsschicht abhängige Dateien 21
3.6.7 main.c 22
3.7 Entwicklungsumgebung 25
3.8 Kommunikationsablauf auf KNXcalibur 25
3.9 Write Property Frame 25
3.10 Benutzerhandbuch 26
3.10.1 Testwerkzeug Visual Test Shell 26
3.10.2 Applikation auf KNXcalibur laden 26
3.10.3 Applikation auf KNXcalibur ausführen 26

4 Prototypimplementierung **29**

5 Aufgetretene Probleme **29**

6 Verwendete Abkürzungen **30**

1 Übersicht

1.1 Projektumfeld

KNX/EIB (**E**uropean **I**nstallation **B**us) ist ein offener Standard für die Gebäudeautomation [KNX04], der die herkömmliche Elektroinstallation in einem Gebäude durch ein Automationsnetzwerk ergänzt oder gänzlich ersetzen soll. Zur Kommunikation zwischen Sensoren und Aktoren werden Telegramme verwendet, die über ein Busmedium (Twisted-Pair TP, Powerline oder Funk) verschickt werden. Durch den Einsatz von dezentral organisierten Mikroprozessoren, ist eine zentrale Verarbeitungseinheit nicht notwendig [KNX07].

BACnet (**B**uilding **A**utomation and **C**ontrol **NET**work) ist ein offener Standard für den Gebäudeautomationsbereich [BAC04]. Ziel von BACnet ist es, eine Kommunikation verschiedener Geräte der technischen Gebäudeausrüstung (allen voran Heizung, Lüftung, Klimatechnik, Beleuchtung und Verschaltung) unterschiedlicher Hersteller zu gewährleisten. BACnet definiert die Anwendungs- und Netzwerk Schicht, das darunter liegende Netzwerk Medium ist frei wählbar. Die Kommunikation in BACnet erfolgt nach dem Client-Server Prinzip.

KNXcalibur ist ein universelles microcontroller-basierendes Gateway für die Gebäudeautomation [Pra05]. Das Gateway stellt verschiedene Hardwarekomponenten für ein Interfacing mit unterschiedlichen Gebäudeanforderungssystemen zur Verfügung.
Die TP-UART-Schnittstelle erlaubt eine Kommunikation mit KNX/EIB. Über die Ethernet Schnittstelle können BACnet Komponenten im Netzwerk angeschlossen werden. Weiters stehen zwei RS232, eine RS485 und eine USB Schnittstelle zur Verfügung. Ein SD/MMC Karte Interface erlaubt eine Vielzahl von Gebäudeautomationsfunktionen auf KNXcalibur zu implementieren. Für weitere Informationen sei auf [Pra05] verwiesen.

1.2 Aufgabenstellung

KNXcalibur soll als BACnet/KNX Gateway fungieren. Ziel der vorliegenden Arbeit ist es, eine Teilmenge des BACnet Stacks auf KNXcalibur zu implementieren. Die BACnet Kommunikation soll über Ethernet erfolgen. Als BACnetClient steht die **Visual Test Shell (VTS)** zur Verfügung. Deren Testumgebung ist auf Sourceforge.net (http://sourceforge.net/) frei verfügbar und wird kontinuierlich von der BACnet Manufacturers Association (BMA) gewartet [Kra05, Seite 140]. Für Testzwecke soll über den BACnet Client, VTS, ein Aktor, der an KNX/EIB angeschlossen ist, ein- bzw. ausgeschaltet werden. Der schematische Schaltungsaufbau ist in Abbildung 1 dargestellt.
Soll eine Nachricht vom BACnet Client zum KNX/EIB Aktor gesendet werden (das gleiche gilt in die andere Richtung), so ist ein Mapping der Nachricht mittels des als BACnet/KNX Gateways fungierenden KNXcalibur erforderlich. Die BACnet Nachricht wird vom Ethernet Controller empfangen. Der BACnet Stack bearbeitet die Nachricht und übergibt die Informationen dem entsprechendem BACnet Objekt. Es erfolgt nun ein Mapping der Information auf das KNX/EIB Objekt. Das KNX/EIB Objekt übergibt die Informationen dem KNX/EIB Stack, der eine entsprechende KNX/EIB Nachricht gene-

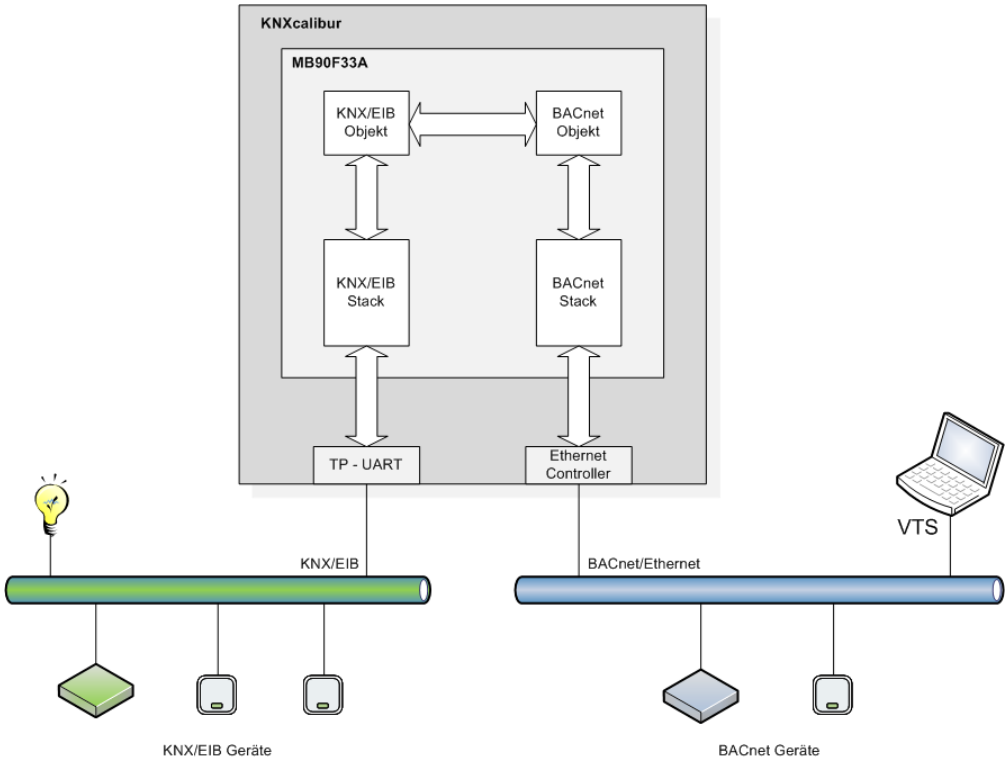

Abbildung 1: BACnet auf KNXcalibur

riert. Diese Nachricht wird vom TP-UART über das KNX/EIB Netzwerk versendet, wo die Nachricht vom entsprechenden Aktor empfangen und ausgewertet wird.

2 BACnet

BACnet wurde im Rahmen einer ASHRAE-Arbeitsgruppe (**A**merican **S**ociety of **H**eating, **R**efrigerating and **A**ir-Conditioning **E**ngineers) entwickelt, um einen firmenneutralen Standard für die Datenkommunikation in und mit Systemen der Gebäudeautomation bereitzustellen. Im Jahr 1995 wurde BACnet **ASHRAE-/ANSI-Norm (135)**, im Jahr 2003 wurde BACnet schließlich als **ISO-Norm (16484-5)** verabschiedet. Seither wird BACnet ständig weiterentwickelt. Die aktuelle Version ist BACnet 2004 [BAC04].

Der BACnet Standard stellt eine gemeinsame Schnittstelle für Gebäudeautomationsgeräte verschiedener Hersteller dar. Die Funktion eines Gerätes wird durch so genannte BIBBs (BACnet Interoperability Building Blocks) festgelegt. Diese definierten BIBBs ermöglichen eine herstellerübergreifende Interoperabilität. BACnet verfolgt einen objektorientierten Ansatz. Obwohl BACnet vorwiegend in der Management- und Automationsebene eingesetzt wird, kann es auch in der Feldebene verwendet werden (Abbildung 2).

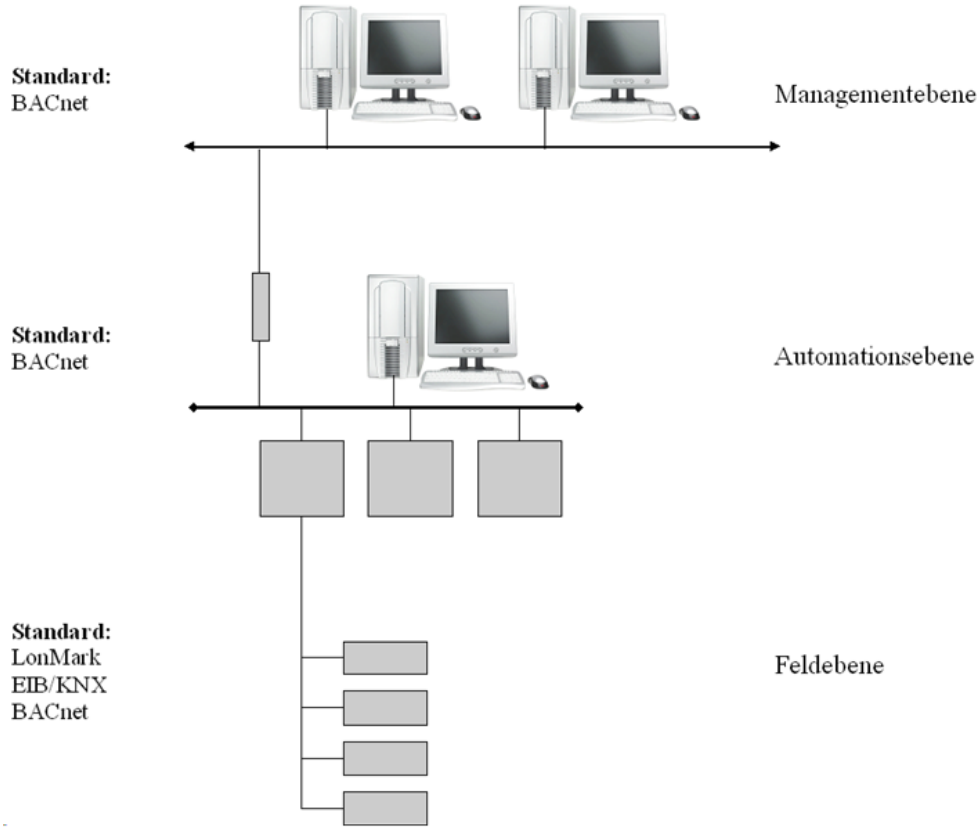

Abbildung 2: Ebenen der Gebäudeautomation

2.1 BACnet Funktionsweise

Damit BACnet Geräte miteinander kommunizieren können, gibt es Rahmenbedingungen, die im BACnet-Standard festgehalten sind [BAC04].

2.1.1 Objekte

BACnet verfolgt einen streng objekt orientierten Ansatz. Sämtliche Daten (sowohl Prozessdaten also auch Geräte-spezifische Daten) werden als Objekte repräsentiert. Objekte sind also physikalische Inputs, Outputs und Software Prozesse. Mit Hilfe dieser Objekte erhält man Zugriff auf geräte-spezifische Informationen sowie Informationen, die für die Funktion einer Gebäudeautomation notwendig sind, ohne im Detail den internen Aufbau oder die Konfiguration des jeweiligen Gerätes zu kennen.
Bisher sind im BACnet-Standard 25 Objekttypen definiert [Kra05]. Darunter fallen(u.a.):

- Analog Eingabe (Analog Input)
- Analog Ausgabe (Analog Output)
- Analogwert (Analog Value)

- Mittelwert (Averaging)
- Binäre Eingabe (Binary Input)
- Binäre Ausgabe (Binary Output)
- Binärwert (Binary Value)

2.1.2 Eigenschaften (Properties)

Objekte haben wiederum verschiedene Eigenschaften (Variablen, Parameter), die die Daten des Objekts selbst repräsentieren. Tabelle 1 zeigt beispielsweise die Eigenschaften, die für das Objekt `Binary Output` zur Verfügung stehen. In der linken Spalte sind die Eigenschaften eingetragen, in der Mitte der Datentyp. In der rechten Spalte findet man die jeweiligen Optionen, die für die ausgewählte Eigenschaft zur Verfügung stehen. Ein `R` steht dabei für einen Parameter, der verbindlich vorgeschrieben und lesbar sein muss; ein `W` für Parameter, die verbindlich vorgeschrieben sind, und verändert werden können; `O` steht für optionale Parameter.

2.1.3 Objektzugriff

Der BACnet Standard spezifiziert verschiedenste Dienste (Services) für den Objektzugriff. Derzeit werden 45 BACnet Dienste unterschieden, die nach ihren Funktionen geordnet, in verschiedenen Gruppen eingeteilt werden können [Kra05]:

- Alarm- und Ereignisdienst (Alarm an Event Services) für Anforderung und Übertragung von Werten, Ereignismeldungen und Quittierungen.
- Datei-Zugriffs-Dienste (File Access Services) für Lesen und Schreiben von Dateien für Datensicherungszwecke.
- Objekt-Zugriffs-Dienste (Objekt Access Services) für Einrichtung, Pflege, Schreiben und Lesen der Dateien in einem BACnet Objekt.
- Device-Management-Dienst (Remote Device Management Services) für Fernbedienung und Ferneinrichtung von Devices im Gebäudeautomationsnetzwerk.
- Virtual-Terminal-Dienste (Virtual Terminal Services) für Einrichten des direkten zeichenorientierten Zugriffs auf die Software in einem Device.

Es wird prinzipiell zwischen unconfirmed und confirmed Diensten unterschieden. Details dazu sind im Kapitel 2.2.2 zu finden.

Property Identifier	Property Datentype	Conformance Code
Object_Identifier	BACnetObjectIdentifier	R
Object_Name	CharacterString	R
Object_Type	BACnetObjectType	R
Present_Value	BACnetBinaryPV	W
Description	CharacterString	O
Device_Type	CharacterString	O
Status_Flags	BACnetStatusFlags	R
Event_State	BACnetEventState	R
Reliability	BACnetReliability	O
Out_Of_Service	BOOLEAN	R
Polarity	BACnetPolarity	R
Inactive_Text	CharacterString	O
Active_Text	CharacterString	O
Change_Of_State_Time	BACnetDateTime	O
Change_Of_State_Count	Unsigned	O
Time_Of_State_Count_Reset	BACnetDateTime	O
Elapsed_Active_Time	Unsigned32	O
Time_Of_Active_Time_Reset	BACnetDateTime	O
Minimum_Off_Time	Unsigned32	O
Minimum_On_Time	Unsigned32	O
Priority_Array	BACnetPriorityArray	R
Relinquish_Default	BACnetBinaryPV	R
Time_Delay	Unsigned	O
Notification_Class	Unsigned	O
Feedback_Value	BACnetBinaryPV	O
Event_Enable	BACnetEventTransitionBits	O
Acked_Transitions	BACnetEventTransitionBits	O
Notify_Type	BACnetNotifyType	O
Event_Time_Stamps	BACnetARRAY[3] of BACnetTimeStamp	O
Profile_Name	CharacterString	O

Tabelle 1: Eigenschaften des Binary Output Object

2.2 Netzwerkkonzept vom BACnet

2.2.1 Schichtenmodell

Das BACnet-Schichtenmodell wurde in vier Schichten unterteilt, um den Kommunikationsoverhead und die Anforderungen an die Hardware und Software gering zu halten (Abbildung 3).

Die Dienste der OSI-Modell Schichten 4 bis 7, also Transport-, Sitzungs-, Darstellungs- und Anwendungsschicht, sind in der BACnet Anwendungsschicht zusammengefasst [Kra05]. Die **BACnet Anwendungsschicht (Application Layer, APDU**) sorgt somit für die Fehlerkontrolle und Korrektur der Nachricht. Sie übernimmt den Verbindungsaufbau und ist für die Synchronisation des Datenflusses verantwortlich (Aufgabe der Transportschicht). Sie überwacht und steuert den Datenaustausch, schützt

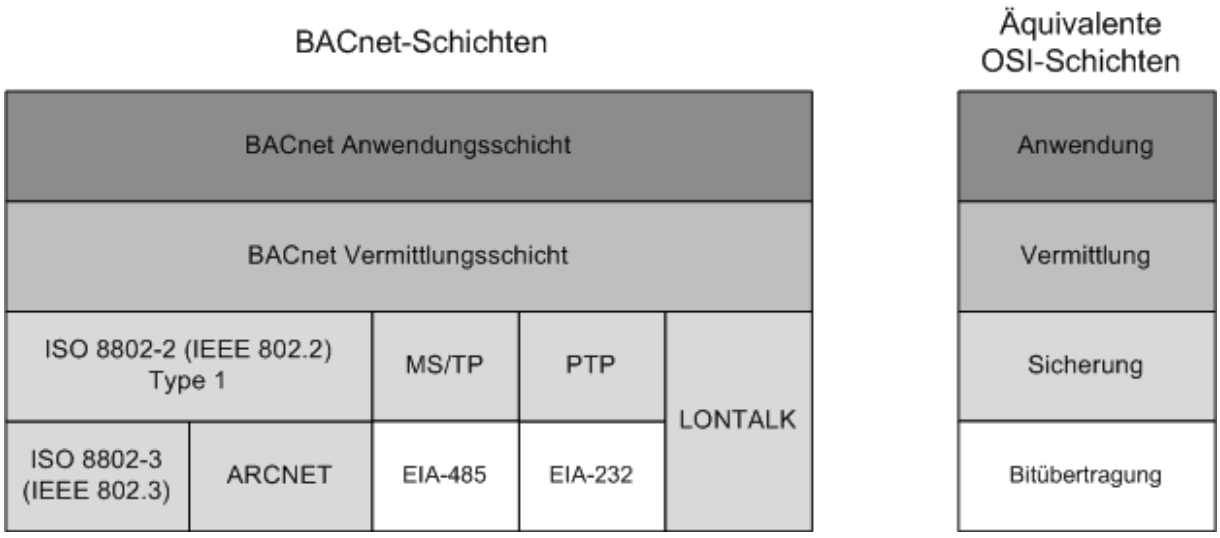

Abbildung 3: BACnet Schichtenmodell mit Referenz auf OSI-Modell

die Daten vor dem Zugriff Unberechtigter durch Verschlüsselung (Aufgabe der Sitzungsschicht). Letztlich verarbeitet sie die kommunikationsbezogenen Informationen der übertragenen Daten (Aufgabe der Darstellungs- und Anwendungsschicht).
Die **BACnet Vermittlungsschicht (Network Layer, NPDU)** sorgt dafür, den schnellsten und kostengünstigsten Weg der Datenpakete durch das Netzwerk zu finden (routing). Die wichtigsten Aufgaben der Vermittlungsschicht ist der Aufbau und die Aktualisierung der Routingtabelle sowie die Überwachung des Datenflusses.

Die **Sicherungsschicht (Data Link Layer, LPDU, MPDU**) gewährleistet eine fehlerfreie Übertragung (Fehlererkennung und Behandlung) und regelt den Zugriff auf das Übertragungsmedium. Die zu übermittelten Daten werden in kleine Blöcke unterteilt (genannt Frames). Die Synchronisation der Frames erfolgt mittels eigener Synchronisationszeichen. Die fehlerfreie Übertragung, also die Fehlererkennung und Fehlerbehandlung, erfolgt durch Hinzufügen von Folgenummern und Prüfsummen, wodurch das Ausbleiben bzw. das Verfälschen von Daten bemerkt wird.

Die **Bitübertragungsschicht (Physical Layer, PPDU)** ist die unterste Schicht und definiert die mechanischen, elektrischen, funktionalen und prozeduralen Parameter. Sie ist für das Aufrechterhalten der physikalischen Verbindung verantwortlich und definiert, wie die digitalen Bits, also logische Pegel, mit Hilfe des physikalischen Mediums übertragen werden.

Abbildung 4 zeigt, wie eine Nachricht vom Anwendungsprogramm die Schichten bzw. den Protokoll Stack durchläuft.

Es sei erwähnt, dass der BACnet Standard nur die Anwendungsschicht und die Netzwerkschicht definiert, Sicherungsschicht und Bitübertragungsschicht hingegen nicht. Um allerdings eine Kommunikation von verschiedenen Produkten unterschiedlicher Hersteller zu ermöglichen, wurden im BACnet-Standard fünf verschiedene Netzwerkoptionen festgehalten. Diese sind:

- Ethernet

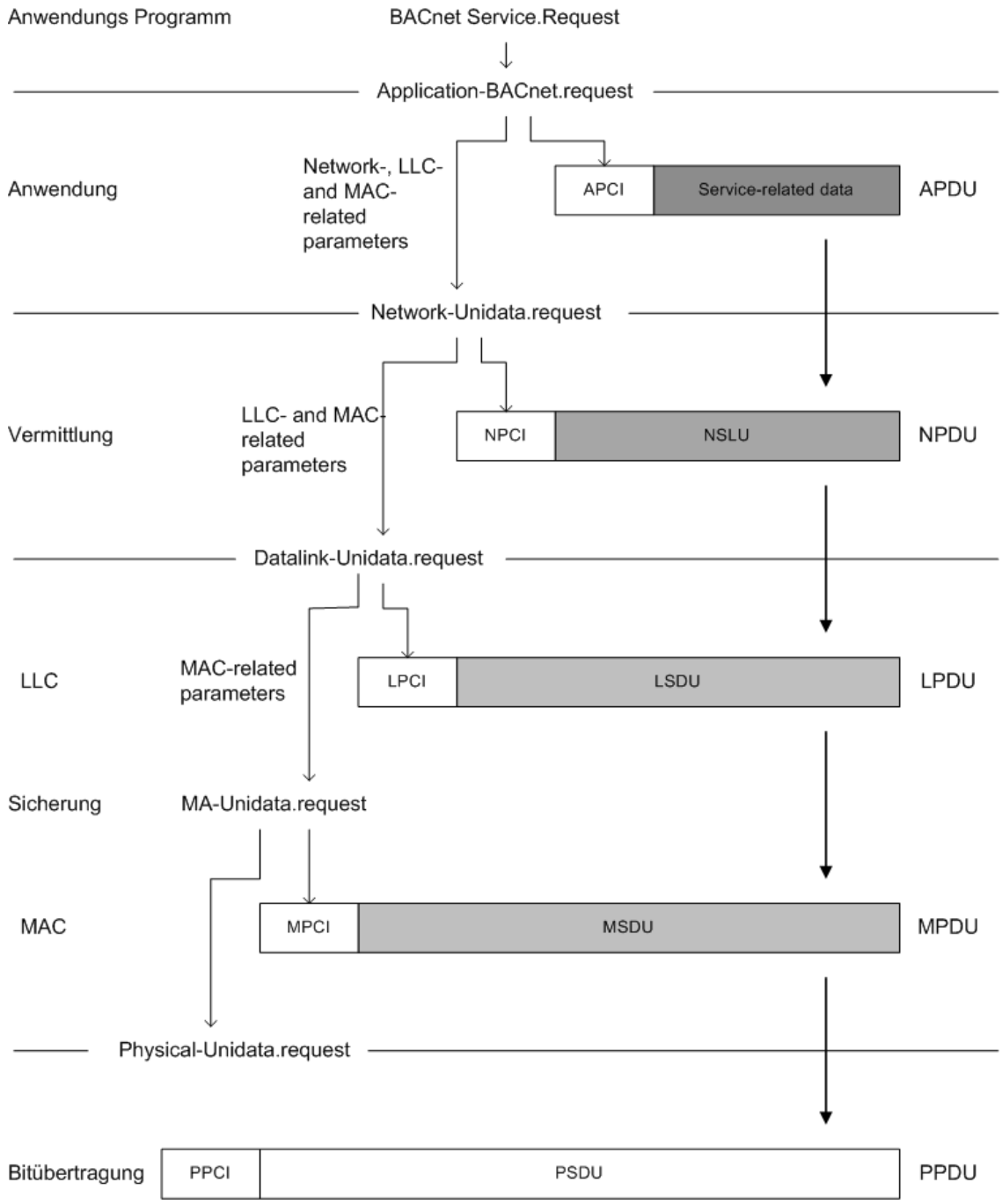

Abbildung 4: BACnet Protokoll Stack

- ARCNET (Attached Resources Computer Network)
- MS/TP (Master-Slave/Token-Passing) über RS-485
- PTP (Point-To-Point) über RS-232
- LonTalk ANSI/EIA709.1

Ethernet beschreibt eine kabelgebundene Netzwerktechnologie, die für lokale Datennetze (LANs,

Local Area Networks) entwickelt wurde. Ethernet verwendet eine Bus- oder Sterntopologie und regelt den Verkehr auf den Kommunikationsleitungen über das CSMA/CD (Carrier Sence Multiple Access with Collision Detection) Verfahren. Ethernet definiert den Kabeltyp, Stecker, logische Pegel der Bits für die Bitübertragungsschicht, Paketformat und Protokoll. Aus Sicht des OSI-Modells wird durch Ethernet sowohl die Bitübertragungsschicht als auch die Sicherungsschicht spezifiziert, beide sind in der IEEE-Norm 802.3 festgehalten.

ARCNET ist ein Token Bus-Netzwerk für LANs. Es basiert auf einer Bus- oder Sterntopologie und unterstützt eine Übertragungsgeschwindigkeit von 2,5 Mbit/s (durch Einsatz von spezieller Hardware sogar bis zu 10 Mbit/s). Einer der Vorteile von ARCNET ist die Möglichkeit, verschiedene Kabeltypen wie Twistedpair, Koaxkabel und Glasfaser in einem Netzwerk zu mischen [ARC02].

MS/TP (Master-Slave/Token-Passing) wurde von der ASHRAE entwickelt und stellt eine Lösung für Lowcost-Automation zur Verfügung. MS/TP kann im einfachem Master-Slave-Modus betrieben werden. Es existiert auch eine Variante mit mehreren gleichberechtigten Mastern, wobei immer ein Master ausgewählt wird, der die ihm zugeordneten Slaves abfragen (pollen) darf. Danach erfolgt die Tokenübergabe an den nächsten Master (Token-Passing-Verfahren).

PTP (Point-To-Point) ist ein Netzwerkprotokoll, mit dessen Hilfe ein Verbindungsaufbau über eine Wählleitung (Modemstrecke) zu einem Gerät oder einem anderen Netzwerk hergestellt werden kann. Da diese Verbindungen meistens kostenpflichtig sind, ist in dem BACnet Standard ein Mechanismus integriert, der es ermöglicht eine Verbindung automatisch wieder abzubrechen und später wieder aufbauen zu können, um nicht kontinuierlich verbunden zu sein. In der RFC 1661 wurde Point-To-Point standardisiert [Sim94].

LonTalk ist das Transportprotokoll für Lon-basierende Gebäudeautomationsnetzwerke. LonTalk wurde von der Echelon Corporation entwickelt. Als Übertragungsmedium stehen TP, Stromleitungen, Glasfaser (und theoretisch auch Funkfrequenzen) zu Verfügung.

2.2.2 Kommunikationsablauf

BACnet unterscheidet Dienste (Services), die nur aus einem Telegramm in eine Richtung bestehen (unconfirmed), und solche, die eine Bestätigung zum Absender (confirmed) zurücksenden.

Die Zeitsequenz Diagramme in Abbildung 5 und Abbildung 6 sind jeweils in drei Bereiche unterteilt. Der Bereich mit dem Namen BACnet Client repräsentiert die Einheit, die die Information anfordert bzw. Aktionen veranlaßt. BACnet Internetwork stellt das BACnet Netzwerk dar. Der Bereich BACnet Server ist jene Einheit, die Information bereitstellt. Die vertikalen Linien stellen die Schnittstelle zwischen BACnet Client und BACnet Server dar. Die gerichteten Pfeile spiegeln den Datenfluss wider.
Um eine Verbindung zu einem BACnet Server aufzubauen, sendet der BACnet Client einen Request über das BACnet Internetzwerk. Im Falle eines unconfirmed Services ein `UNCONF_SERV.request` und beim einem confirmed Service ein `CONF_SERV.request`. Das BACnet Internetzwerk sendet die PDUs `BACnet-Unconfirmed-Request-PDU` bzw. `BACnet-Confirmed-Request-PDU`

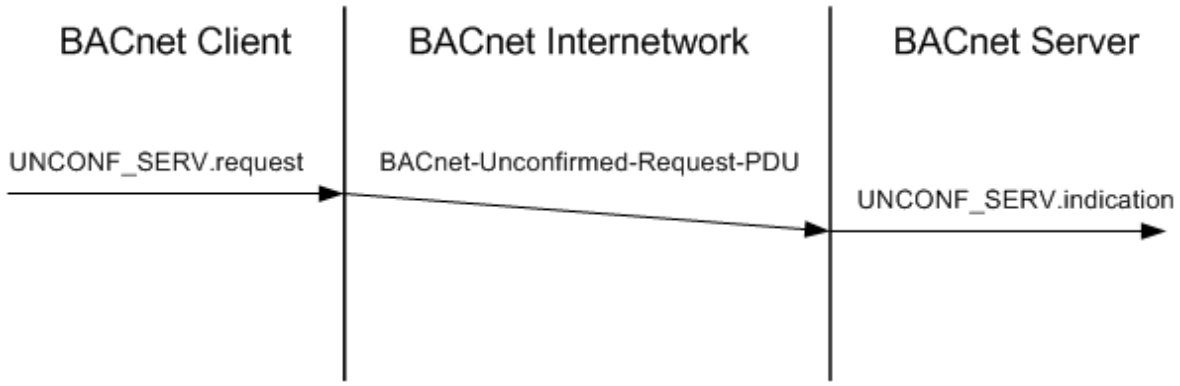

Abbildung 5: BACnet unconfirmed Service

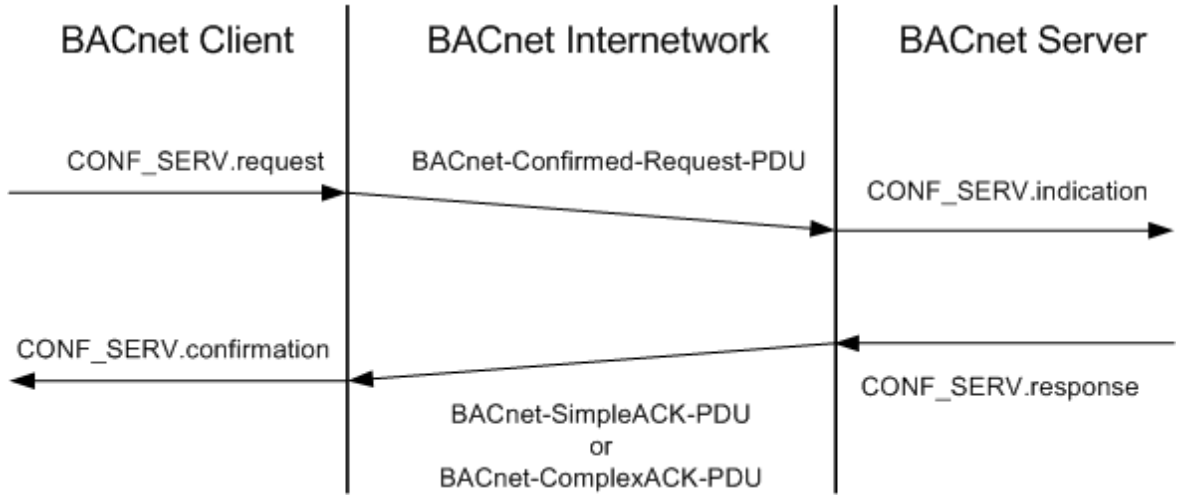

Abbildung 6: BACnet confirmed Service

an den BACnet Server, welcher die Empfangsmeldung `UNCONF_SERV.indication` bzw. `CONF_SERV.indication` erhält.

Bei einem confirmed Service sendet der Server eine `CONF_SERV.response` in das BACnet Internetzwerk, welches die Nachricht `BACnet-SimpleACK-PDU` oder `BACnet-ComplexACK-PDU` an den Client sendet. Der Client erhält in diesem Fall die Rückmeldung `CONF_SERV.confirmation` [Kra05].

Wie aus dem Zeitsequenz Diagramm in Abbildung 5 ersichtlich erwartet ein unconfirmed-Dienst keine Rückantwort (Quittierung). Ein BACnet-Device kann eine unconfirmed-Dienstanforderung auslösen, verarbeiten oder auch beides. Beim confirmed-Dienst wird, wie in Abbildung 6 ersichtlich, eine Rückantwort (Quittierung) erwartet. Ein BACnet-Device kann eine confirmed Dienstanforderung auslösen, verarbeiten oder beides.

Bei BACnet werden zwei Arten von Rückantworten (Quittierungen) unterschieden:

- *simple Acknowledgement* bestätigt ausschließlich, dass der Empfänger die Dienstanforderung empfangen hat.

- *complex Acknowledgement* bestätigt das Empfangen der Dienstanforderung und enthält außerdem noch Informationen, die vom Sender angefordert wurden (beispielsweise den Present Value eines Binary Output Objekts).

2.2.3 BACnet Ethernet Frame

Eine Möglichkeit, um BACnet LSDUs (Link Service Data Units) übertragen zu können, ist über Ethernet (ISO 8802-3). Ethernet übernimmt somit das Service der Sicherungsschicht und der Bitübertragungsschicht (Abbildung 3).
In Abbildung 7 ist der Aufbau eines BACnet Ethernet Frames abgebildet. Die ersten 6 Bytes identifizieren die MAC Adresse des BACnet Zielgerätes (DA = destination MAC layer address), das die Daten empfangen soll. Diese Adresse kann auch eine Broadcast-Adresse sein. Durch die darauf folgenden 6 Bytes wird die MAC Quelladresse des Senders identifiziert, damit das BACnet Zielgerät weiß, auf welche Adresse eine eventuelle Empfangsbestätigung zu senden ist (SA = source MAC layer address). Das 2 Byte LLC-Length Feld (logical link control) wird zur Angabe der Länge der zu übertragenen Daten verwendet.
Die beiden nächsten, jeweils 1 Byte großen Felder, DSAP (destination service access point) und SSAP (source service access point) werden dazu verwendet, um dem Zielgerät mitzuteilen, welchem Dienst die übertragenen Daten zuzuordnen sind bzw. von welchem Dienst diese Daten kommen. Mit dem Inhalt X'82' wird der BACnet Dienst identifiziert.
Das M Byte große Feld NPCI (network protocol control information) enthält die Kontrollinformation (u.a. BACnet Adresse, Priorität der Nachricht) der BACnet Netzwerkschicht. Die N Bytes große APDU (application layer protocol data unit) bildet die Dateneinheit der BACnet Anwendungsschicht und enthält die BACnet Nutzdaten [BAC04, Seite 87].

3 Implementierung

3.1 Übersicht Objekt-Implementierung

Um die Aufgabenstellung zu erfüllen, muss auf KNXcalibur ein BACnet Objekt implementiert werden, das auf ein entsprechendes KNX/EIB Objekt gemappt wird. Somit ist es möglich, von einem BACnet Sensor aus einen KNX/EIB Aktor zu steuern. Bei der KNXcalibur Implementierung wird als BACnet Sensor die VTS verwendet. Eine an KNX/EIB angeschlossene Lampe agiert als simpler Aktor. Um zwei Zustände wie das Ein- und Ausschalten eines Ausganges in BACnet zu verarbeiten, wird das BACnet Objekt `Binary Output` verwendet.

In Tabelle 1 findet man die für das Objekt `Binary Output` definierte Eigenschaft `Present Value`, der den Zustand des Ausgangs repräsentiert und auch verändert werden kann.

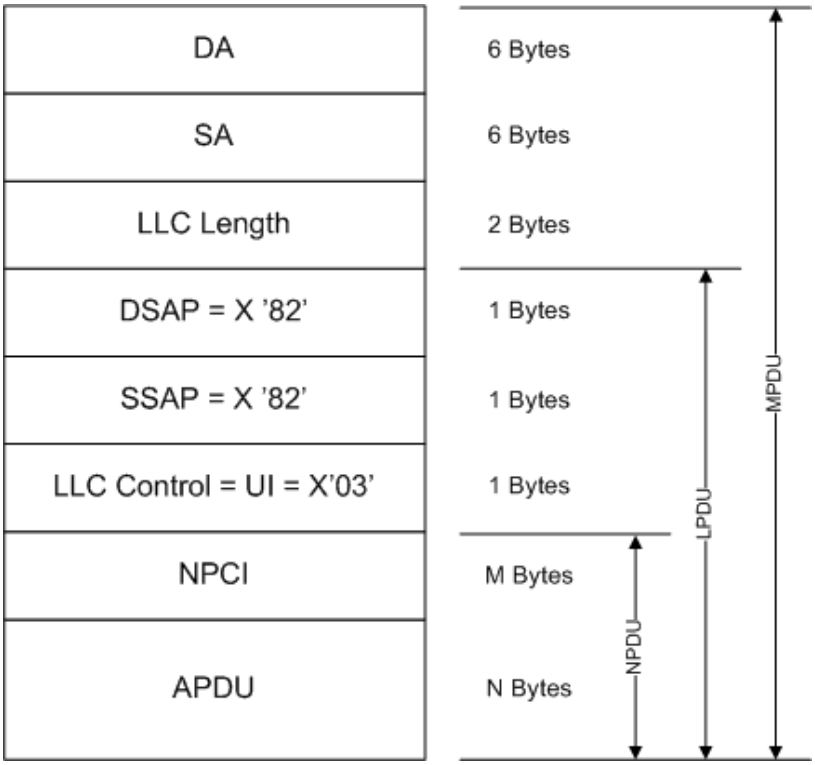

Abbildung 7: BACnet Ethernet Frame

3.2 Binär Ausgabe-Objekt (Binary Output)

Durch das Objekt Binary Output wird ein Objekttyp spezifiziert , dessen Eigenschaften extern sichtbare Charakteristika des Datenpunkts für einen binären Ausgang repräsentieren [Kra05]. Tabelle 1 zeigt alle Eigenschaften, die für das Objekt Binary Output möglich sind.
Um eine eindeutige Identifizierung jedes BACnet Objekts zu gewährleisten, sind die Properties `Objekt_Identifier`, `Objekt_Name` und `Objekt_Type` zwingend erforderlich.

Objekt_Identifier (Objektbezeichner): Um auf ein Objekt in einem BACnet Gerät zuzugreifen, benötigt man den `Objekt_Identifier`, der aus zwei Teilen besteht. Der erste Teil gibt den Objekttypen an (Binary Output) wieder. Der zweite Teil ist eine laufende Nummer für diesen Objekttyp im Gerät, genannt Instanz_Nummer (Objektnummer).
Der Objekt_Identifier hat den Datentyp `BACnetObjektIdentifier` und enthält einen im betrachteten BACnet-Gerät einmaligen numerischen Code der zu Identifikationszwecken dient. Damit können unterschiedliche BACnet Objekte vom selben Typ in einem einzigen BACnet Gerät implementiert werden.

Objekt_Name (Objektname): Diese Eigenschaft hat den Datentyp `CharacterString` (Zeichenfolge) und repräsentiert den Namen des Objekts. Um nicht technische (System-) Adressen benutzen zu müssen, legt der BACnet Standard fest, dass jedes Objekt über einen Objektnamen referenziert werden muss. Dieser Name muss im BACnet Netzwerk eindeutig sein. Der Name muss mindestens aus einem Zeichen bestehen. Es können alle druckbaren Zeichen verwendet werden.

Objekt_Type (Objekttyp): Hat den Datentyp `BACnetObjectType` und spezifiziert den Objekttyp.

Present_Value (Aktualwert): Diese Eigenschaft mit dem Datentyp `BACnetBinaryPV` stellt den logischen Status des `Binären Outputs` dar, der durch die zwei Zustände, Active und Inactive repräsentiert wird. Active bzw. Inactive spiegelt typischerweise das Einschalten bzw. Ausschalten einer physikalischen Einheit wider. Ob Active bzw. Inactive den logischen Zustand eingeschalten bzw. ausgeschalten repräsentiert, ist über die Eigenschaft **Polarity** mit dem Datentype `BACnetPolarity` zu definieren (siehe Tabelle 2).

Present_Value	Polarity	Physical State of Output	Physical State of Device
INACTIVE	NORMAL	OFF or INACTIVE	not running
ACTIVE	NORMAL	ON or ACTIVE	running
INACTIVE	REVERSE	ON or ACTIVE	not running
ACTIVE	REVERSE	OFF or INACTIVE	running

Tabelle 2: Beziehung zwischen den BACnet Eigenschaften Present_Value und Polarity

Description (Objektbeschreibung): Diese Eigenschaft hat den Datentype `CharacterString` (Zeichenfolge) und darf alle druckbaren Zeichen enthalten. Der Inhalt kann selbst gewählt werden und enthält üblicherweise eine Beschreibung des Objekts.

3.3 Objekt Access Service

Um den ”Present Value”zu ändern, also den Ausgang ein bzw. auszuschalten, benötigt man ein Objekt Access Service. Dieser Dienst zum Schreiben heißt ”WriteProperty” (Detailbeschreibung siehe [BAC04, Seite 299]). Mit dem Dienst ”ReadProperty”(Detailbeschreibung siehe [BAC04, Seite 285]) kann der aktuelle Wert gelesen werden. Der schematische Ablauf von WriteProperty und ReadProperty ist in Abbildung 8 dargestellt.

Die VTS agiert als BACnet Client. Sie ändert den Present Value, indem ein WriteProperty Service an KNXcalibur mit dem gewünschten Zustand gesendet wird.
KNXcalibur verarbeitet die BACnet Nachricht und generiert eine entsprechende KNX/EIB-Nachricht (`A_GroupWrite`), die eine Lampe, die an KNX/EIB angeschlossen ist, schaltet.
Sendet die VTS ein ReadProperty Service, verarbeitet KNXcalibur die BACnet Nachricht und sendet eine entsprechende KNX/EIB-Nachricht, um den aktuellen Wert des KNX/EIB Aktors auszulesen. Nach Erhalt des Aktorwertes wird eine entsprechende BACnet ReadProperty Response Nachricht generiert und an den BACnet Client gesendet.

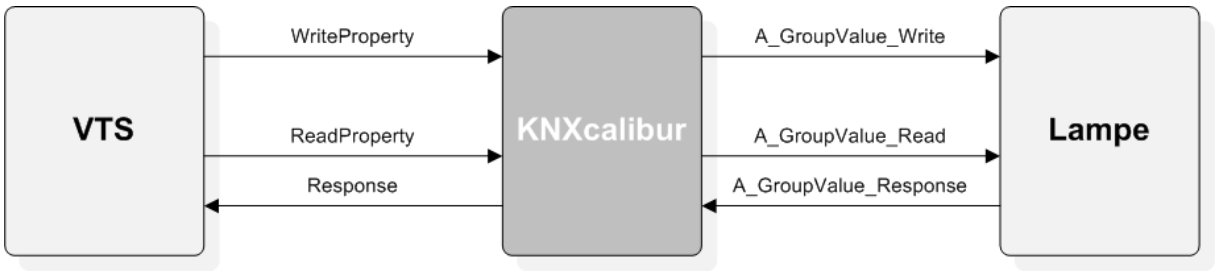

Abbildung 8: Objekt Access Service

3.4 Remote Device Management Service

Um dynamisches ”Device binding” zu unterstützen, wurden die ”Remote Device Management Services” ”Who-Is” und ”I-Am” und das ”Who-Has” und ”I-Have” implementiert. Hierbei handelt es sich um Broadcast-Dienste.

Who-Is and I-Am Services
Das Who-Is Service wird vom BACnet Usern benutzt, um den ”Device Objekt Identifier” und die ”Netzwerk Adresse“ auszulesen (siehe auch [BAC04, Seite 218]). Der Kommunikationsablauf ist in Abbildung 9 dargestellt. Der genaue Kommunikationsablauf zwischen VTS und KNXcalibur ist in Abschnitt 3.8 auf Seite 25 erörtert.

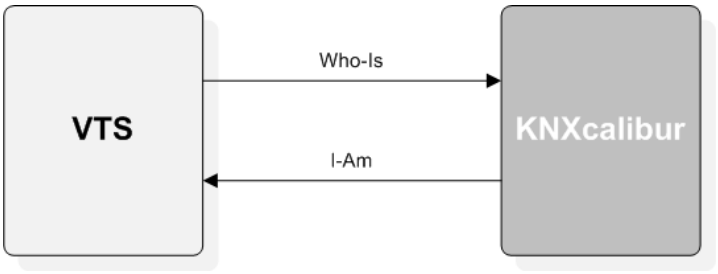

Abbildung 9: Who-Is und I-Am Service

Who-Has und I-Have Service
Das Who-Has Service erlaubt einem BACnet User den ”Device Objekt Identifier” und die ”Netzwerk Adresse” eines anderen Geräts im BACnet Netzwerk zu identifizieren, wenn man dessen ”Objekt Namen” oder ”Object Identifier” kennt (siehe auch [BAC04, Seite 216]). Der Kommunikationsablauf ist in Abbildung 10 dargestellt.

3.5 Ethernet Implementierung

Bei der Implementierung von BACnet auf KNXcalibur wurde zur Datenkommunikation auf Schicht 1 und 2 die Ethernettechnologie (IEEE 802.3) verwendet. Gründe dafür waren, dass Ethernet nicht nur den Vorzug bietet, dass es für BACnet die schnellste Netzwerktopologie darstellt, sondern derzeit auch

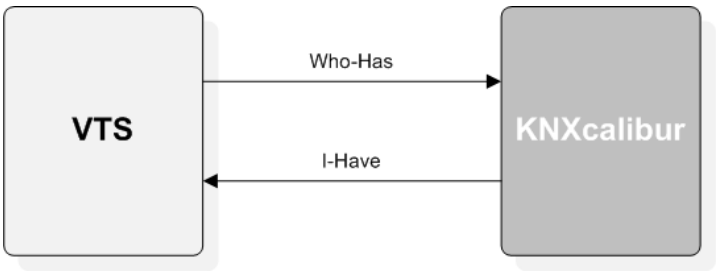

Abbildung 10: Who-Has und I-Have Service

die meist verbreitete LAN Technologie in der Industrie ist. Die Anordnung von Ethernet im BACnet Schichtenmodell ist in Abbildung 11 dargestellt.

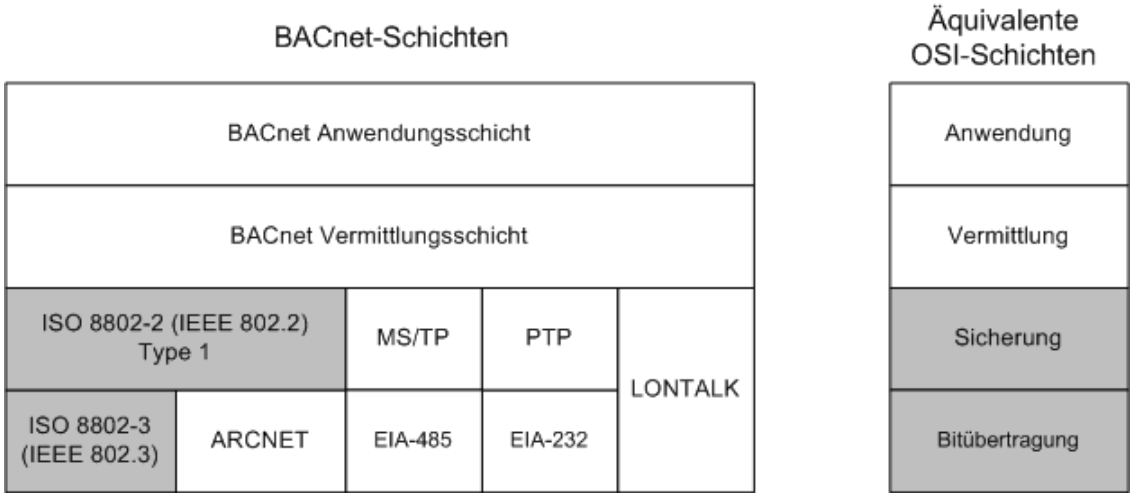

Abbildung 11: Ethernet Implementierung

3.6 Beschreibung des Open Source Stacks

3.6.1 Ordnerstruktur

Die Ordnerstruktur der KNXcalibur BACnet Implementierung wurde zwecks Kompatibilität an die vorhandene Sourceforge Struktur angepasst.

/bacnet-stack/
... in diesem Ordner befinden sich alle notwendigen Dateien für den BACnet Stack.

/bacnet-stack/demo/handler/
... die Handler findet man in diesem Ordner. Sie sind mit h_* gekennzeichnet. Durch diese Handler ist das Verhalten des BACnet Gerätes definiert. Diese Handler Dateien müssen für die jeweilige Anwendung angepasst werden. In unserem Fall zum Beispiel die Datei h_wp.c (handler write property).

/bacnet-stack/demo/object/

... hier findet man die bereits implementierten BACnet Objekte.

/bacnet-stack/ports/MB90330/
... alle für KNXcalibur abhängigen Dateien sind hier enthalten.

3.6.2 Wichtige Dateien

- config.h - In dieser Datei sind die Einstellungen für den Stack vorzunehmen. Zum Beispiel Deklaration der physikalischen Schicht.
- datalink.h - Hier ist die Sicherungsschicht enthalten. Ebenso findet man hier auch die Funktionsdefinitionen dieser Schicht.
- bacdef.h - In dieser Datei sind Makros für den Stack definiert, die eventuell bei einer Implementierung auf eine andere Hardware angepasst werden müssen. Hier findet man auch Typ Definitionen des BACnet Stacks.
- bacenum.h - enthält Makro Definitionen und Typdefinitionen für den Stack.

3.6.3 BACnet Stack Dateien

- abort.c - BACnet Abort service encode/decode
- bacerror.c - BACnet Error service encode/decode
- reject.c - BACnet Reject service encode/decode
- rp.c - BACnet ReadProperty service encode/decode
- arf.c - AtomicReadFile service encode/decode
- awf.c - AtomicWriteFile service encode/decode
- rpm.c - ReadPropertyMultiple service encode/decode
- iam.c - I-Am service encode/decode
- whois.c - WhoIs service encode/decode
- wp.c - WriteProperty service encode/decode
- dcc.c - DeviceCommunicationControl service encode/decode
- ihave.c - I-Have service encode/decode
- rd.c - ReinitializedDevice service encode/decode
- timesync.c - TimeSynchronization service encode/decode

- whohas.c - WhoHas service encode/decode
- address.c - zuständig für address binding.
- tsm.c - Transaction State Machine, regelt das Senden von Nachrichten

3.6.4 Beispiele für BACnet Objekte

- demo/object/ai.c - analog input object demo
- demo/object/ao.c - analog output object demo
- demo/object/bacfile.c - file object demo
- demo/object/device.c - device object demo
- demo/object/bi.c - binary input object demo
- demo/object/bo.c - binary output object demo
- demo/object/lsp.c - life safety point demo

3.6.5 Dateien für die Sicherungsschicht

- apdu.c - Zuteilung der Services zu den Handlern für Eigenschaften (Properties)
- bacdcode.c - BACnet encoding und decoding
- bacstr.c - BACnet string encoding und decoding
- bigend.c - überprüft ob CPU auf bigendian basiert
- datalink.c - API für Sicherungsschicht
- npdu.c - Zuteilung der Nachrichten an den APDU Handler

3.6.6 Von der Implementierung der Sicherungsschicht abhängige Dateien

- bip.c - BACnet/IP Funktion
- dllmstp.c - MS/TP Sicherungsschicht
- mstp.c - MS/TP State Machine
- crc.c - Checksum für MS/TP
- ringbuf.c - MS/TP Ring Puffer
- arcnet.h - ARCNET Sicherungsschicht API
- ethernet.h - BACnet Ethernet Sicherungsschicht API

3.6.7 main.c

Das Hauptprogramm ist folgendermaßen strukturiert:

1. Initialisieren der Schnittstelle zur Datenkommunikation

In der Datei *ethernet.h* findet man den notwendigen Prototypen, der im File *ethernet.c* implementiert werden muss.

*bool ethernet_init (char * interface_name);*

Parameter:

- **interface_name:** Name des zu initialisierenden Ethernet Geräts z.B.: eth0

Returnwerte:

- 1, wenn Ethernet init erfolgreich war und jetzt verfügbar ist
- 0, wenn Ethernet init nicht erfolgreich war
- negative, im Fehlerfall

2. MAC Adresse des Objekts festlegen.

Mit dem `BACnetObjektIdentifier` und der MAC Adresse des Geräts ist eine eindeutige Zuordnung jedes Objekts in einem Netzwerk gewährleistet. Dadurch ist eine Kommunikation zwischen den einzelnen Objekten bzw. zwischen Objekten und VTS möglich. Den notwendigen Prototypen findet man in der Datei *ethernet.h*, der in *ethernet.c* implementiert werden muss.

*void ethernet_set_my_address(BACNET_ADDRESS * my_address);*

Parameter:

- **my_address:** Lokale MAC-Adresse

3. Instanz Nummer und Name des Objektes setzen.

In der Datei *device.h* bzw. *device.c* findet man alle notwendigen Properties, die für ein Objekt eingestellt werden können. Zwei, die unbedingt gesetzt werden sollten, sind die Instanz Nummer und der Name des Geräts (siehe Punkt 3.2 auf Seite 16).

Jedem Objekt ist eine Instanz Nummer zugeordnet:
bool Device_Set_Object_Instance_Number(uint32_t object_id);

Parameter:

- **object_id:** Objekt Instanz Nummer

Setzen des repräsentativen Objekt Namen, der über das BACnet Netzwerk eindeutig sein muss, erfolgt mittels (siehe auch 3.2 auf Seite 16):

*bool Device_Set_Object_Name(const char *name, size_t length);*

Parameter:

- **name:** willkürlicher Name des Geräts
- **length:** Zeichenanzahl im Parameter name

4. Service Handler aufrufen.

In der Funktion *Init_Service_Handler* werden alle für das Objekt erforderlichen Services registriert. Je nach dem, ob es sich um ein unbestätigtes (unconfirmed) Service oder ein bestätigtes (confirmed) Service handelt, muss der passende Handler aufgerufen werden. Mit *apdu_set_unconfirmed_handler* wird ein unbestätigtes Service registriert. Mit *apdu_set_confirmed_handler* wir ein bestätigtes Service registriert.

static void Init_Service_Handlers(void);

Handler die aufgerufen werden:

- Um dynamisches Device binding zu unterstützen, muss das `Who-Is` Service und das `Who-Has` Service registriert werden:

  apdu_set_unconfirmed_handler(SERVICE_UNCONFIRMED_WHO_IS, handler_who_is);
  apdu_set_unconfirmed_handler(SERVICE_UNCONFIRMED_WHO_HAS, handler_who_has);

- Um das Service Read Property zu unterstützen, muss der zugehörige Handler aufgerufen werden:

  apdu_set_confirmed_handler(SERVICE_CONFIRMED_READ_PROPERTY,
  handler_read_property);

- Um das Service Write Property zu unterstützen, muss der zugehörige Handler aufgerufen werden:

  apdu_set_confirmed_handler(SERVICE_CONFIRMED_WRITE_PROPERTY,
  handler_write_property);

- Um ankommende Fehlermeldungen verarbeiten zu können, muss MyErrorHandler aufgerufen werden:

  apdu_set_error_handler(SERVICE_CONFIRMED_WRITE_PROPERTY, MyErrorHandler);

Dabei ist der Parameter, der den Handlern an erster Stelle übergeben wird, ein Makro, das zur internen Identifikation des Services im BACnet Stack dient. Der zweite Parameter ist der Name der Funktion, definiert in der zugehörigen Datei h_* des Services, das die Verarbeitung der BACnet Nachrichten übernimmt.
Für das Service `WriteProperty` lautet die zugehörige Datei `h_wp.c`, zu finden im Ordner */bacnet-stack/demo/handler/*. In dieser Datei befindet sich die Funktion, die für die Verarbeitung der `WriteProperty` Nachricht verantwortlich ist. Hier wird der `Present Value` des BACnet Objektes auf ein KNX Objekt gemappt, und somit eine Nachricht in das KNX/EIB Netzwerk gesendet.

5. In einer Endlosschleife werden die BACnet Frames empfangen.

In der Datei *ethernet.h* ist der Prototyp für die Funktion *ethernet_receive* definiert, die wiederum in der Datei *ethernet.c* zu implementieren ist. Die Empfangsfunktion ist blocking programmiert und gibt die Länge der empfangenen PDU Daten zurück.

*uint16_t ethernet_receive (BACNET_ADDRESS * src, uint8_t * pdu, uint16_t max_pdu, unsigned timeout);*

Parameter:

- **src:** Source MAC-Adresse
- **pdu:** PDU Daten
- **max_pdu:** maximal verfügbarer Platz in der PDU
- **timeout:** Millisekunden, die auf ein Datenpaket gewartet werden

Returnwerte:

- Retourniert die Anzahl an Bytes in der PDU, oder 0 im Fehlerfall

6. Wurden Daten empfangen, werden sie dem "npdu_handler " übergeben.

Wenn die Länge der PDU Daten (Rückgabeparameter der Funktion ethernet_receive) größer als 1 ist, wird der *npdu_handler* aufgerufen. Dieser ist für die Zerlegung und Decodierung des NPDU Frames verantwortlich. Abhängig von den darin enthaltenen Daten wird dann automatisch der Service Handler ausgeführt, sofern dieses Service zuvor in der Funktion *Init_Service_Handlers* registriert wurde.

*void npdu_handler(BACNET_ADDRESS * src, uint8_t * pdu, uint16_t pdu_len);*

Parameter:

- **src:** Source MAC-Adresse
- **pdu:** PDU Daten
- **pdu_len:** PDU Länge

3.7 Entwicklungsumgebung

Für die Software-Entwicklung wurde die für den Microcontroller Fujitsu MB90F33A bereitgestellte Entwicklungsumgebung "F2MC-16 Family SOFTUNE Workbench V30L32" verwendet. Zum Test der auf dem Microcontroller implementierten BACnet Funktionalitäten kam die Visual Test Shell Version 3.4.9 (VTS) zum Einsatz.

3.8 Kommunikationsablauf auf KNXcalibur

Der Kommunikationsablauf zwischen dem BACnet Client und dem BACnet Server im Fall der KNXcalibur Implementierung ist in Abbildung 12 auf Seite 31 dargestellt.

Um dynamisches "address binding" zu unterstützen wird erst das unconfirmed Service Who-Is an KNXcalibur gesendet (`unconfirmed_WHO_IS`), worauf dieser mit dem uncofirmed I-Am (`unconfirmed_I_AM`) antwortet. Als nächstes Remote Device Management Service wird das unconfirmed Service Who-Has (`unconfirmed_WHO_HAS`) gesendet, worauf der Server mit unconfirmed I-Have `unconfirmed_I_HAVE` antwortet (siehe auch Punkt 3.4 auf Seite 3.4).

Soll nun der KNX/EIB Aktor eingeschaltet werden, so wird vom Client ein confirmed Service Write Property (`confirmed_WRITE_PROPERTY`) mit ein `Present_Value` "1" an KNXcalibur gesendet. Dieser quittiert die Nachricht mit einem simple Acknowledge (`simpleACK_WRITE_PROPERTY`), um das korrekte Empfangen der `confirmed_WRITE_PROPERTY` Nachricht zu bestätigen. KNXcalibur generiert die KNX-Nachricht `A_GroupWrite` und sendet die Nachricht zum Einschalten der Lampe ins KNX/EIB Netzwerk.

Soll nur der Status der Lampe abgefragt werden, so sendet der Client ein confirmed Service Read Property (`confirmed_READ_PROPERTY`). KNXcalibur generiert die KNX-Nachricht `A_GroupValue_Read` und sendet diese ins KNX/EIB Netzwerk. Der Empfang der KNX-Nachricht wird vom KNX-Objekt mit einer `A_GroupValue_Response` Nachricht an KNXcalibur bestätigt. Worauf KNXcalibur ein Read Property complex Acknowledge (`complexACK_READ_PROPERTY`) ins BACnet Netzwerk absendet. Ein complex Acknowledge gibt nicht nur Auskunft, dass eine Nachricht eingegangen ist, sondern enthält auch Rückgabewerte, in diesem Fall den aktuellen Status der Lampe "1" (Lampe ein) oder "0" (Lampe aus).

3.9 Write Property Frame

Im Abschnitt 2.2.3 auf Seite 15 wurde der prinzipielle Aufbau eines Frames besprochen. Abbildung 13 zeigt ein "confirmed Write Property Frame", so wie es von der VTS zu dem am KNXcalibur implementierten `Binary Output Objekt` gesendet wird.

3.10 Benutzerhandbuch

3.10.1 Testwerkzeug Visual Test Shell

Die VTS wurde von der BACnet Manufacturers Association (BMA) entwickelt, und unterliegt der ”Public Domain Software”. Sie ist auf Sourceforge.net (http://sourceforge.net/) frei verfügbar.
Für die Grundkonfiguration der VTS sei auf den Installations-Guide verwiesen (VTS3Guide.doc).

Nachdem die Ethernetschnittstelle konfiguriert wurde, können Nachrichten über die eingerichtete Schnittstelle empfangen und gesendet werden.
Als erstes muss dazu die Schnittstelle für die BACnet Kommunikation im Programmmenü ausgewählt werden. (Der Name *eth*, der für Ethernet steht, kann bei der VTS Konfiguration, wie im VTS Installations Guide beschrieben, frei gewählt werden.):
send → eth(Ethernet)

Nachdem die Schnittstelle festgelegt wurde, ist über das Menü ein Service auszuwählen (siehe Abbildung 14):
für ”Remote Device Management” Services:
send →Remote Device Management → Who Is
send →Remote Device Management → Who Has

für ”Objekt Access” Services:
send → Objekt Access → Write Property
send → Objekt Access → Read Property

3.10.2 Applikation auf KNXcalibur laden

Die Entwicklungsumgebung Softtune Workbench ist eine IDE für das Betriebssystem Windows und stellt einen C Compiler und einen Simulator zur Verfügung. Die Programmierung kann sowohl in C als auch in Assembler erfolgen. Bei der BACnet Implementierung auf KNXcalibur wurde ausschließlich C Code generiert.

Mit dem von Fujitsu mitgelieferten ”Flash Progammer” wird das Programm auf den MB90F33A geladen. Im Flash Programmer sind noch die Hardware Einstellungen für Microcontroller auf KNXcalibur zu treffen. Der verwendete Microcontroller ist ein Fujitsu MB90F33A und arbeitet mit einer externen Frequenz von 6 MHz (siehe Abbildung 15).

3.10.3 Applikation auf KNXcalibur ausführen

WHO IS

Um das Remote Device Management Services Who-Is vom VTS Client abzusetzen, muss zuerst so wie in Abschnitt 3.10.1 auf Seite 26 das Service ausgewählt werden. Anschließend öffnet sich ein Konfigurationsfenster (Abbildung 16). Um dynamisches "Address binding" zu unterstützen, muss die MAC Adresse "FF-FF-FF-FF-FF-FF" eingetragen werden. Nach Drücken des Send Buttons wird die Broadcast Nachricht in das BACnet Netzwerk gesendet.

Als Antwort sollte das BACnet Geräte eine I-Am Antwort generieren.

WHO HAS

Das Remote Device Management Services Who-Has wird so wie in Abschnitt 3.10.1 auf Seite 26 beschrieben im VTS Client ausgewählt. Um eine Who has Broadcast Nachricht in das BACnet Netzwerk zu senden, muss im Who Has Konfigurationsfenser die MAC Adresse "FF-FF-FF-FF-FF-FF" in das Feld Destination (Registerkarte Ethernet) eingegeben werden (Abbildung 17).

Um ein bestimmtes Objekt im BACnet Netzwerk zu suchen, hat man zwei Möglichkeiten. Eine Möglichkeit ist, man gibt den Namen des BACnet Objekts in dem Feld Objekt Name an (Abbildung 18).

Bei der BACnet Implementierung auf KNXcalibur wurde dem Objekt der Name "KNX" zugewiesen. Deshalb kann man hier im Feld Objekt Name (Registerkarte Who-Has) KNX eingeben.

Ist der Name des Objekts nicht bekannt, so muss auf den Button ID... im Feld Objekt ID gedrückt werden, wodurch sich das Fenster in Abbildung 19 öffnet.

Hier muss nun der Objekt Type und die Instance angegeben werden. In der vorliegenden KNXcalibur Implementierung handelt es sich um einen Binary Output mit der Instance Nummer 4.
Nach Betätigen des Button Send kann ein Antwort I-Have von allen BACnet Objekten, auf die die Einstellungen zutreffen, erwartet werden.

READ PROPERTY

Damit die VTS das Objekt Access Services Read Property senden kann, muss wie in Abschnitt 3.10.1 auf Seite 26 erklärt, das Service ausgewählt werden. Anschließend öffnet sich das in Abbildung 20 dargestellte Konfigurationsfenster.

Zu Beginn muss im Feld Destination (Registerkarte Ethernet) die Zieladresse eingegeben werden. Diese Adresse sollte man als Antwort auf die Who-Has Anfrage mit I-Have erhalten haben.

In der Registerkarte ReadProperty müssen die Einstellungen für die Objekt ID und die Property vorgenommen werden (Abbildung 22), die zuvor mit dem I-Have Service ermittelt wurden.

Nach Betätigen des Buttons ID... kommt man zur Ansicht 21.

Hier muss nun der Objekt Type und die Instance angegeben werden. In diesem Beispiel handelt es sich um einen Binary-Output mit der Instance Nummer 4.

Unter dem Punkt Property muss noch angegeben werden, welche Information vom Objekt ausgelesen werden soll. In der vorliegenden Implementierung kann erfragt werden ob die Lampe zurzeit ein - bzw. ausgeschaltet ist. Es muss also die Property Present_Value abgefragt werden (Abbildung 22).

Nach Betätigen des Buttons Send erhält man als Antwort ein komplexes Acknowledgement, also ein Acknowledgement complexACK_READ_PROPERTY, das den aktuellen Present_Value enthält.

WRITE PROPERTY

Um das Objekt Access Service Write Property auszuführen, muss es wie in Abschnitt 3.10.1 auf Seite 26 erklärt, gewählt werden.

Um auf das implementierte Binary Output Objekt zuzugreifen, muss die Adresse des BACnet Servers eingegeben werden, auf dem sich das Objekt befindet. Dazu wird, wie in Abbildung 23 dargestellt, im Feld Destination (Registerkarte Ethernet) die MAC Adresse des BACnet Servers eingegeben. Diese Adresse des BACnet Servers wurde durch die vorherige Who-Has Anfrage mit der Antwort I-Have ermittelt.

Weiters muss das Objekt identifiziert werden. Dazu muss in der Registerkarte WriteProperty im Feld Property angegeben werden, welcher Wert geändert werden soll. Im vorliegendem Fall soll die Lampe eingeschalten werden, das heißt, der Present_Value ist zu ändern.

Nach Betätigen des Button ID... im Feld Objekt ID in der Registerkarte WriteProperty kommt man zu folgender in Abbildung 25 dargestellten Ansicht.

Hier muss nun der Objekt Typ und die Instance angegeben werden. In der vorliegenden Implementierung handelt es sich um einen Binary Output mit der Instance Nummer 4. Auf der Registerkarte WriteProperty wird durch Klicken auf den Button Any unter Value ein neues Fenster geöffnet, wo der Datentyp und der zu übertragende Wert anzugeben ist.

Zuerst muss durch Drücken des Buttons Add ein Daten Typ hinzugefügt werden. Anschließend müssen Type und Value gesetzt werden.

Der Datentyp für Binary Present Value ist folgendermaßen definiert: *BACnetBinaryPV ::= ENUMERATED {inactive (0),active (1)}*

Deshalb muss bei Type Enumerated angegeben werden. Damit die Lampe eingeschaltet wird, muss der Wert (Value) auf active gesetzt werden. Nachdem der Button Send gedrückt wurde, antwortet KNXcalibur mit einem simple Acknowledgement Write Property (simpleACK_WRITE_PROPERTY).

4 Prototypimplementierung

Teile dieser Arbeit wurden im Rahmen einer Prototypimplementierung umgesetzt und getestet. Diese Prototypimplementierung wurde auf der Konnex Scientific Conference 2006 in Wien veröffentlicht. Für nähere Informationen sei auf [PKN06] verwiesen.

5 Aufgetretene Probleme

1. Fehlende Dokumentation des Sourceforge BACnet Stacks.
2. Bei der VTS können für ein Objekt mehr Eigenschaften eingestellt werden, als dem Objekt laut BACnet Standard überhaupt zur Verfügung stehen. So kann beispielsweise bei einem Binary Output Objekt dem Property Present_Value, der laut BACnet Standard den Datentyp BACnetBinaryPV hat, ein beliebiger Datentyp eingestellt werden.
3. Im BACnet Stack von Sourceforge wird nicht durchgängig ANSI C verwendet, deshalb mussten einige Routinen selbst implementiert werden. Die sind beispielsweise die Funktion write(), die Daten in einen Puffer schreibt, und die Funktion strcasecmp() zum Vergleichen zweier Zeichenketten.

Literatur

[ARC02] ARCNET Trade Association. http://www.arcnet.com/, 2002.

[BAC04] BACnet – A Data Communication Protocol for Building Automation and Control Networks. ANSI/ASHRAE 135, 2004.

[KNX04] KNX Specification. KNX Association, Diegem, Belgium, Version 1.1, 2004.

[KNX07] KNX Association. http://www.knx.org/, 2007.

[Kra05] Hans. R. Kranz. *BACnet Gebäude-Automation 1.4*. Promotor Verlag, 1st edition, 2005. ISBN 3-922420-02-8.

[PKN06] Fritz Praus, Wolfgang Kastner, and Georg Neugschwandtner. A versatile networked embedded platform for KNX/EIB. In *Proc. Konnex Scientific Conference*, November 2006.

[Pra05] Fritz Praus. A versatile networked embedded platform for KNX/EIB. Diplomarbeit, Technische Universität Wien, Wien, 2005.

[Sim94] W. Simpson. RFC 1661 The Point-to-Point Protocol (PPP). http://tools.ietf.org/html/rfc1661, 1994.

[Wei07] Jürgen Weidinger. Doxygen Output of BACnet implementation of KNXcalibur. http://www.auto.tuwien.ac.at/~jweidinger/, 2007.

6 Verwendete Abkürzungen

ANSI	American National Standards Institute
APDU	application layer protocol data unit
ASHRAE	American Society of Heating, Refrigerating and Air-Conditioning Engineers
BACnet	Building Automation and Control NETwork
BIBB	BACnet Interoperability Building Block
BMA	BACnet Manufacturers Association
DA	local destination MAC layer address
NPDU	network layer protocol data unit
DSAP	LLC destination service access point (X'82' for BACnet)
ISO	International Organisation for Standardization
KNXcalibur	A versatile networked embedded platform for KNX/EIB
LLC	logical link control (ISO 8802-2)
LPDU	link protocol data unit
MPDU	MAC layer protocol data unit
NPCI	network protocol control information
SA	local network source MAC layer address
SSAP	LLC source service access point (X'82' for BACnet)
TPUART	Twisted Pair Universal Asynchronous Receiver Transmitter
VTS	Visual Test Shell Version 3.4.9

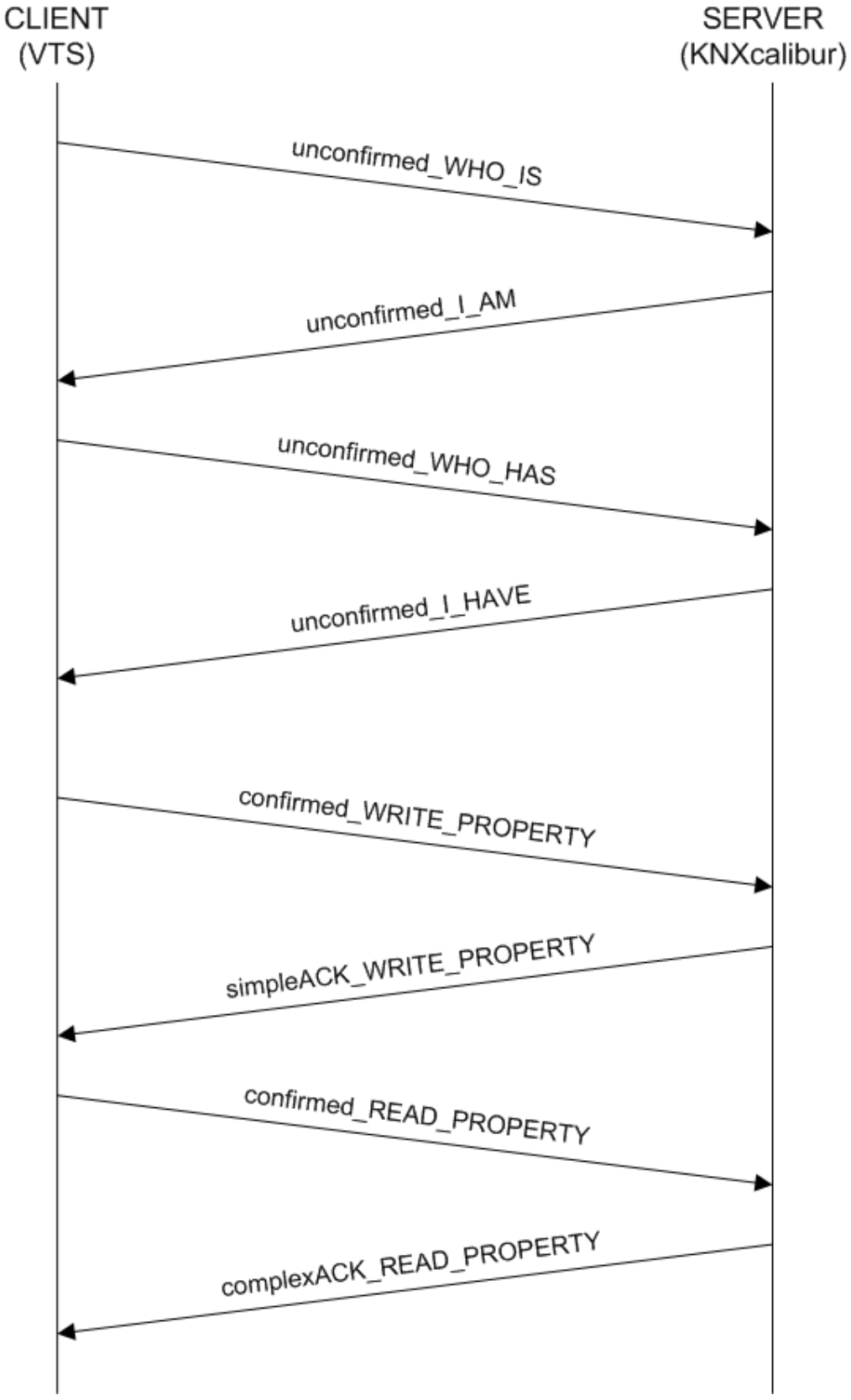

Abbildung 12: KNXcalibur Kommunikationsablauf

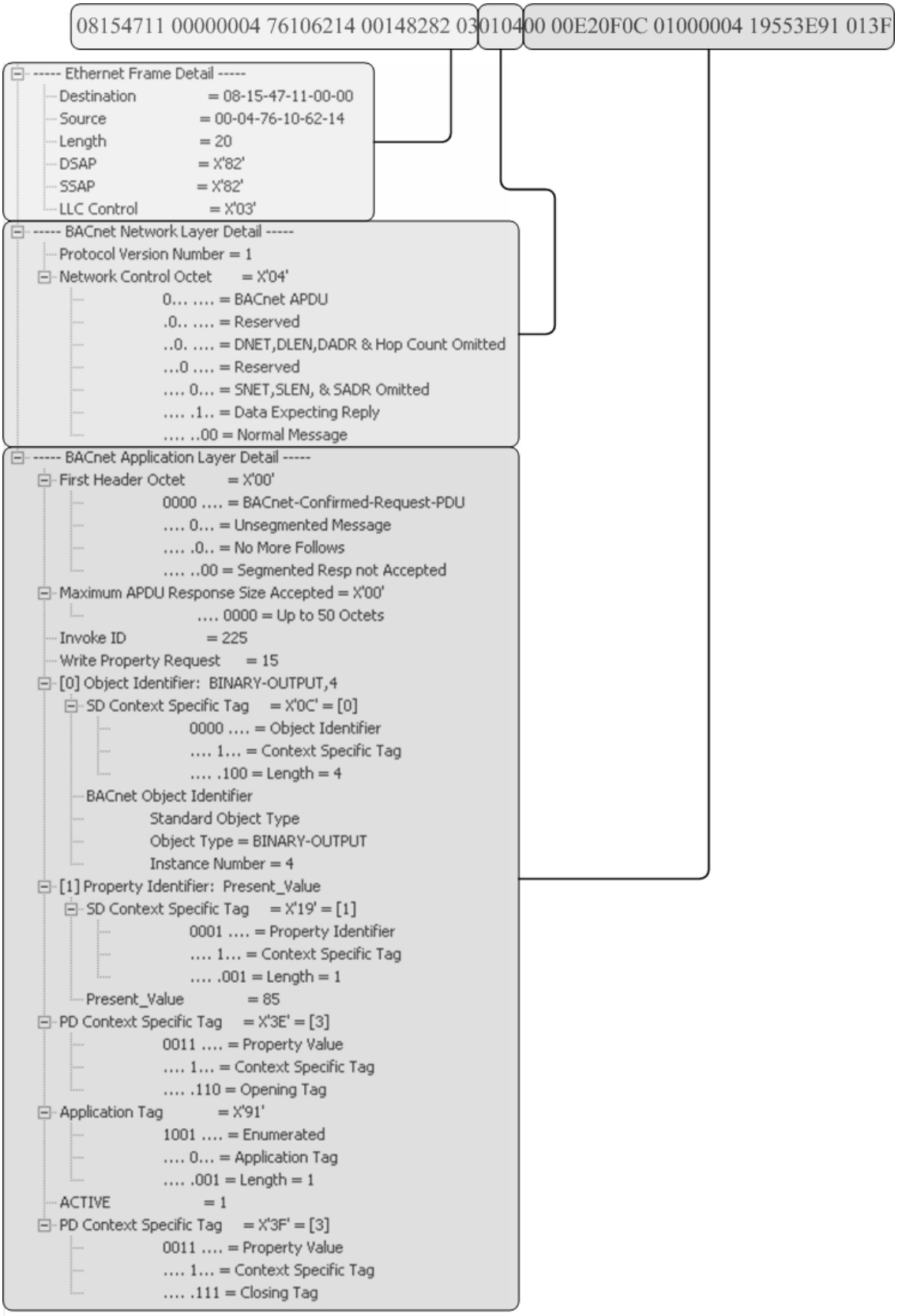

Abbildung 13: Write Property Frame

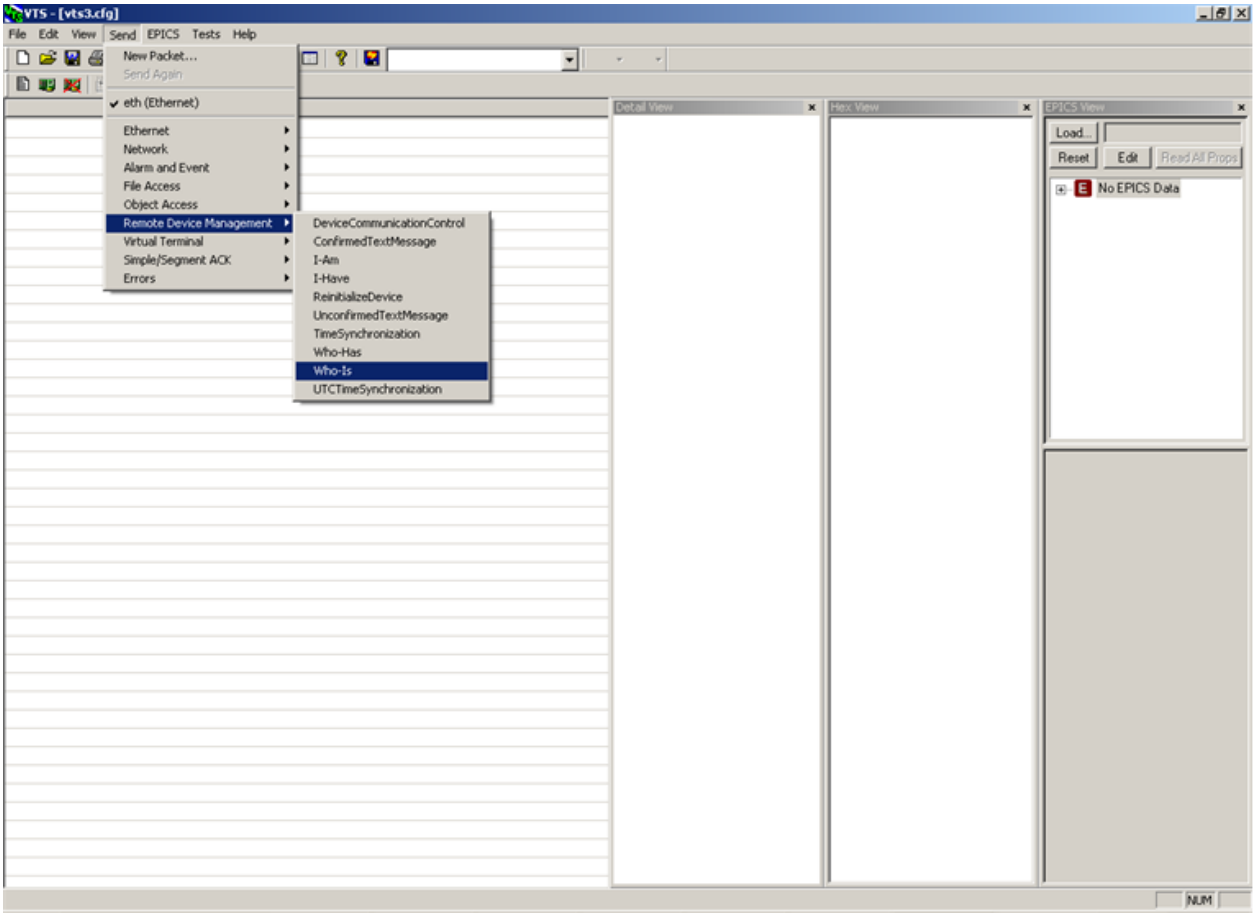

Abbildung 14: VTS - Visual Test Shell

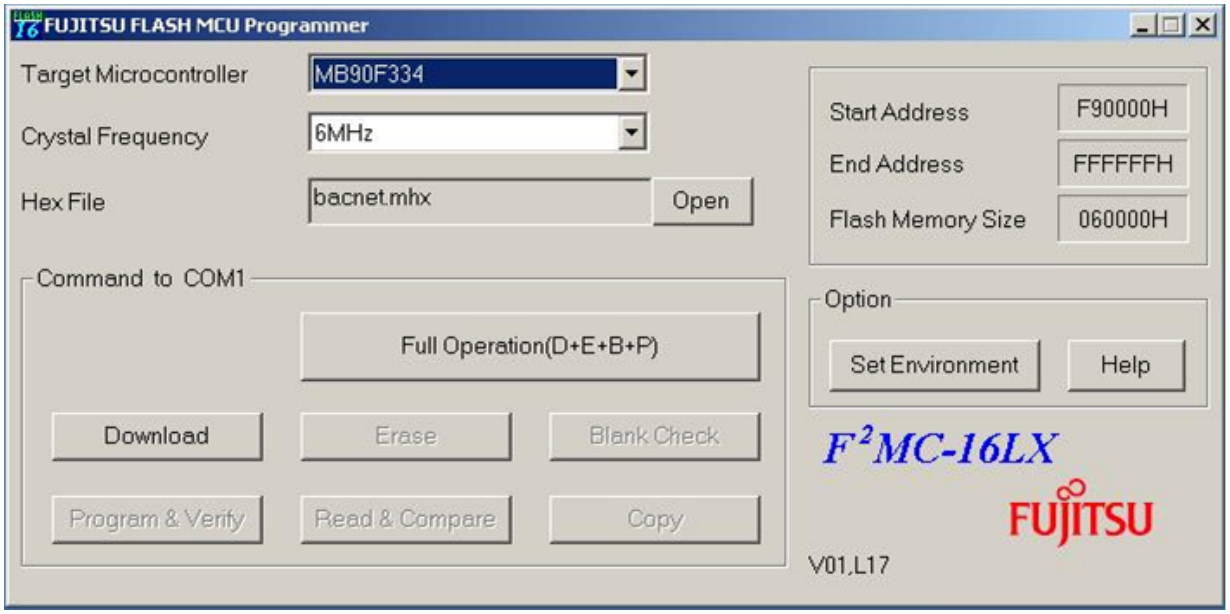

Abbildung 15: Flash Programmer

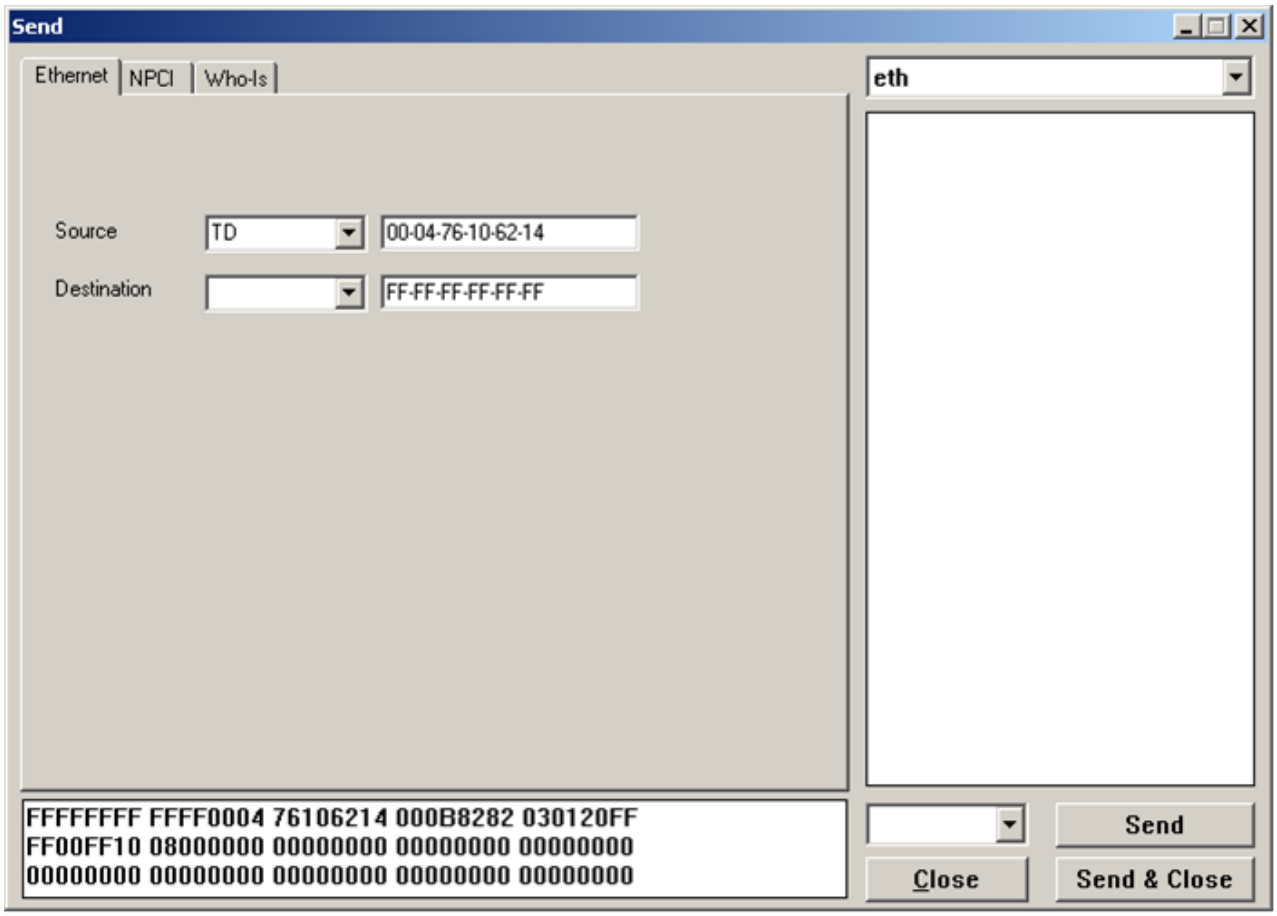

Abbildung 16: VTS - Who Is, Ethernet Konfiguration

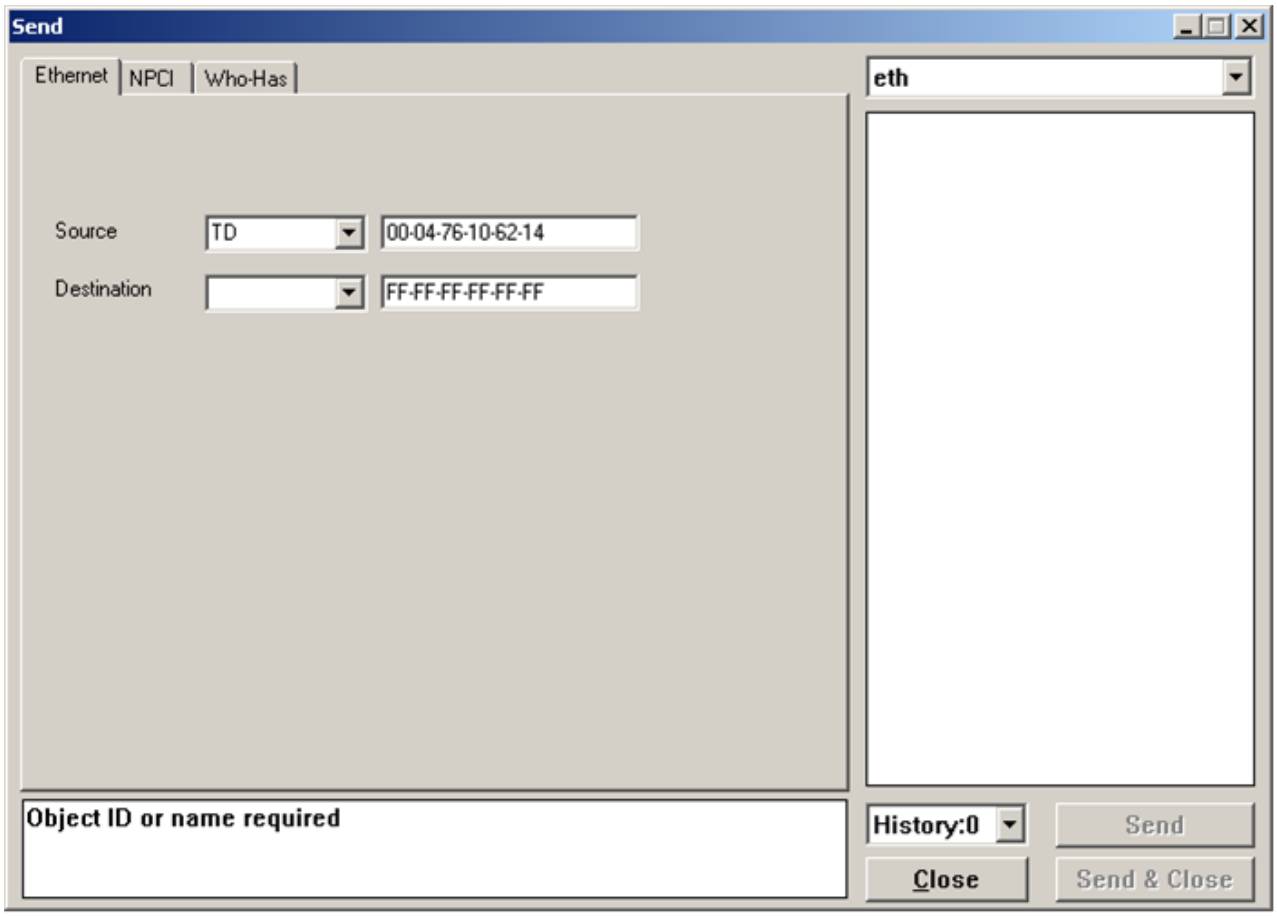

Abbildung 17: VTS - Who Has, Ethernet Konfiguration

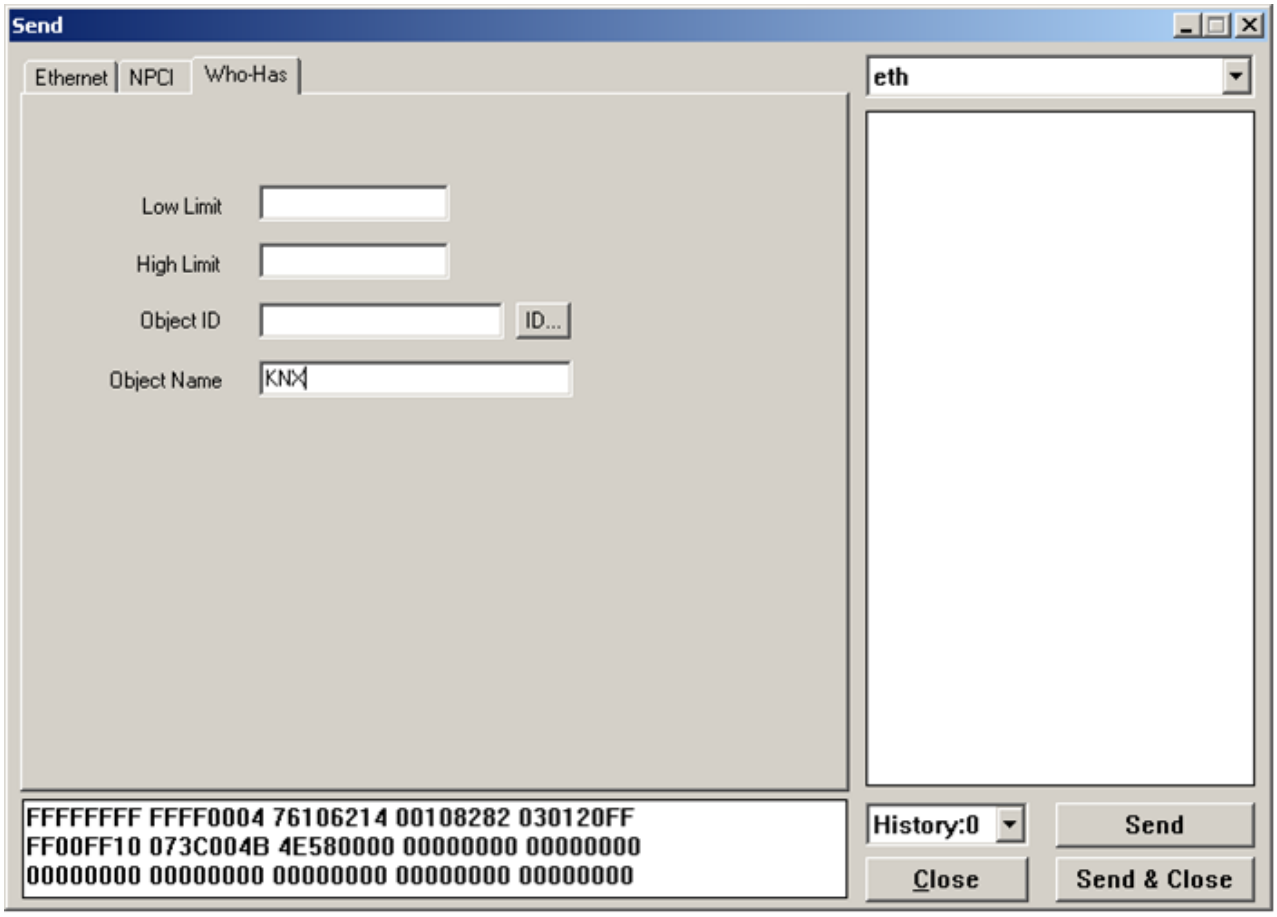

Abbildung 18: VTS - Who Has, Objekt Name

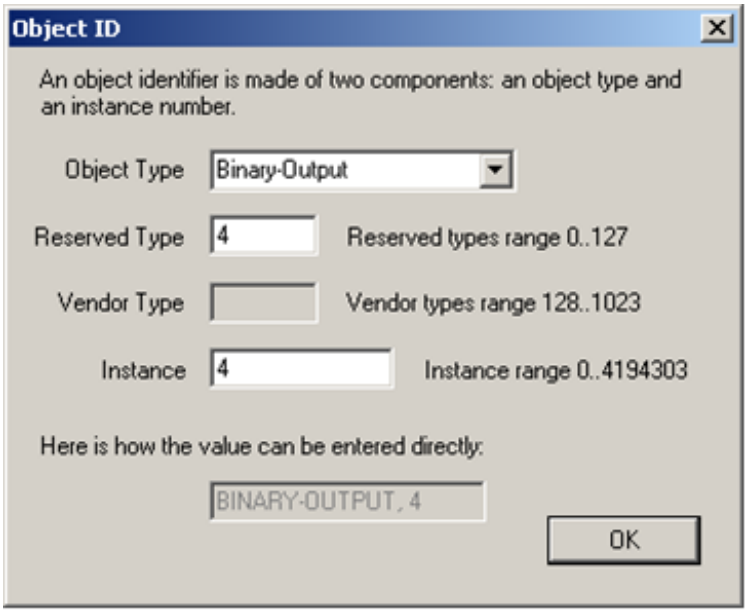

Abbildung 19: VTS - Who Has, Objekt ID

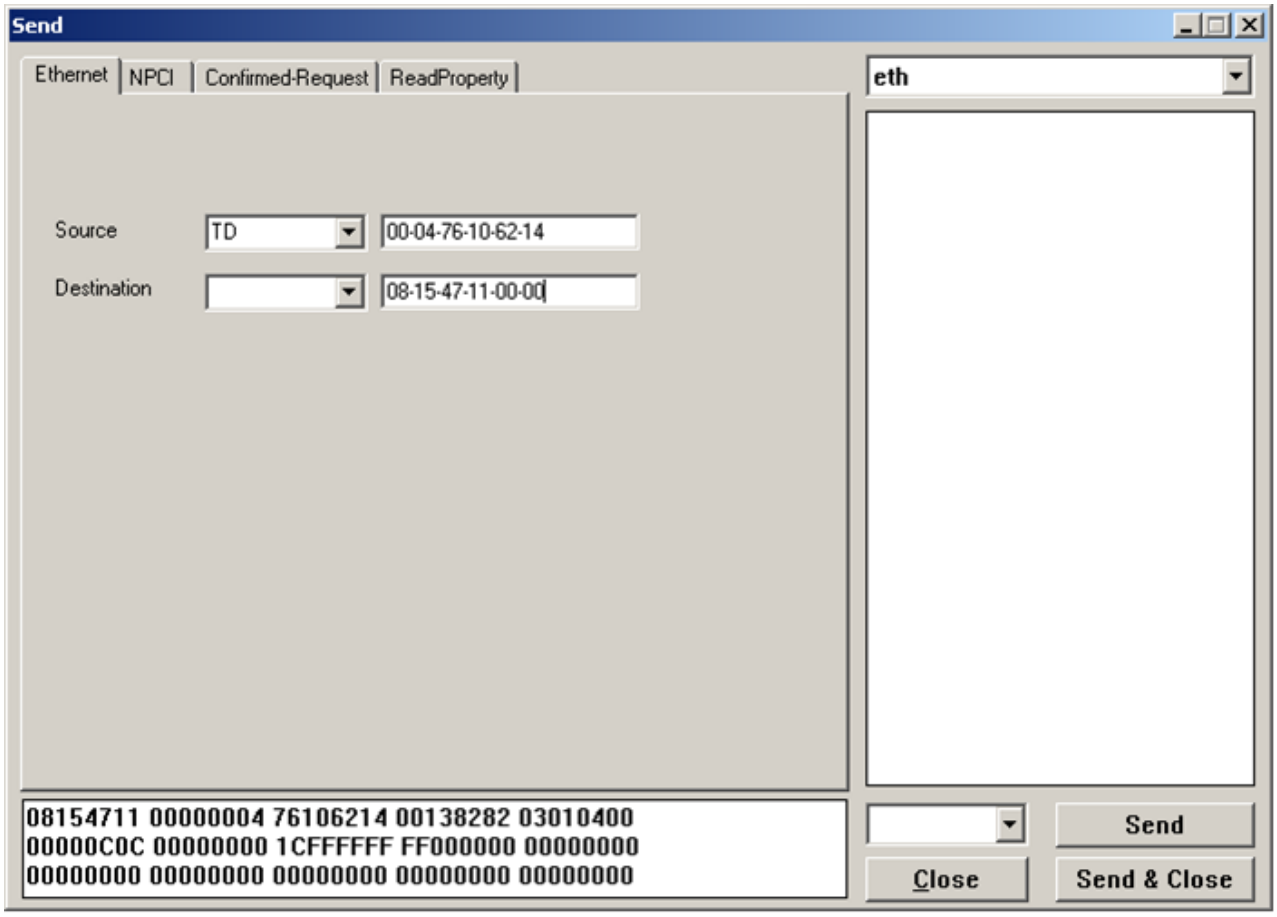

Abbildung 20: VTS - Read Property, Ethernet Konfiguration

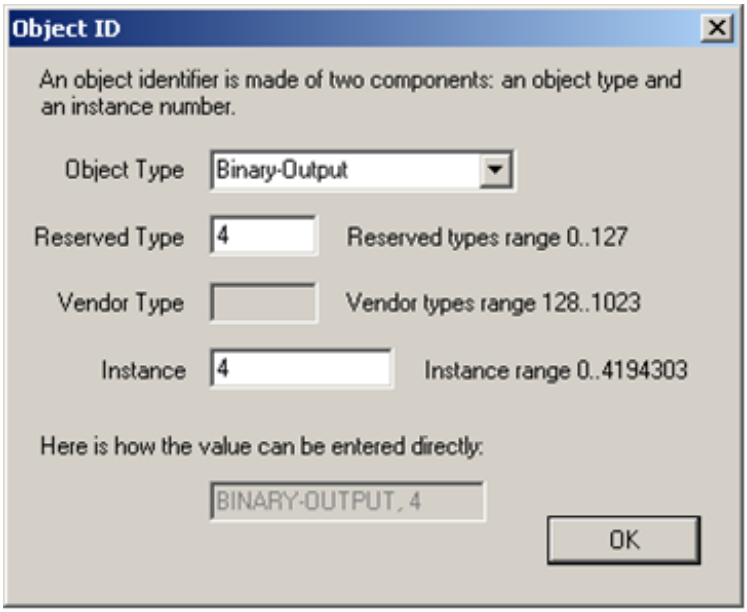

Abbildung 21: VTS - Read Property, Objekt ID

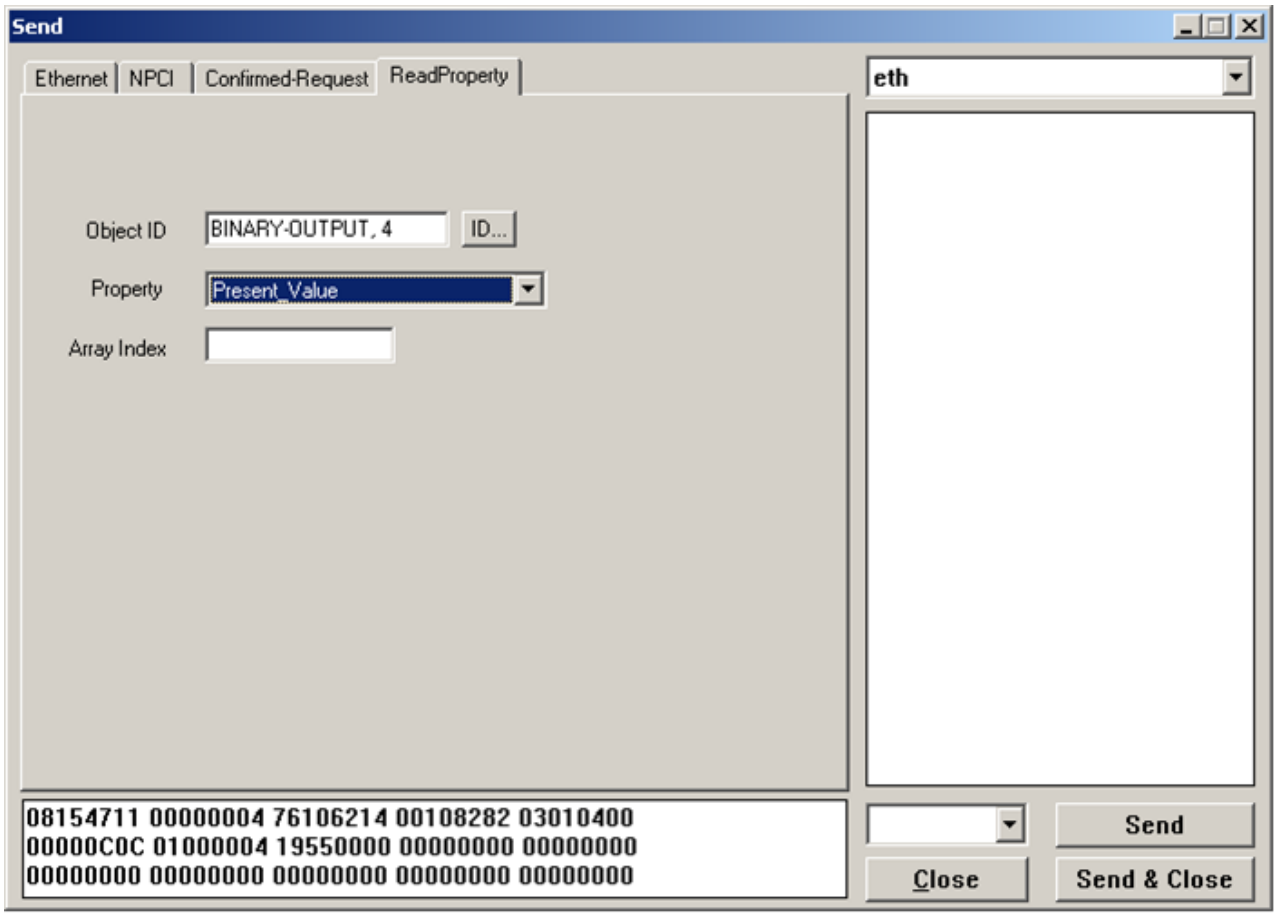

Abbildung 22: VTS - Read Property, Property

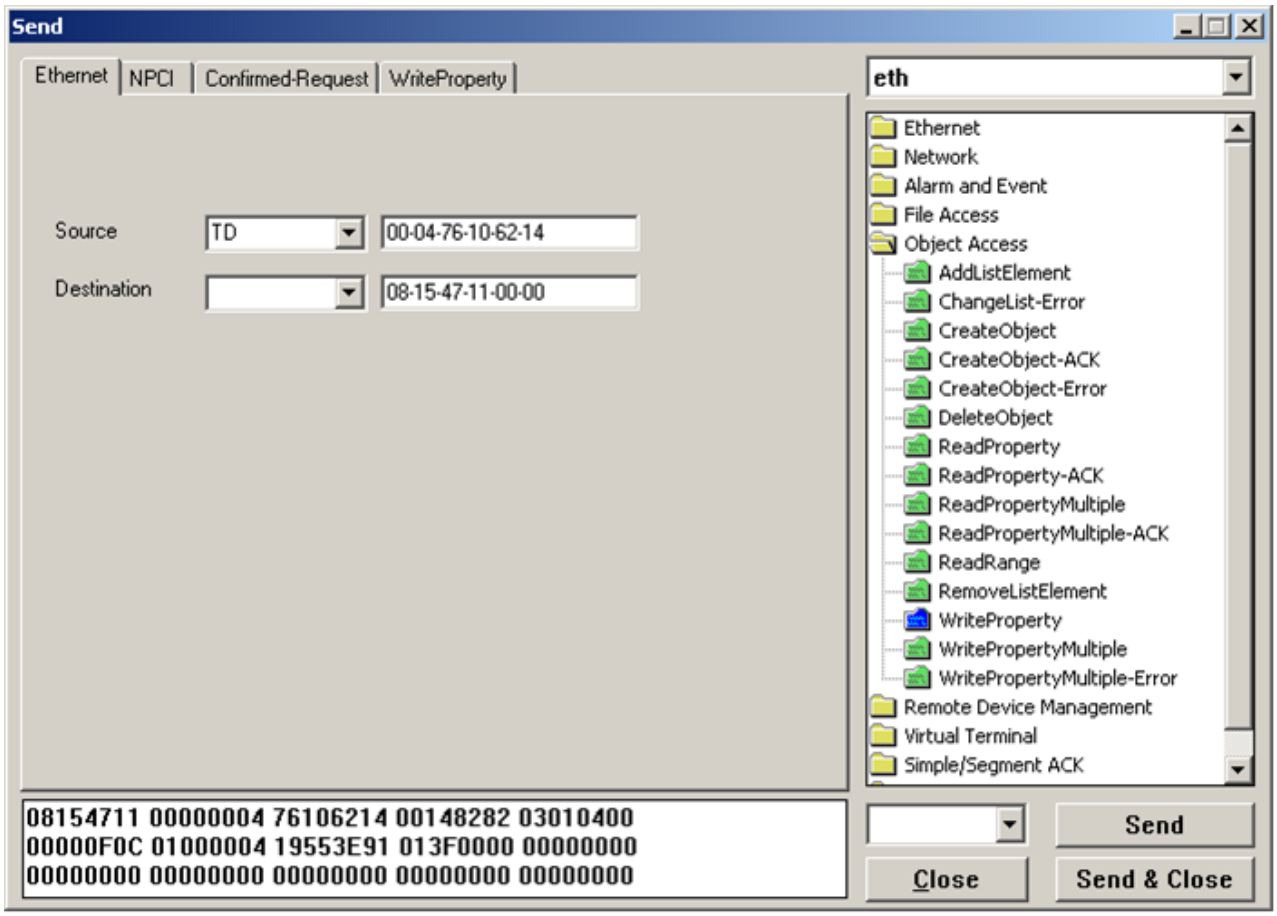

Abbildung 23: VTS - Write Property, Ethernet Konfiguration

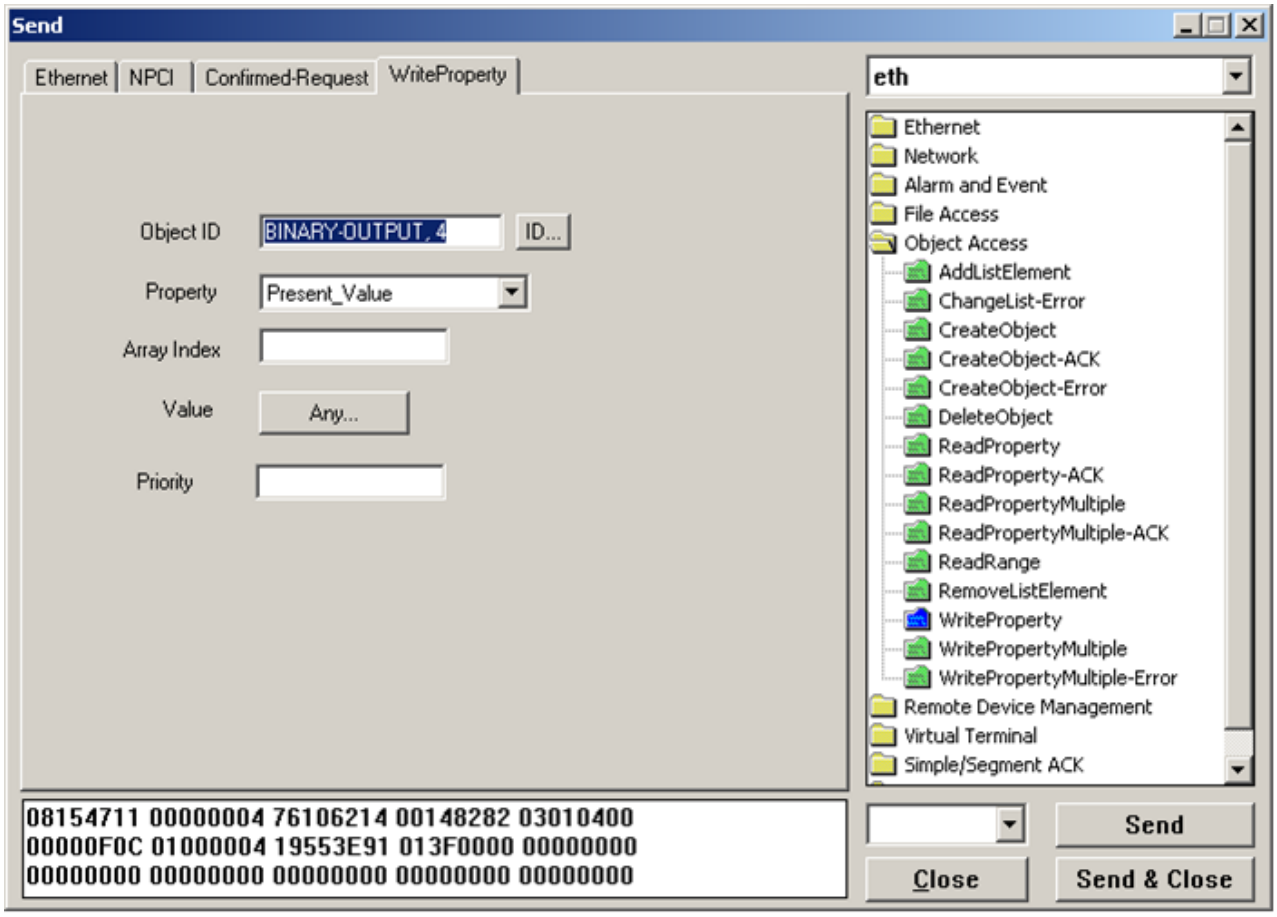

Abbildung 24: VTS - Write Property, Property

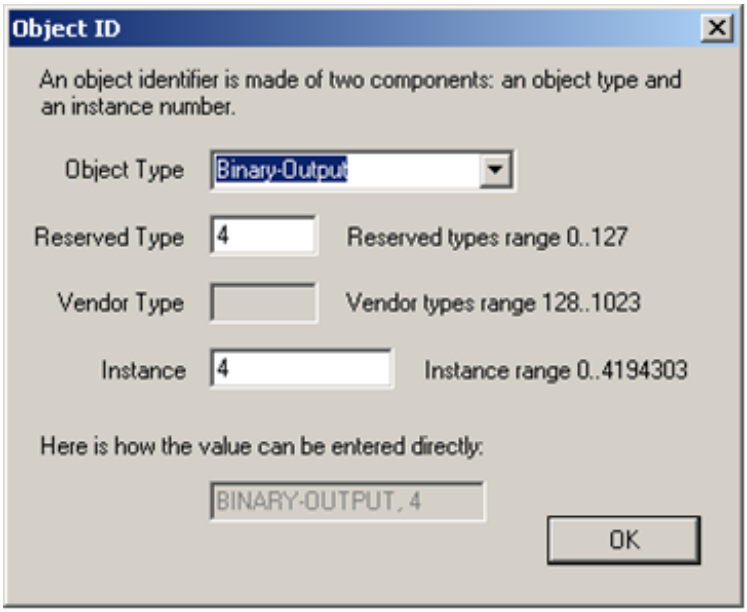

Abbildung 25: VTS - Write Property, Objekt ID

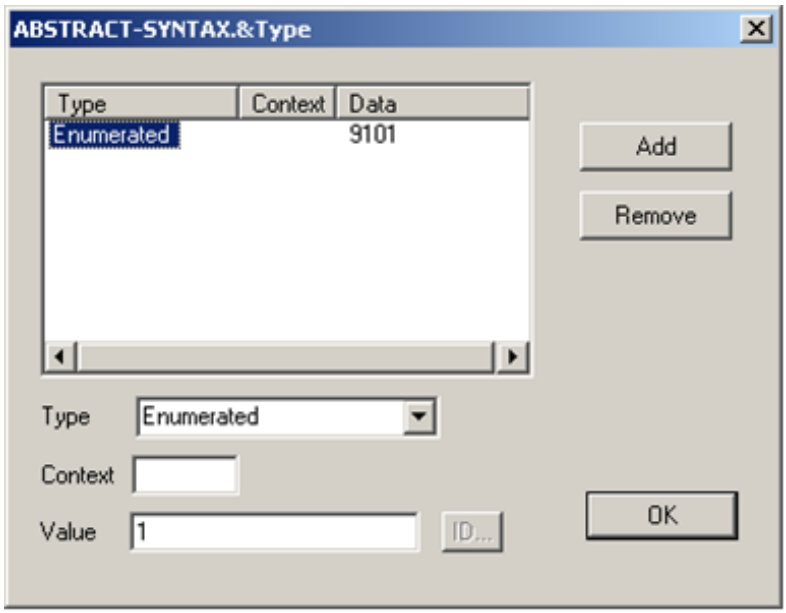

Abbildung 26: VTS - Write Property, Value