Getting started mit Modbus

Inhalt

Projekt History

Datum	Author	Modifikation
30.04.2009	TCS / sr	V1 Erstellung Dokumentation (Version 1) und Projekt für PG5 1.4.300
01.07.2009	TCS / sr	V2 Projekt und Dokumentation überarbeitet und angepasst.
15.07.2009	TCS / sr	PG5 2.0 Projekt importiert und Doku angepasst
01.12.2009	TCS / sdu	V3 Erklärung der UID und des Media Mappings verbessert

1 Einführung

Dieses Dokument soll einen einfachen Einstieg für die Verwendung des Saia Modbus bieten. Mit dem dazugehörigen PG5 Projekt kann es als Leitfaden für die Realisierung einer Modbus Kommunikation dienen.
Die in diesem Dokument enthaltenen Informationen sind ein Abstrakt von den entsprechenden Manuals und Online Hilfen und sollen Ihnen den Einstieg erleichtern. Für weitere Informationen konsultieren Sie bitte die entsprechenden Dokumente (siehe Kapitel „Referenzen“).

2 Benötigte Hard- und Software

Hardware
Dieses Projekt ist für folgende Hardwarekonstellation konfiguriert:

- PCD3.M5540 als Client IP und/oder RS
  Firmware 1.10.16 oder höher
- PCD3.M3330 als Server IP
  Firmware 1.10.16 oder höher
- PCD2.M5540 als Server RS
  Firmware 1.10.16 oder höher
- Ethernet Kabel (CAT5) für die Verbindung zwischen PCD3 Client (IP) und PCD3 Server (IP)
- Kable zum Verbinden der RS Schnittstellen RS 485
- Ein USB-Kabel (max. 1.8m) zur Programmierung der PCD

Software
Zur Programmierung der PCD werden folgenden Software inklusive gültiger Lizenz benötigt:

- PG5 2.0.100
  Zur Programmierung der PCD. (HLK Bibliothek wurde für die Uhr Funktionalität benützt, ist aber nicht Bedingung für den Betrieb der Modbus Kommunikation.)
- Modbus Saia Bibliothek

Natürlich ist es auch möglich, dieses Projekt mit anderer Hardware zu betreiben. Dafür müssen je nach Hardware spezifische Anpassungen der Konfiguration vorgenommen werden (Hardware Konfiguration in PG5, Software Settings in PG5, Verfügbarer Speicher im Fupla und entsprechende Einstellungen für die Kommunikation zwischen den PCDs).
Die Modbus Saia Bibliothek wird nur von den neuen Systemen unterstützt wie PCD3 und PCD2.M5xxx.

3 Grundlagen Modbus

Dieses Kapitel soll einen kurzen Überblick über Modbus geben. Über typische Applikationen, den geschichtlichen Hintergrund, die Besonderheiten von Modbus und den Vergleich zum S-Bus. Das Mapping der PCD Ressourcen ist kurz erklärt. Ebenfalls werden die Unterschiede zwischen der Saia Modbus Bibliothek und der bestehenden Bibliothek von Engiby aufgezeigt.

3.1 Typische Applikationen

3.1.1 Serial

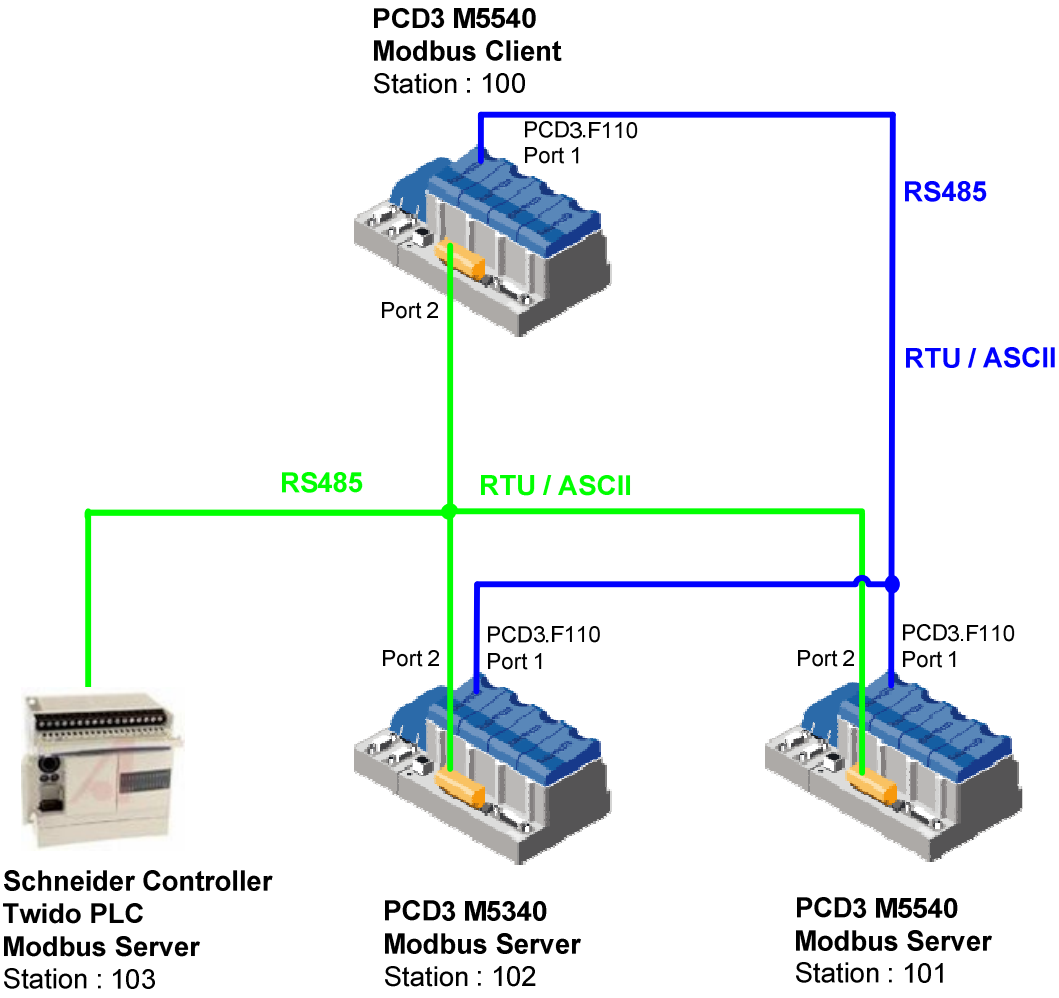

Bild 3.1.1.1 Serial Struktur

3.1.2 Ethernet

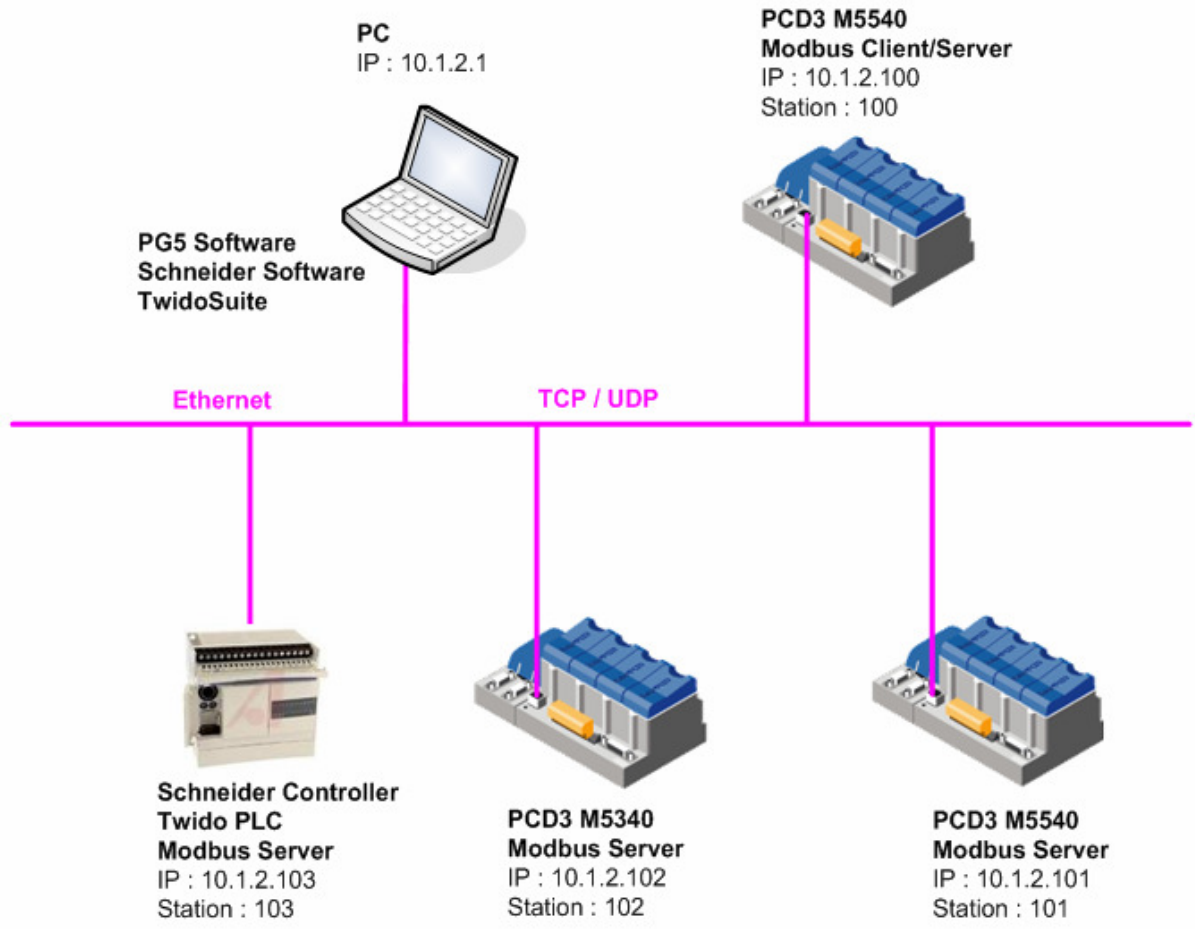

Bild 3.1.2.1 Ethernet Struktur

3.2 Geschichtlicher Hintergrund

Modbus wurde im Jahr 1979 von der Firma Modicon (heute Schneider) eingeführt. Es handelt sich um einen Single Master Bus. Das Modbus Protokoll wurde im Laufe der Zeit zu einem de facto Standard, der von sehr vielen Herstellern unterstützt wird.
Modbus Kommunikationsprotokolle:

- Modbus RTU: Binär codiert mit Breaks zwischen den Telegrammen. Diese Kommunikationsart ist effizient, aber empfindlich auf Delays in der Charakter Übertragung
- Modbus ASCII: Lässt sich vom Menschen leicht interpretieren / lesen. Es werden Start und Stopp Charakter verwendet. Die Telegramme sind etwa 2-mal länger als mit Modbus RTU.
- Modbus TCP/UDP: Wird als Client/Server Netzwerk betrieben. Implementierung von Modbus auf Ethernet (TCP/IP und UDP/IP). Dies erlaubt die Realisierung einer Multi-Master Netzwerktopologie.

3.3 Vergleich S-Bus ⇔ Modbus

Obwohl Serial S-Bus und Modbus RTU / ASCI auf denselben physical Layers verfügbar sind, bestehen einige Unterschiede. In den folgenden Tabellen sind die Protokolle einander gegenüber gestellt.

- Ein wichtiger Punkt ist, dass ein S-Bus Server (Slave) immer eine Adresse hat, während ein Modbus Server (Slave) mehrere UIDs (Unique Identifier) haben kann.
- Im S-Bus sind alle Medias (Register, Inputs, Flags etc.) fix adressiert. Bei Modbus jedoch kann für jede UID ein spezifisches Mapping definiert werden.
- Modbus Holding Register sind 16 Bit lang. PCD Register sind 32 Bit lang.

3.3.1 Vergleich Serial S-Bus ⇔ Modbus RTU/ASCII

Kriterium	Serial S-Bus	Modbus RTU/ASCII
Physikalische Schnittstelle	RS 485, RS 232, RS 422, current loop...	Dito, Serial Modem nicht unterstütz von unserer Implementation
Max. Anzahl Master	1	Dito
Max. Anzahl Slaves	252	247
Adressen pro Slave	1 S-Bus Adresse	Bis zu 9 Unique Ids (UID)*
Broadcast Adresse	255	0
Checksumme	CRC 16	Dito
1 Bit Daten	Input / Output, Flag	Diskret Input (DI, nur lesbar) Coil (les- und schreibbar)
16/32 Bit Daten	Register (32 Bit) DB (32 Bit)	Input Register (IR, 16 Bit nur lesbar) Holding Register (HR, 16 Bit, les- und schreibbar)
Erste Daten Adresse	0	1
Unterstützte Baudraten	300-115000	1200-115000**
Anzahl Ports die man mit dem Protokoll betreiben kann	So viele wie es Serielle Schnittstellen hat.	Maximum 10
Können Protokolle gemischt werden auf einer PCD	Ja	Dito
Maximale Telegramm Länge (Daten)	128 Bytes	253 Bytes
PCD3.F2xx Module unterstützt	Ja	Noch nicht
Einfluss auf die Leistung der Steuerung.	Die Leistung nimmt ab mit jedem konfigurierten Port, je nach Baudrate.	Dito

*bei unserer Implementation, plus default UID 0

** von den unterstützen Steuerungen abhängig

3.3.2 Vergleich Ether-S-Bus ⇔ Modbus TCP/UDP

Kriterium	Ether S-Bus	Modbus TCP/UDP
Physikalische Schnittstelle	Ethernet	Dito
Verbindung	Nur UDP	TCP oder UDP
Standard Port	5050	502
Maximale Anzahl Server pro Netzwerk	Viele	Dito
Maximale Anzahl Clients pro Netzwerk	Viele	Dito
Maximale Anzahl von Client Verbindungen pro Netzwerk	Keine Einschränkung, da sie sequenziell abgearbeitet werden	10
Maximale Anzahl von Server Verbindungen pro Netzwerk	Keine Einschränkung, da sie sequenziell abgearbeitet werden	26 minus Anzahl der Client Verbindungen
Adressen pro Slave / Server	1 S-Bus Adresse	Bis zu 9 Unique Ids (UID)*

*bei unserer Implementation, plus default UID 0

Wichtiger Unterschied: Bei S-Bus ist ein Kommunikations-Channel immer einem physikalischen Port zugeordnet. Ein Modbus Channel besteht aus Port und Protokoll. Das heisst es können auf einem physikalischen Ethernet Port mehrere Channels mit unterschiedlichen Protokollen definiert werden.

z.B TCP auf 502, UDP auf 502, TCP auf 503 und ASCII auf dem seriellen Port 2.

3.4 Modbus Media Mapping und Eigenschaften

3.4.1 Master/Slave ⇔ Client/Server

Der serielle Modbus ist Master/Slave orientiert. Modbus TCP/UDP jedoch ist Client/Server orientiert. Zur Vereinfachung wird in allen FBoxen die Bezeichnung Client/Server verwendet. Wobei Client=Master und Server=Slave entspricht.

3.4.2 16 Bit ⇔ 32 Bit

Modbus ist 16 Bit (Word) orientiert. Das Übertragen von 32 Bit Werten ist üblich, aber nicht standardisiert. Weder für Integer noch für Floatingpoint Werte. Daher findet man auch verschiedene Wege des Mappings von 32 Bit Werten auf Holding Register und vice versa.

Damit wir all diese Fälle abdecken können unterstützen wir alle nötigen Mapping Methoden und Prozesse wie „Word swapping“ und Konvertierung. Wenn 32 Bit Daten mit einem Fremdprodukt ausgetauscht werden, müssen folgende Mapping Einstellungen bestimmt sein:

- Singned / unsigned?
- Lower word zuerst oder zuletzt?
- Bei Floatingpoint: IEEE oder FFP

3.4.3 Offset

Modbus Ressourcen werden von 1 an aufwärts gezählt, während PCD Register bei 0 beginnen. Dieser Offset kann zu Problemen führen, da vielleicht auf Grund dieses Offsets der falsche Wert gelesen oder geschrieben wird. Ein überprüfen der Offset Einstellungen auf beiden Seiten ist daher notwendig. Die FBoxen erlauben die Konfiguration eines Offsets.

3.4.4 Was ist eine UID

Eine UID (Unique Identifier) ist wie eine S-Bus Adresse, jedoch kann jede Station mehrere UIDs haben. Auf einem seriellen Netzwerk darf jede UID nur einmal vorkommen. Zu jeder UID gehört ein spezifisches Mapping. Eine UID kann über alle Modbus Kommunikationsports angesprochen werden (TCP, UDP, Serial).

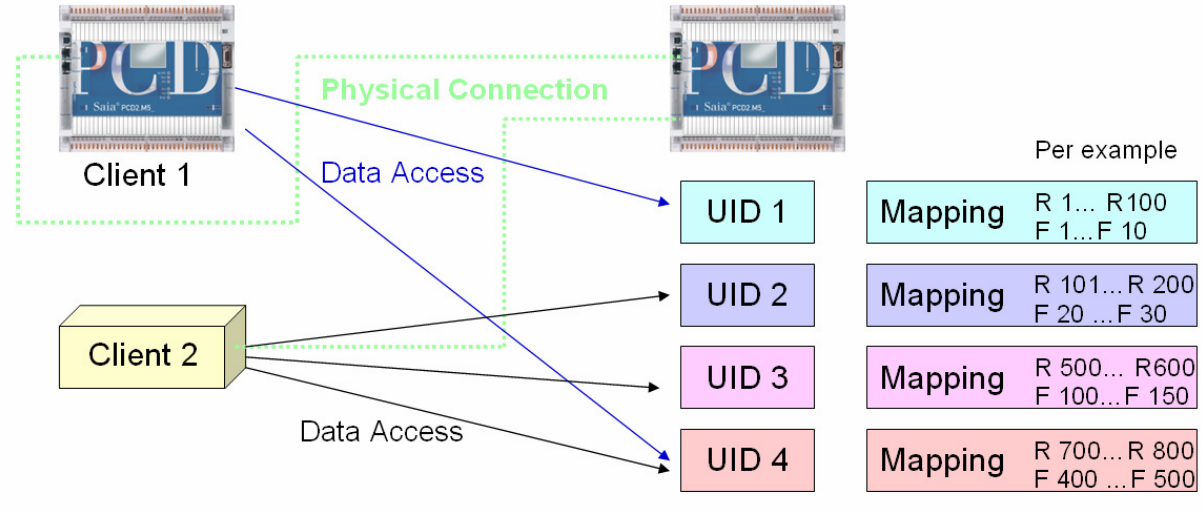

Bild 3.4.4.1 UID

3.4.5 Warum benötigen wir ein Mapping

Jede UID gibt mit dem Mapping Zugriff auf bestimmte Ressourcen. Als Server können mit den folgenden FBoxen Mappings erstellt werden:

Bild 3.4.5.1 Mapping FBoxen

Als Client können die Daten mit folgenden FBoxen gelesen oder geschrieben und auf die entsprechenden Ressourcen verteilt werden:

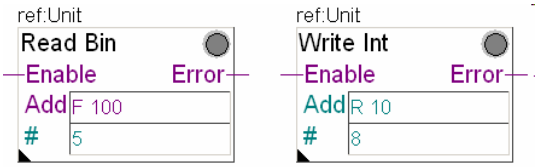

Bild 3.4.5.2 Read/Write FBoxen

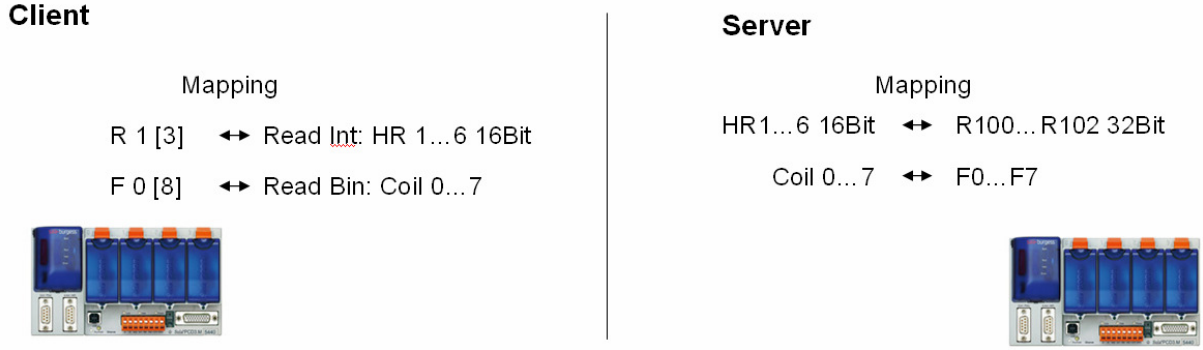

Bild 3.4.5.3 Read/Write FBoxen

3.4.6 Mapping und UID's

Jede Server PCD kann bis zu 9 Modbus UID's haben. Diese können alle vom Anwender konfiguriert werden. Es kann für jede UID ein separates Mapping definiert werden. Es können bis zu 10 Mapping Bereiche pro UID erstellt werden. Anstelle des Anwender konfigurierbaren Mapping kann immer das Default Mapping verwendet werden. In diesem Fall sind alle Ressourcen verfügbar. Wenn anderen Mapping vorhanden sind, muss sichergestellt werden, dass man keine Kollisionen verursacht.

Die UID's sind immer für alle Modbus Ports gültig (Serial und TCP/UDP). In einem seriellen Netzwerk darf jede UID nur einmal vorkommen.

UID's können in Laufzeit umkonfiguriert werden.

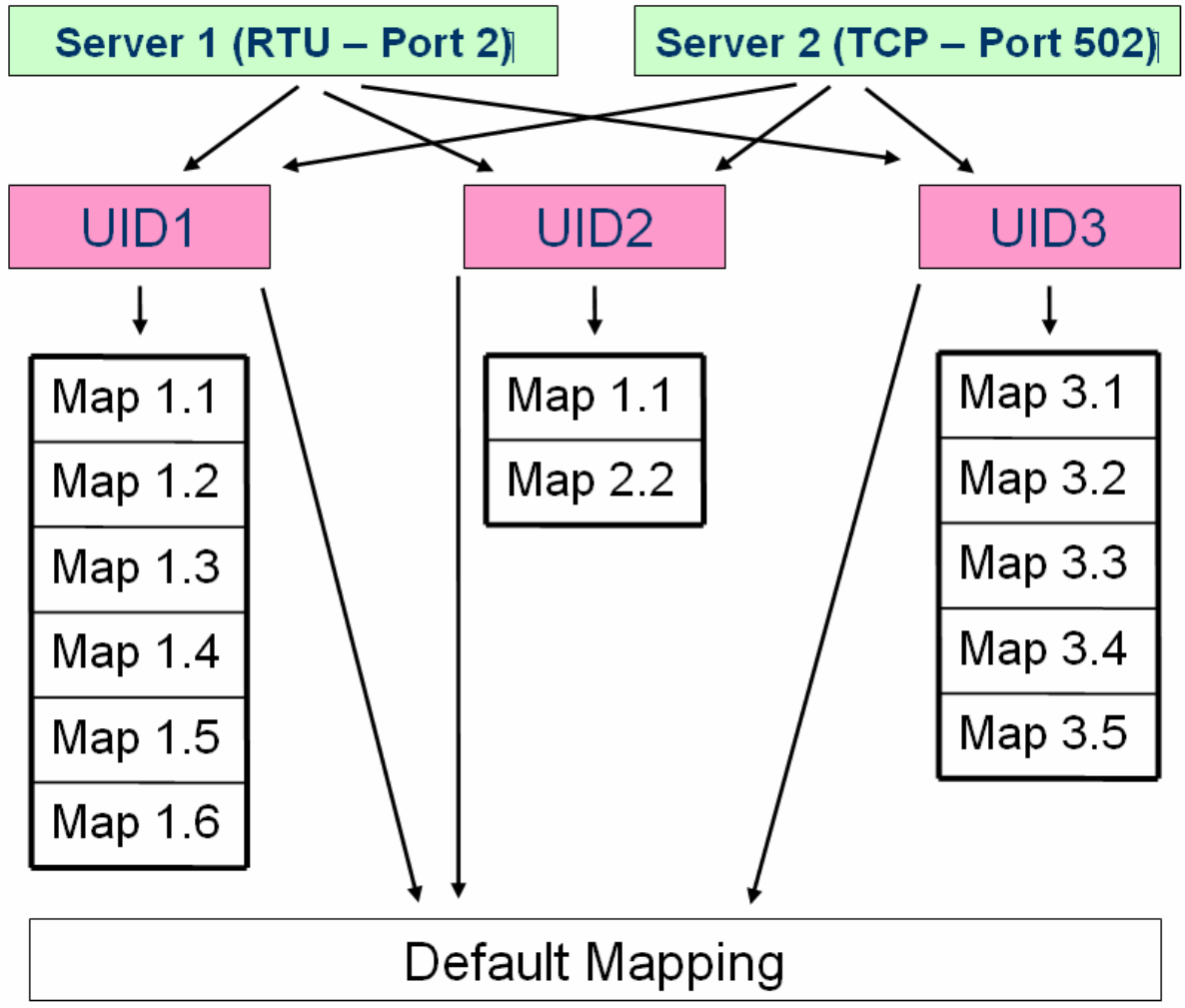

Bild 3.4.6.1 Mapping

Wenn kein Mapping erstellt wird, ist das Default Mapping aktiv. Sobald aber ein Mapping für eine UID erstellt wird, hat dieses Priorität.

UID 0 ist die Default UID für den seriellen Modbus. Diese UID ist immer vorhanden und kann benutzt werden um Broadcast Telegramme zu verschicken.

Bei der TCP/UDP Kommunikation ist die Default UID ebenfalls immer aktiv, aber es ist kein Default mapping definiert. Anfragen an einen nicht existierende UID werden immer von der Default UID beantwortet. Sobald für die Default UID ein Mapping

definiert ist, wird dies für die Default UID und alle nicht konfigurierten UID's verwendet.

3.4.7 Channel Definition

Ein Modbus Channel besteht immer aus einem Paar aus Port und Protokoll. Zum Beispiel TCP auf Port 502 oder ASCII auf dem Seriellen Port 2.

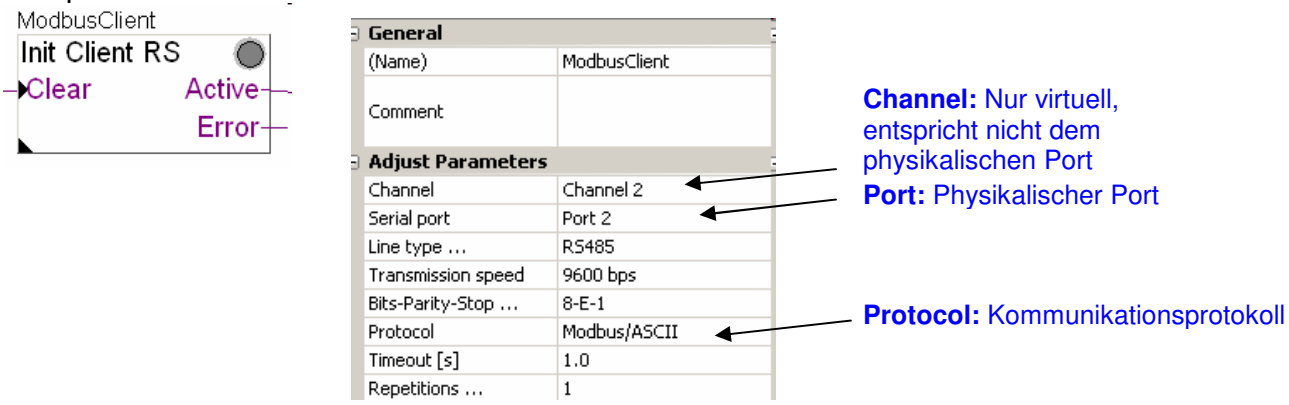

Bild 3.4.7.3 Read/Write FBoxen

Jeder Channel kann nur einmal definiert werden. Es können aber bis zu 4 Server-Channels auf einer Steuerung erstellt werden.

3.4.8 Verbindungen

Wenn Daten von einem Server angefragt werden durch einen Client, wird automatisch eine Verbindung geöffnet. Der Anwender muss sich nicht um das Öffnen und Schliessen der Verbindungen kümmern.
Trotzdem kann es wichtig sein zu wissen, wann eine Verbindung geöffnet oder geschlossen wird, da die Anzahl Verbindungen limitiert ist. Für Details kann das Modbus Handbuch 26/866 konsultiert werden.

3.4.9 Holes

Werden 32 Bit Werte auf Register gemappt, kann man zwischen zwei Varianten wählen. Man kann mit Holes (Löcher) arbeiten. Die Daten sind dann einfacher zu interpretieren, weil die Register Adressen auf der PCD mit den Holding Register Adressen auf dem Modbus übereinstimmen. Es werden aber in diesem Fall nur die geraden Register genutzt, was dazu führt, dass die ungeraden Register unbenutzt bleiben. Ohne Holes (Löcher) ist es kompakter und Ressourcen sparender, die Zuordnung gestaltet sich aber etwas komplizierter.

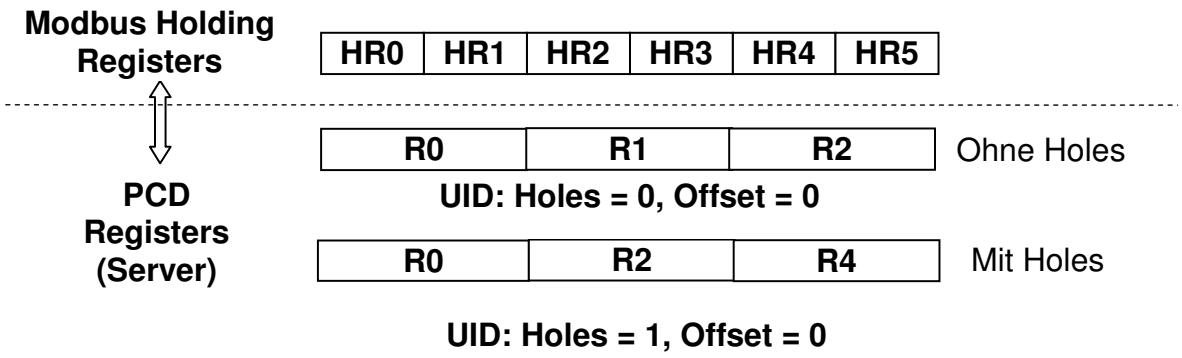

Bild 3.4.9.1 Holes

3.4.10 Programmierung des Client

Als Client müssen wir wissen, auf welche UID wir uns verbinden sollen und wo die benötigten Daten gemappt sind.
Weiter ist abzuklären, ob es sich um binäre, 16 Bit Werte oder 32 Bit Werte handelt und ob es allenfalls einen Offset hat oder Holes.

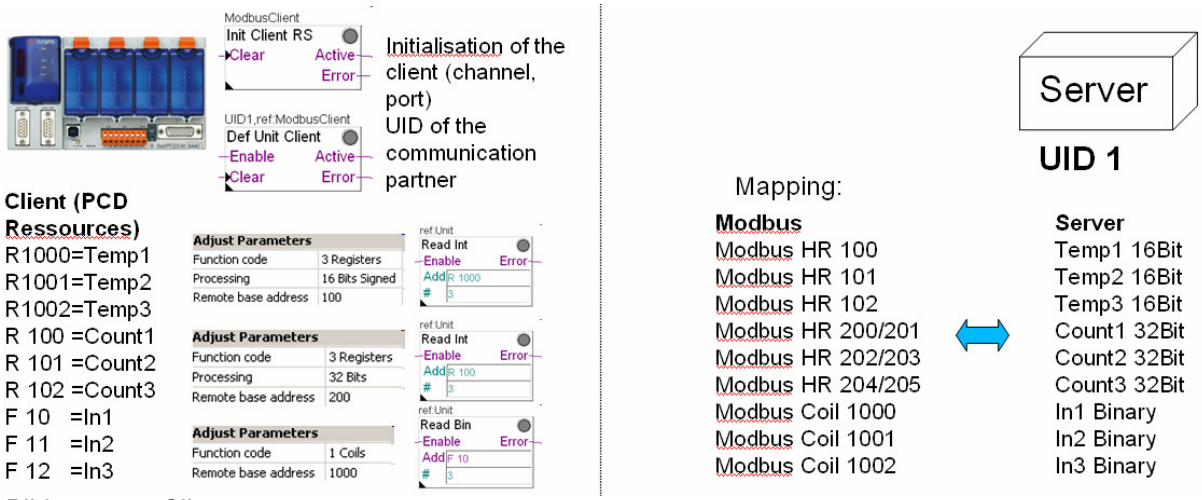

Bild 3.4.10.1 Client

Es ist wichtig, dass der ganze Bereich reserviert wird für die empfangenen Daten. Das heisst, wenn die Basis Adresse Register 1000 ist und 3 Elemente gelesen werden, muss Register 1000, 1001 und 1002 reserviert werden. PG5 bemerkt beim kompilieren nicht, dass all diese Register verwendet sind. Am Besten man reserviert einen Array R 1000 [3] oder man stellt auf eine andere Art sicher, dass diese Register nicht anderweitig verwendet oder überschrieben werden.

3.4.11 Programmierung des Servers

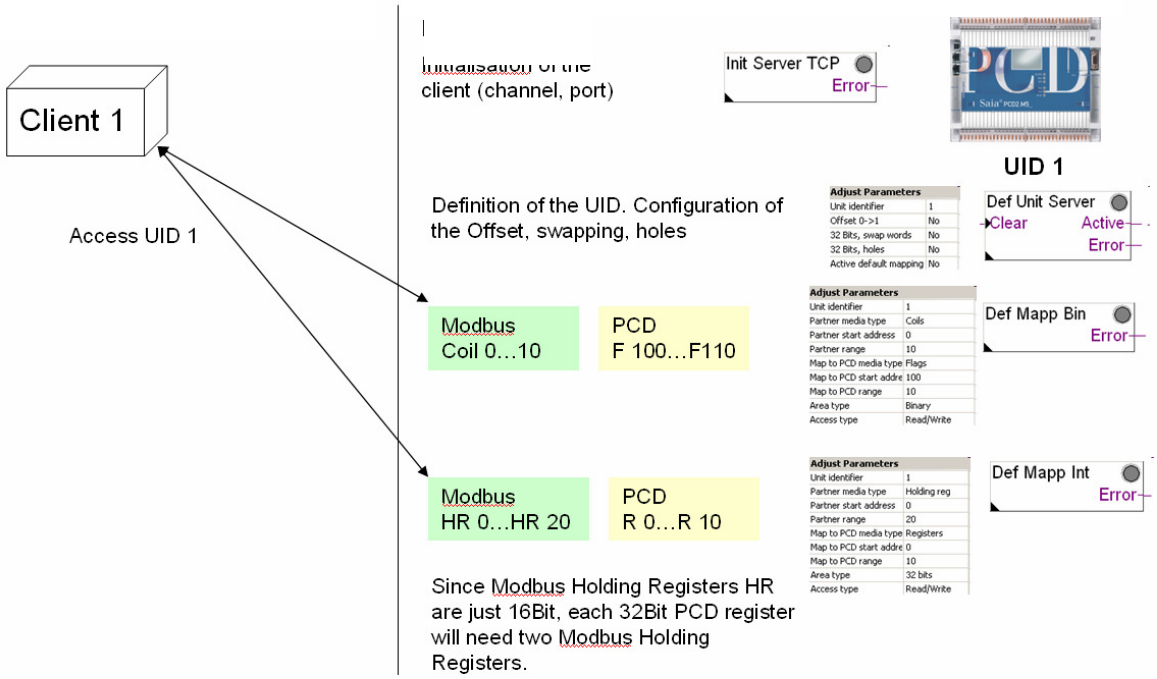

Bild 3.4.11.1 Server

Als Server stellen werden mit dem Datamapping die Ressourcen für die jeweilige UID zur Verfügung.

3.5 Vergleich Saia Lösung ⇔ Engiby Lösung

Wurde vorher die Engiby Library verwendet, so ist darauf zu achten, dass das Mapping korrekt übernommen wird. Die Bezeichnungen sind teilweise unterschiedlich, was zu Verwirrung führen kann. Hier deshalb ein Überblick über die unterschiedlichen Bezeichnungen in den Engiby und Saia FBoxen:

Saia FBoxen	**Engiby FBoxen**
32 bit swap words NO	LITTLE Endian
32 bit swap words YES	BIG Endian
Holding Register	R
Input Register	InR

3.6 Limiten die zu beachten sind

Zusammenfassung der Limitierungen auf dem Modbus Netzwerk, die es zu beachten gilt:

- Maximal 247 Server (Slaves) pro Bus für Serial Modbus
- Maximal 10 Channels im Ganzen, davon höchstens 4 Server Channels (Port+Protokoll)
- Maximal 9 Anwender Spezifische UID's (+default ID)
- Maximal 10 Mapping Bereiche pro UID
- Serielle Modemverbindung wird nicht unterstützt
- Unterstützte Modbus Funktionen:
    - READ_COILS
    - READ-DISC_INPUT
    - READ_HOLD_REG
    - READ_INPUT_REG
    - WRITE_SINGLE_COIL
    - WRITE_SINGLE_REG (16Bit only)
    - WRITE_MULTIPLE_COILS
    - WRITE_MULTIPLE_REGS

4 Beschreibung des Beispielprojekts

Das Beispielprojekt besteht aus 3 Saia PCD's. Es gibt eine Kommunikation über IP (Modbus TCP) zwischen dem „Client“ und dem „Modbus_Server“ und eine serielle Kommunikation zwischen dem „Client“ und dem „Server_RS“ RS 485 (Modbus ASCII).

4.1 Datenaustausch über Modbus

Die Ressourcen der PCD (R, F, T, C, I, O…) können nicht direkt vom Modbus übernommen werden. Auf dem Modbus stehen uns Holding Register und Coils (lesen/schreiben) oder Input Register und Discret Inputs (nur lesen) zur Verfügung. Aus diesem Grund muss auf dem Server für jede UID ein Mapping definiert werden, sofern nicht das Default Mapping genutzt wird. Mit diesem Mapping weisen wir die zu übertragenden Ressourcen der PCD den Modbus Ressourcen zu. Details zum Mapping können dem Modbus Handbuch 26/866 und dem Kapitel 3.4.4 dieses Dokuments entnommen werden.

4.1.1 Kommunikation IP

Zwischen der PCD „Client“ und der PCD „Modbus_Server“ werden folgende Daten ausgetauscht:
Auf dem „Modbus_Server“ wird die Uhr in die Register 100, 101 und 102 gespeichert (32 Bit Register). Diese wird über Modbus auf den „Client“ übertragen und dort in die Register 100, 101 und 102 gespeichert. So kann die Uhr auf dem Client mit den Werten vom Server überschrieben werden.
Die Flags 1000 bis 1007 werden vom „Client“ über Modbus zum „Modbus_Server“ auf die Flags 1000 bis 1007 übertragen. Es wird immer ein Flag nach dem anderen hoch gesetzt. Wenn die Kommunikation korrekt funktioniert kann dieses hochzählen auch auf dem „Modbus_Server“ beobachtet werden.
Das Datamapping sieht in diesem Fall folgendermassen aus:

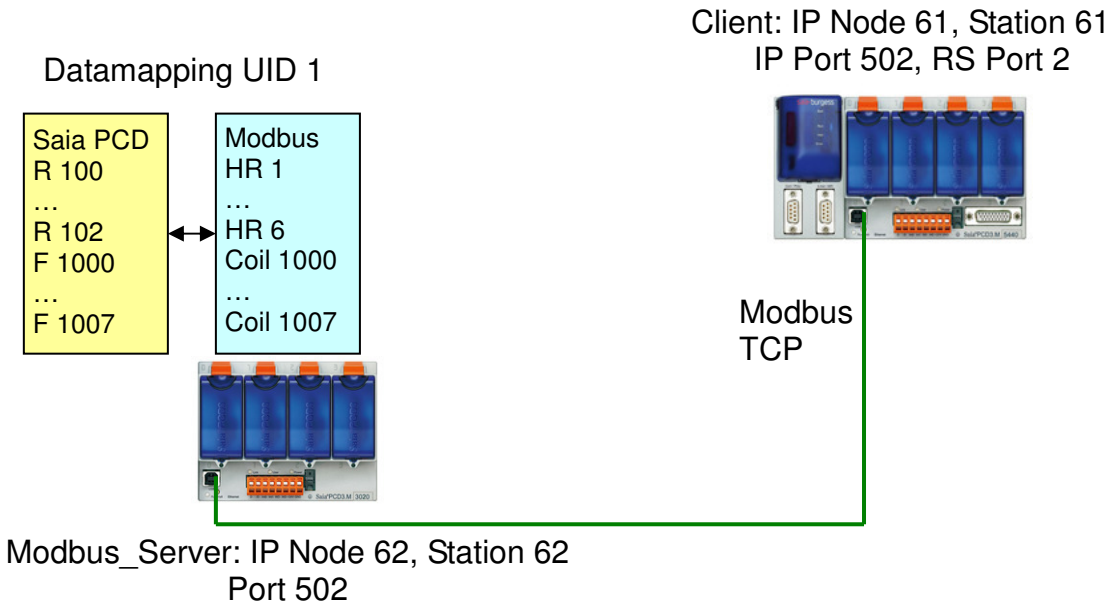

Bild 4.1.1.1 Modbus Netzwerktopologie

Client IP Saia PCD Medias (R u. F)	**Modbus Holding Register / Coils**	**Server IP Saia PCD Medias (R u. F)**
R 100…R 102 (32 Bit)	<= HR 1…6 (16 Bit) <=	R 100…R 102 (32 Bit)
F 1000…F 1007	=> Coil 1000…1007=>	F 1000…F 1007

4.1.2 Kommunikation RS 485

Zwischen der PCD „Client“ und der PCD „Server_RS“ werden folgende Daten ausgetauscht:
Der „Client“ liest 8 Register und 8 Flags vom „Server RS“. Es werden die Register 0 bis 7 und die Flags 0 bis 7 vom „Server_RS“ gelesen. Die Register werden in diesem Fall als 16 Bit signed Werte übertragen. Das Default Mapping wird genutzt. Für 16 Bit signed Werte heisst dies, dass alle PCD Register als 16 Bit signed Werte auf dem Modbus zur Verfügung gestellt werden. Sie befinden sich in den Modbus Holding Register 1…10000. Die Flags werden ebenfalls alle zur Verfügung gestellt und befinden sich auf den Coils 1…10000.
Das Datamapping für die im Projekt verwendeten Ressourcen sieht folgendermassen aus:

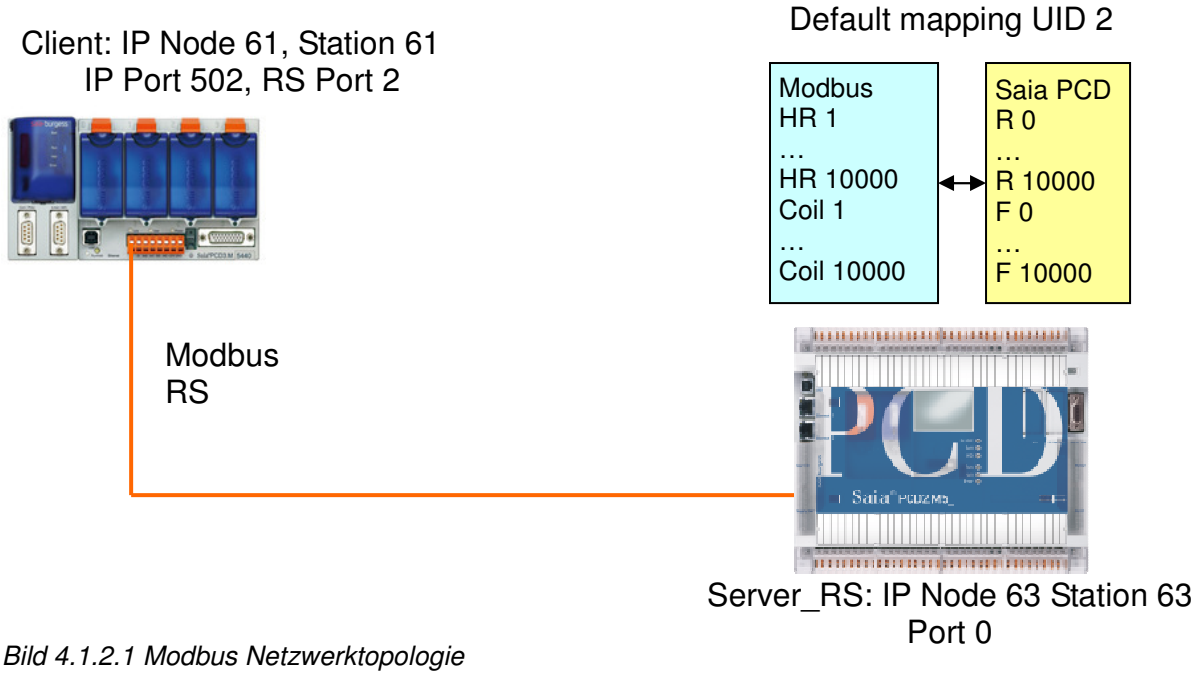

Bild 4.1.2.1 Modbus Netzwerktopologie

Client RS Saia PCD Medias (R u. F)	**Modbus Holding Register / Coils**	**Server RS Saia PCD Medias (R u. F)**
R 0…R 7 (16 Bit signed)	<= HR 1…8 (16 Bit) <=	R 0…R 7 (16 Bit signed)
F 100…F 107	=> Coil 1…8=>	F 0…F 7

5 Vorbereitung des Beispielprojekts

Um das Projekt in das PG5 zu importieren, kann die Funktion „Restore“ aus dem Menu „Projekt“ des PG5 Projekt Managers verwendet werden.

Damit eine Modbus Kommunikation möglich ist, müssen mindestens zwei Stationen konfiguriert und programmiert werden.

5.1 PCD konfigurieren

Um die PCD vorzubereiten, sind 3 Schritte notwendig:

Online Verbindung zur PCD erstellen
Bevor eine Verbindung erstellt werden kann, muss PG5 „wissen“, über welches Medium/Kable auf die PCD zugegriffen werden soll. Das wird in den „Online settings“ von dem PG5 Project Tree definiert:

Als „Channel“ wird hier „S-Bus USB“ gewählt, da im Moment die IP-Konfiguration noch nicht geladen ist. Die Option PGU ist zu aktivieren.

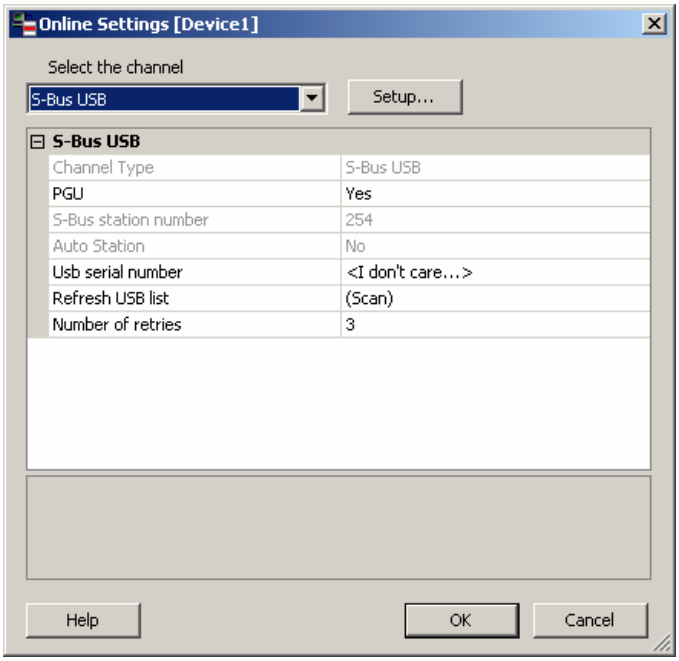

Bild 5.1.1 Online Settings

Nach diesen Einstellungen kann mit dem „Online Configurator“ überprüft werden, ob die Kommunikation funktioniert.

Hardware konfigurieren
Mit dem Device Configurator werden Einstellungen wie die IP-Adresse, Verwendung des Speichers und Aktivierung des „Run/Stop“ Switches der PCD konfiguriert. Zu finden ist der „Device Configurator“ der PCD ebenfalls im PG5 Project Tree, direkt unter den „Online Settings“.

Bitte geben Sie im Tab „TCP/IP“ eine in Ihr Netzwerk passende und noch nicht vergebene IP Adresse und Subnet-Maske ein, bevor Sie weitermachen

Um die Konfiguration auf die Steuerung zu laden, muss lediglich der Button „Download Configuration“ vom „Device Configurator“ Fenster geklickt werden oder mit einem rechts Klick auf der CPU im Project Tree. Über das Kontext Menü kann ebenfalls ein „Download Configuration“ ausgelöst werden.
Bei der Nachfrage, was auf die Steuerung geladen werden soll, muss beim ersten Download auch die Option „Memory Allocation“ gewählt werden, um den Speicher korrekt zu konfigurieren.

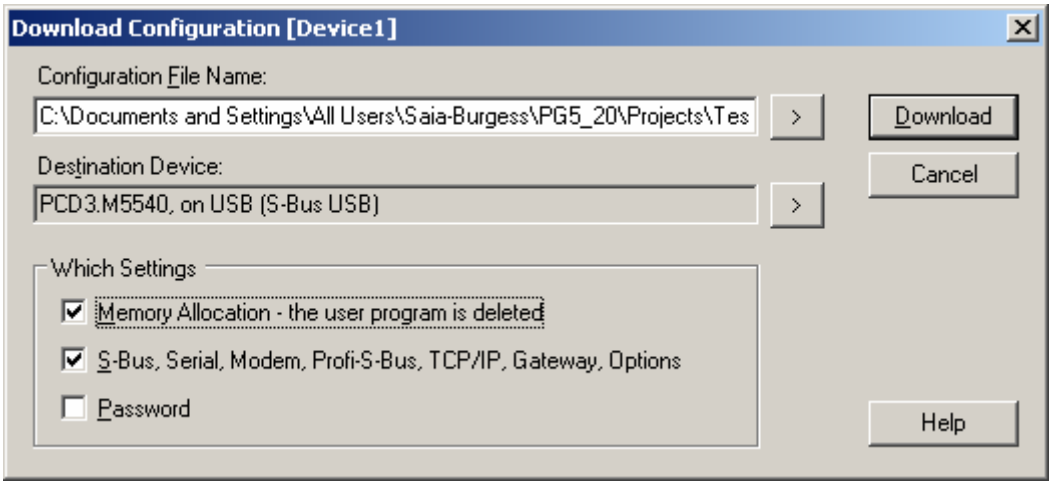

Bild 5.1.2 Download

Wenn der genaue Typ der PCD nicht bekannt ist, oder wenn die bestehende Konfiguration der Hardware nicht geändert werden soll, kann im Fenster „Hardware Settings“ auch der Button „Upload“ verwendet werden. Somit werden die momentane Konfiguration der PCD in das PG5 Projekt übernommen.

Die Hardware Settings sind auf allen PCDs, welche man nützen wir entsprechend anzupassen:
Client, S-Bus Adresse 61, IP Node 61: Dies ist der Client (Master) IP und RS
Modbus Server, SBus Adresse 62, IP Node 62. Dies ist der Server (Slave) IP
Server_RS, SBus Adresse 63, IP Node 63. Dies ist der Server (Slave) RS

5.2 Weitere Konfiguration

Das Beispiel ist so ausgelegt, dass es einen Client (Master) gibt, auf welchem sich je ein Fupla File befindet für die Kommunikationsmöglichkeiten IP und RS.

Soll nun also nur eine Modbus Kommunikation auf IP realisiert werden, ist das andere File zu deaktivieren. Wird nur eine RS Kommunikation gewünscht, ist entsprechend das IP File zu deaktivieren und das RS file zu aktivieren:

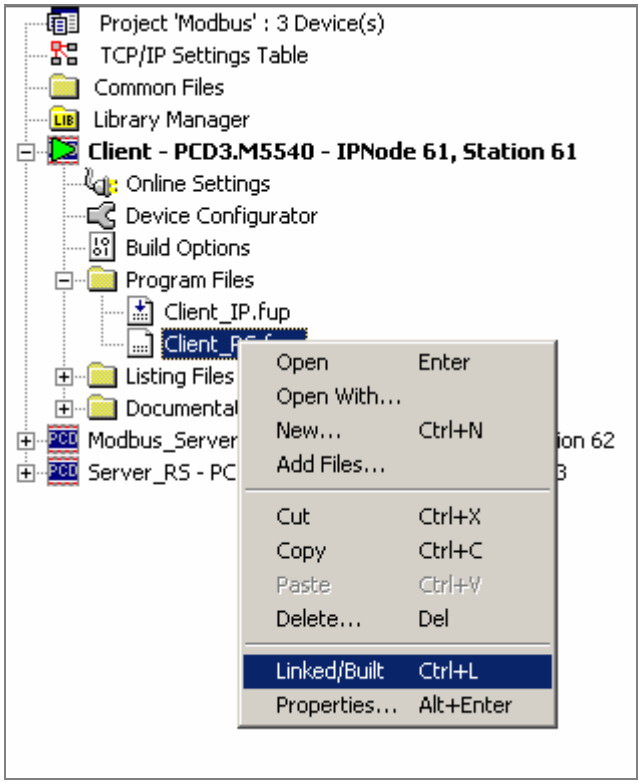

Bild 5.2.1. Linken (rechts klicken Linked aktivieren)

5.2.1 Kommunikation IP

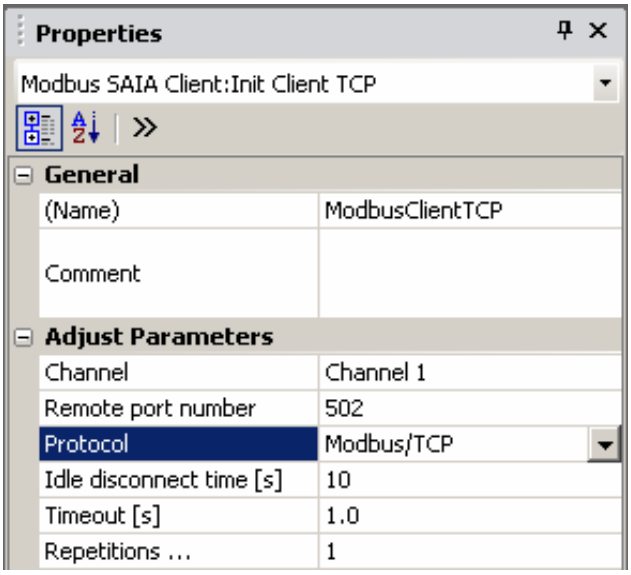

Bild 5.2.1.1 Client TCP

Im Fupla Client_IP.fup sind einige Einstellungen zu überprüfen und gemäss Ihrer Infrastruktur anzupassen. Es ist eine Modbus TCP Kommunikation konfiguriert. Wird Modbus UDP gewünscht, so kann dies in der FBOX Init Client TCP eingestellt werden. Wichtig ist, dass in diesem Fall auch der Kommunikationspartner die gleichen Einstellungen erhält.

In der FBox Def Unit Client ist die IP Adresse der Gegenstation anzupassen.

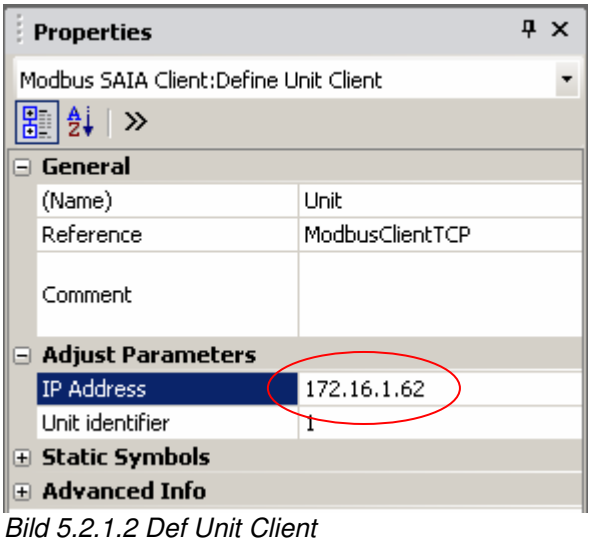

Bild 5.2.1.2 Def Unit Client

5.2.2 Kommunikation RS

Für eine serielle Kommunikation sind im Fupla Client_RS.fup die Einstellungen zu überprüfen und wenn nötig anzupassen. Die serielle Kommunikation erfolgt über den onboard RS 485 Port 2 der PCD3. Soll ein anderer Port verwendet werden, dann ist dies in der FBox Init Client RS anzupassen. Wenn die Baudrate oder das Protokoll geändert wird, so ist dies auch auf der Gegenstation anzupassen.

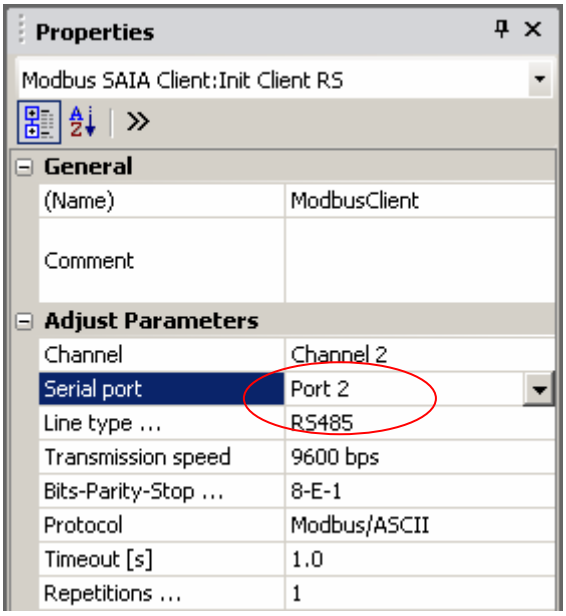

Bild 5.2.2.1 Init Client RS

Auf dem Server_RS ist ebenfalls der Kommunikationsport anzupassen, sofern nicht über den onboard Port 2 der PCD3 kommuniziert wird.

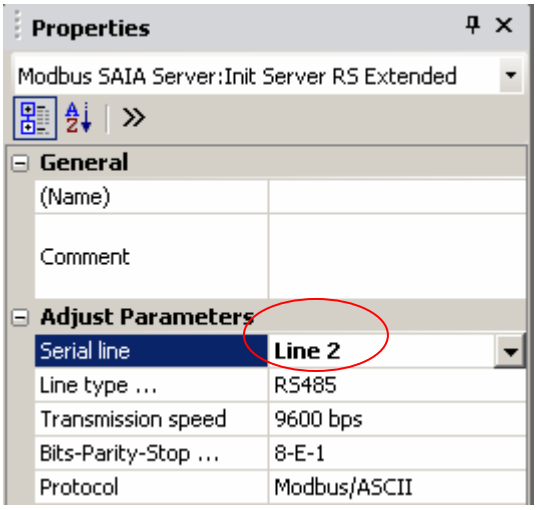

Bild 5.2.2.1 Init Server RS Extended

5.3 PCD Programmieren

Programm auf die Steuerung laden

Nun fehlt nur noch das Programmieren der PCD. Dazu muss zuerst das Programm übersetzt („Built“) werden. Dazu können Sie den „Rebuild All“ Button  verwenden.

Nachdem der „Build“ des Programms korrekt erfolgt ist, können Sie das Programm mit dem Button „Download Program“  auf die PCD laden. Somit ist die PCD vorbereitet. Je nach Einstellung Ihres PG5 wird die Steuerung automatisch in RUN gehen nach dem Download. Sollte dies nicht der Fall sein, setzen Sie die Steuerungen in RUN.

6 Programmierung der PCD

Dieses Kapitel enthält eine kurze Beschreibung der Applikation.

6.1 Client

Diese Steuerung agiert mit dem Fupla Client_IP als TCP Client und mit dem Fupla Client_RS als RS Client (RS 485).

6.1.1 Client_IP

Page 1
Auf der ersten Seite dieses Fupla Programms wird die Modbus-Kommunikation programmiert.

Die FBox Init Client TCP definiert den Kommunikations-Channel, die Remote Port Nummer, das Protokoll und das Timing Verhalten.

Die FBox Def Unit Client definiert die IP Adresse des Servers und den Unit Identifier. Das Flag Enable Def muss aktiviert sein.

Mit der FBox Read Int werden Integer Werte gelesen. Das Feld Add gibt die Adresse an, wo die gelesenen Elemente hinkopiert werden (Register: ReadIP, R 100). # definiert wie viele Register gelesen werden sollen (10 = R 100 bis R 109).

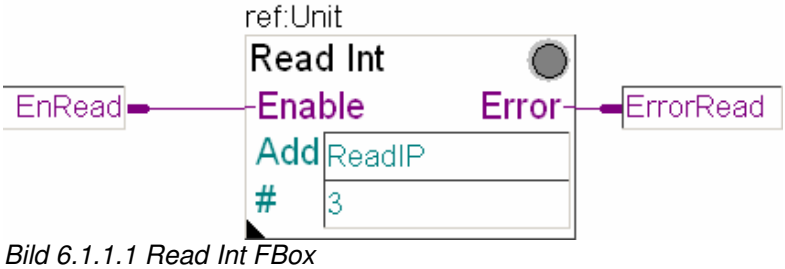

Bild 6.1.1.1 Read Int FBox

Es werden Register gelesen, 32 Bit. Die Basis Adresse bei den Modbus Holding Registern ist in unserem Fall die Adresse 1. Auf dem Modbus Server werden die zu übertragenden Daten (PCD Register 100 – 102) mit dem DataMapping auf die Modbus Holding Register 1 bis 6 gemappt. Auf dem Modbus stehen uns nur 16 Bit Holding Register zur Verfügung, während wir auf der Saia PCD Seite 32 Bit Werte empfangen / senden wollen. Daher benötigen wir pro Saia PCD Register 2 Modbus Holding Register.

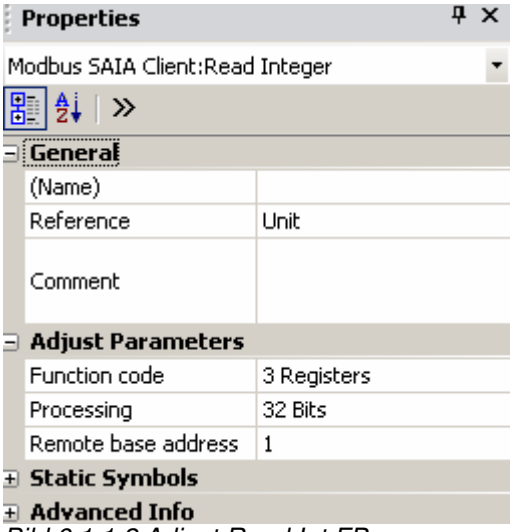

Bild 6.1.1.2 Adjust Read Int FBox

Das EnRead Flag muss hoch gesetzt werden, damit Daten gelesen werden.
Wichtig: Wenn wie in diesem Fall mehrere Register gelesen werden (3 Register in diesem Fall), dann muss ein entsprechend grosser Register Bereich für den Empfang der Daten reserviert werden (Array von 3 PCD Registern).

Mit der FBox Write Bin werden Binäre Werte geschrieben. Es werden 8 Elemente geschrieben. Diese zu übertragenden Flags sind Flag WriteBin = F 1000 und die darauf folgenden Flags (F 1000 bis F 1007).
Das EnWrite Flag muss hoch gesetzt werden um den Schreibvorgang zu starten. Auf dem Modbus werden diese Flags auf die Coils 1000 bis 1007 gemappt.

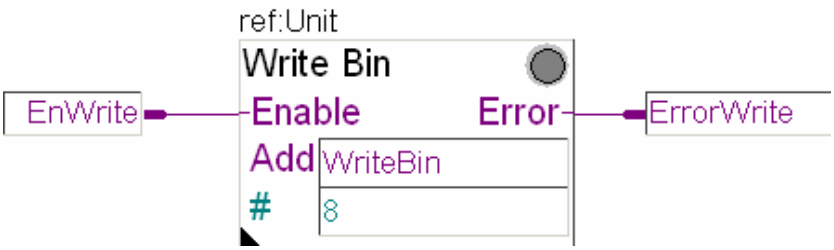

Bild 6.1.1.3 Write Bin FBox

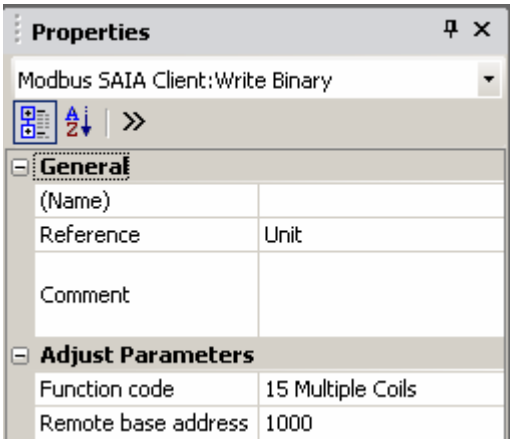

Bild 6.1.1.3 Adjust Write Bin FBox

Page 2
Mit den ersten 3 Registern, welche über den Modbus empfangen werden, erhalten wir die Uhrzeit und das Datum von der anderen Steuerung. Auf dieser Seite wird diese Uhrzeit dann auf die Hardware Uhr der Steuerung übertragen. Da wir im ersten Register die Uhrzeit in Stunden, Minuten und Sekunden erhalten, ist eine Division durch 100 nötig um dann nur noch Stunden und Minuten zu erhalten, wie es die FBox Write Clock erwartet.

Page 3
Hier wird dafür gesorgt, dass die zu übertragenden Flags immer ihren Zustand wechseln, damit man auf der Gegenseite sofort erkennen kann, ob die Kommunikation läuft.

6.1.2 Client_RS

Page 1

Auf dieser Seite erfolgt die Initialisation des Client RS. Mit der FBox Init Client RS wird ein Channel (Kommunikationschannel = virtuell) definiert, ein Port (physikalischer Port auf der PCD), die physikalische Schnittstelle, Kommunikationsparameter, Protokoll und Timing.

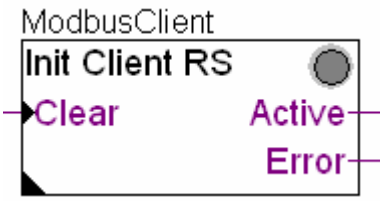

Bild 6.1.2.1 Init Client RS FBox

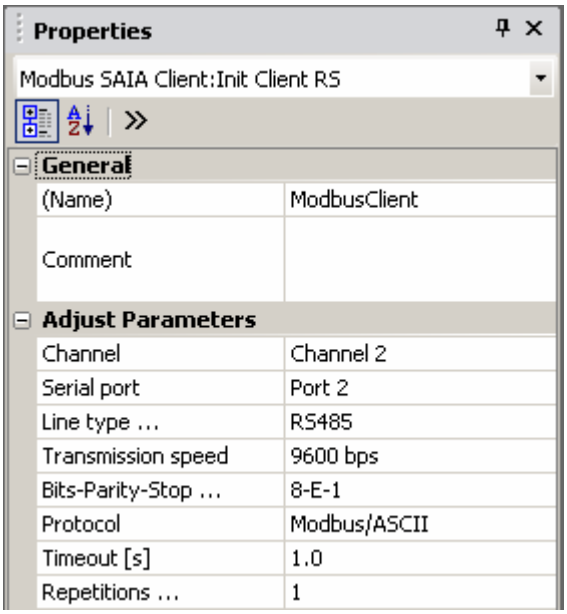

Bild 6.1.2.2 Adjust Init Client RS FBox

Die FBox Def Unit Client definiert den Unit identifier UID des Kommunikationspartners. IP Adresse muss in diesem Fall keine eingegeben werden. Das Feld IP Address kann irgendeine Adresse beinhalten.

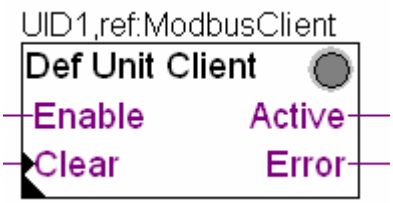

Bild 6.1.2.3 Def Unit Client FBox

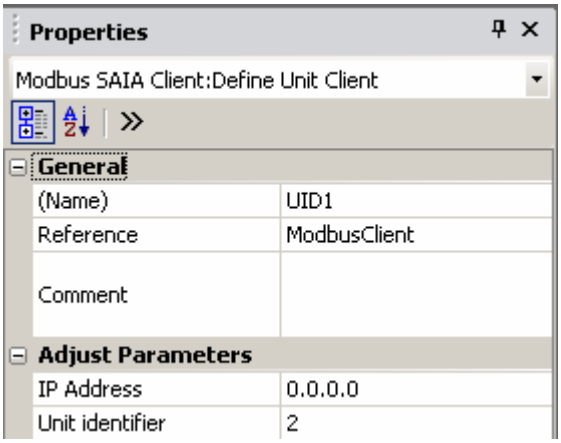

Bild 6.1.2.4 Def Unit Client Fbox

Das EnDefRS Flag muss hoch gesetzt werden.

Bild 6.1.2.4 Read Int FBox

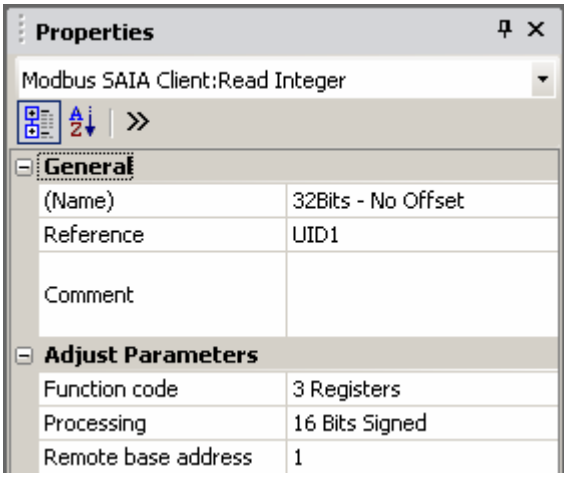

Bild 6.1.2.5 Adjust Read Int FBox

Es werden 16 Bit Signed Register gelesen von der Modbus Basisadresse 1. Diese werden auf die Adresse ReadInt R 0 bis R 9 abgelegt. Da auf dem Server das Default Mapping eingeschaltet ist, sind die PCD Register 0 bis 9 (16 Bit signed) auf die Modbus Holdingregister 1 bis 10 gemappt.

Default Mapping

	Modbus Request			PCD			
Access Type	**MediaType**	**Start Addr.**	**Range**	**MediaType**	**Start Addr.**	**Range**	**AreaType**
ReadWrite	MB_HOLDING_REG_MEDIA	1	10000	PCD_REG_MEDIA	0	10000	MB_AREA_16BITS_SIGNED
ReadWrite	MB_HOLDING_REG_MEDIA	10001	10000	PCD_REG_MEDIA	0	10000	MB_AREA_32BITS
ReadWrite	MB_HOLDING_REG_MEDIA	20001	10000	PCD_REG_MEDIA	0	10000	MB_AREA_32BITS_IEEE
ReadOnly	MB_INPUT_REG_MEDIA	1	10000	PCD_TC_MEDIA	0	10000	MB_AREA_16BITS_SIGNED
ReadOnly	MB_INPUT_REG_MEDIA	10001	10000	PCD_TC_MEDIA	0	10000	MB_AREA_32BITS
ReadWrite	MB_COILS_MEDIA	1	10000	PCD_FLAG_MEDIA	0	10000	MB_AREA_COILS
ReadOnly	MB_DISCR_INPUT_MEDIA	1	10000	PCD_IO_MEDIA	0	10000	MB_AREA_COILS

Bild 6.1.2.6 Default Mapping

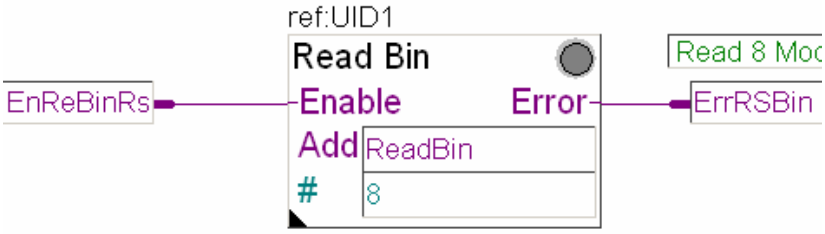

Bild 6.1.2.7 Read Bin FBox

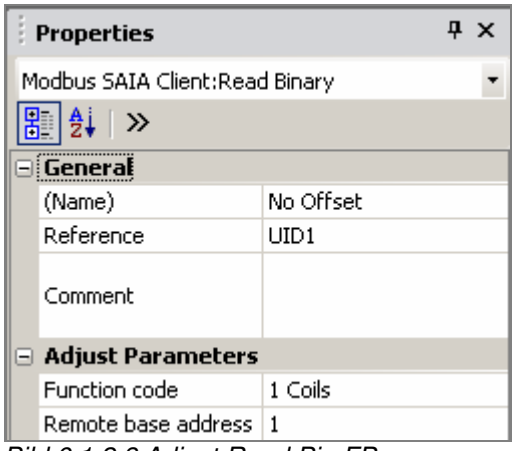

Bild 6.1.2.8 Adjust Read Bin FBox

Es werden 8 Binäre Werte gelesen vom Modbus Coil 1 an. Abgelegt werden Sie auf dem Flag ReadBin F 100 bis F 107. Da auf dem Server das Default Mapping eingeschaltet ist, entsprechen die Modbus Coils 1 bis 8 den Flags 0 bis 7 auf dem Server.

Page 2

Die Flags 100 bis 107 werden Bin zu Int gewandelt. Wenn die Übertragung erfolgreich ist, wird hier hochgezählt.
Gelesene Register werden mit der Additions- FBox addiert. Auch hier sollte sich der Wert bei aktiver Übertragung laufend verändern.

6.1.3 Modbus Server (IP)

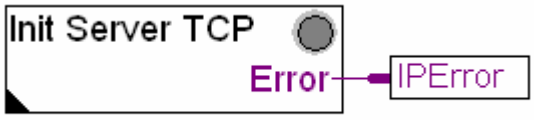

Bild 6.1.3.1 Init Server TCP FBox

Mit dieser FBox wird Port und Protokoll für die IP Verbindung definiert.

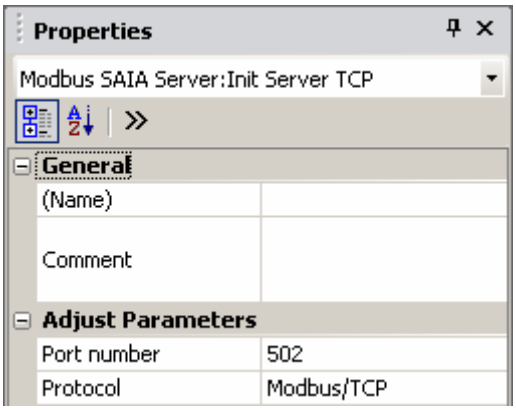

Bild 6.1.3.2 Adjust Init Server TCP FBox

Die FBox Def Unit Server definiert den Unit identifier UID. Es kann ausserdem ein Offset definiert werden, damit nicht bei 0 sonder bei 1 begonnen wird. Word swapping kann aktiviert werden wenn dies benötigt wird und die Holes können eingeschaltet werden, wenn dies gewünscht ist.

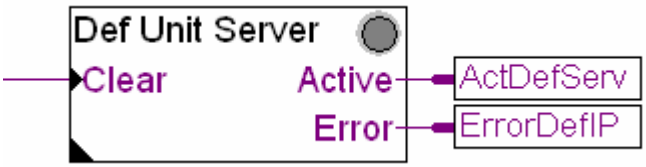

Bild 6.1.3.3 Def Unit Server TCP FBox

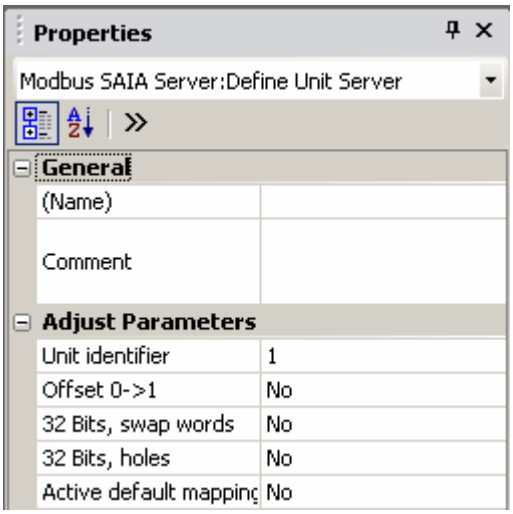

Bild 6.1.3.4 Adjust Def Unit Server TCP FBox

Mit der FBox Def Mapp Bin wird ein Mapping für den Binären Bereich erstellt.

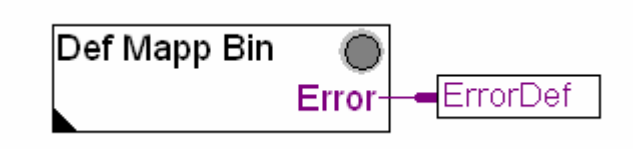

Bild 6.1.3.5 Def Mapp Bin FBox

In diesem Fall werden die Modbus Coils 1000 bis 1007 auf die PCD Flags 1000 bis 1007 gemappt.

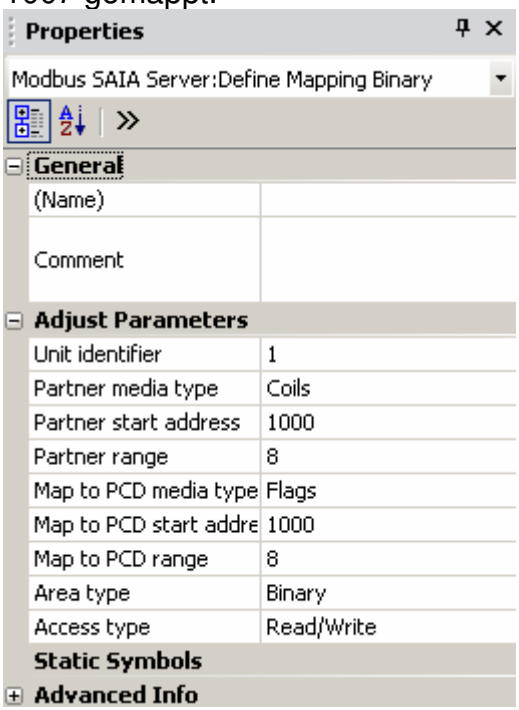

Bild 6.1.3.6 Adjust Def Mapp Bin FBox

Mit der FBox Def Mapp Int wird ein Mapping für den Integerbereich erstellt.

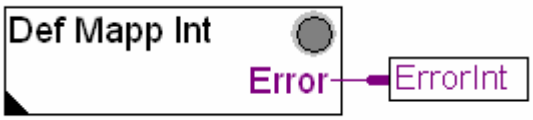

Bild 6.1.3.7 Def Mapp Int FBox

Die Modbus Holdingregister 1 bis 6 werden auf die PCD Register 100 bis 102 gemappt. Da die PCD Register 32 Bit breit sind und die Modbus Holdingregister nur 16 Bit breit, werden zwei Holding Register jeweils auf ein PCD Register gemappt.

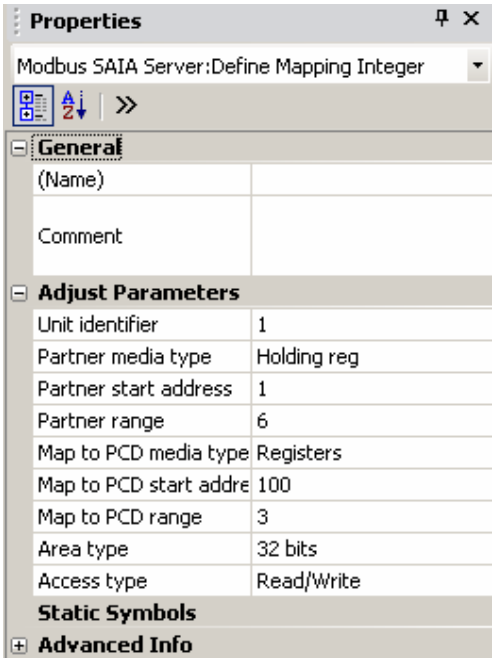

Bild 6.1.3.8 Adjust Def Mapp Int FBox

Page 2

Hier wird die Uhrzeit und das Datum der PCD gelesen. Diese wird dann zum Client übertragen. Weiter werden die empfangenen Flags Binär zu Integer gewandelt. Man sieht bei laufender Kommunikation ein Hochzählen.

6.1.4 Server_RS

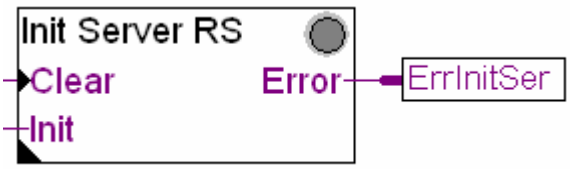

Bild 6.1.4.1 Init Server RS FBox

Dieser FBox initialisiert die Entsprechende Schnittstelle für Modbus.

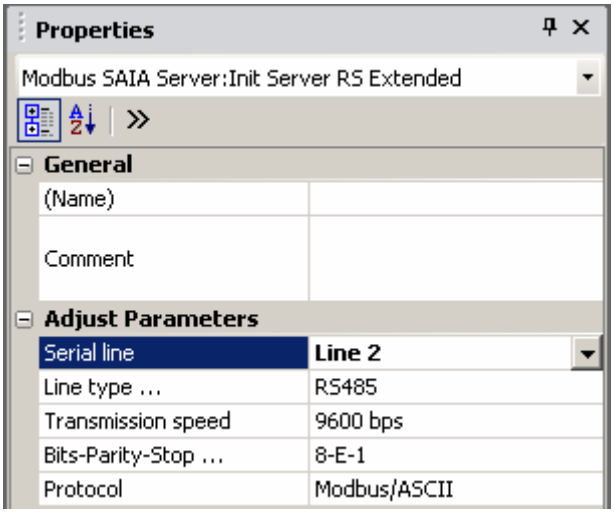

Bild 6.1.4.2 Adjust Init Server RS FBox

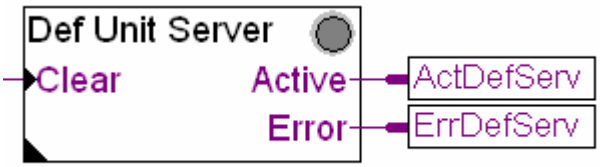

Bild 6.1.4.3 Def Unit Server FBox

Mit dieser FBox wird der Unit identifier UID eingestellt und definiert, ob ein Offset gemacht werden soll. Word swapping und holes können aktiviert werden. Das Default mapping kann ausserdem aktiviert werden.

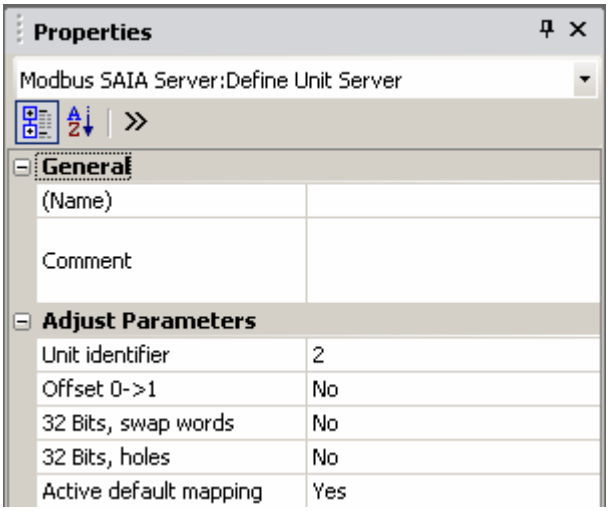

Bild 6.1.4.4 Adjust Def Unit Server FBox

Die folgenden FBoxen sorgen für ein Blinken der zu übertragenden Flags und ein Wechsel der Werte bei den zu übertragenden Register.

6.2 Übertragene Daten

6.2.1 IP Ethernet

Die Register 100 bis 102 des Server IP werden auf die Holdingregister 1 bis 6 übertragen. Auf dem Client IP werden diese dann auf die Register 100 bis 102 ausgelesen.
Die Flags 1000 bis 1007 des Clients werden auf die Coils 1000 bis 1007 gemappt. Auf dem Server werden die Coils 1000 bis 1007 dann auf die Flags 1000 bis 1007 gemappt.

6.2.2 Seriell RS 485

Es werden die Flags 0 bis 7 vom Server_RS auf die Modbus Coils 1 bis 8 übertragen. Diese werden dann auf dem Client_RS auf die Flags 100 bis 107 geschrieben.

Die Register 0 bis 9 (16 Bit) werden auf die Modbus Holding Register 1 bis 10 übertragen und auf dem Client auf die Register 0 bis 9 geschrieben.

7 Fehlersuche

Symptom	Möglicher Grund	Lösung
Default Mapping wird nicht gelesen, obwohl die FBox „Default Mapping“ = „Yes“ eingestellt ist	Sobald ein Mapping für eine UID konfiguriert ist, wird das Default Mapping deaktiviert	Alle Mappings auf der PCD für diese UID löschen.
Daten kommen nicht an, werden nicht übertragen, die FBox ist aber grün.	Die Daten kommen möglicherweise an einer anderen Adresse an.	Mapping überprüfen
FBox ist rot, Kommunikation funktioniert nicht / nur teilweise	Kommunikationsparameter stimmen nicht mit der Gegenstation überein. Port schon anderweitig konfiguriert (HW Settings). Verdrahtung nicht korrekt	Kommunikationsparameter, Konfiguration und Verdrahtung überprüfen
Ein Projekt, welches vorher mit der Engiby Library realisiert wurde, funktioniert mit der Saia Modbus Library nicht mehr.	Das Mapping stimmt möglicherweise nicht überein. Die unterschiedlichen Bezeichnungen können zu Verwechslungen führen.	Gemäss Kapitel 3.6.1 überprüfen, ob das Mapping korrekt übernommen wurde.
Die „Define unit Server“ FBox hat einen „Range Error“	Der Range ist inkorrekt definiert (z.B falsche Länge) oder von einem 32-Bit Mapping wird nur ein „halbes“ PCD Register gelesen (z.B nur HR 0 statt 0 und 1)	Range überprüfen und überprüfen ob 32 Bit Register richtig gelesen werden.

8 Referenzen

Thema	Dokument	Nr
Modbus	Manual Modbus	26/866
Divers	Saia® FAQ Manager www.sbc-support.ch/faq	-