Applikationsbeispiel

Verbinden eines Mikrocontrollers mit einer SPS mit Hilfe eines Anybus-IC für Profibus

Hinweis: Dieses Dokument ersetzt nicht die offiziellen Handbücher und Dokumentationen, die in den aktuellsten Versionen unter www.anybus.de zur Verfügung stehen.

Erstellt	Version	Name	Kommentar
12.2007	0.1	DOW	Erstentwurf
12.2007	1.0	DOW	kleinere Korrekturen
01.2009	1.1	DOW	Anpassung Initialisierung; Überarbeitung Sample-Code; Übersicht Mapping-Parameter angehängt; kleinere Korrekturen
07.2010	1.2	DOW	Byteorder CRC in Beschreibung der Modbus-Kommandos korrigiert, Anpassung der Initialisierung in Beschreibung und Sample-Code, CRC Berechnung erläutert
12.2011	1.3	DOW	Änderung Kontaktdaten Samplecode optisch überarbeitet

Inhalt:

1. AUFGABENSTELLUNG 3

2. LÖSUNG 3

2.1 SYSTEMAUFBAU 3
2.2 FUNKTIONSPRINZIP 4
2.2.1 Wie läuft die Kommunikation zwischen ABIC und Host Applikation? 4
2.2.2 Wie ist das verwendete Modbus Protokoll aufgebaut? 4
2.2.3 Übersicht über verwendbare Modbus-Kommandos 4

3. ERLÄUTERUNGEN ZUM BEISPIELCODE 6

3.1 REALISIERTE APPLIKATION 6
3.2 STRUKTUR DES BEISPIEL-CODES 7
3.3 STANDARDEINSTELLUNGEN UND DEREN VERÄNDERUNGEN 9
3.4 ÜBERBLICK ÜBER DEN INITIALISIERUNGSABLAUF 10
3.4.1 Automatische Baudratendetektion 10
3.4.2 Prüfen, ob Parameter im IC den gewünschten entsprechen 10
3.4.3 Zurücksetzen der gespeicherten Parameter im ABIC 10
3.4.4 Durchführen eines Hardwareselbsttests 10
3.4.5 Parameter im ABIC auf definierte Werte setzen 10
3.4.6 ABIC auf Normalbetrieb schalten 11
3.5 ABLAUF DES DATENAUSTAUSCHS 11
3.6 MODBUS RTU CRC-BERECHNUNG 11

4. EINBINDEN DES ANYBUS-IC IN EIN STEP7-PROJEKT 12

5. ANHANG 17

5.1 ÜBERBLICK ÜBER MAPPING-PARAMETER 17
5.2 WEITERE DOKUMENTE 17

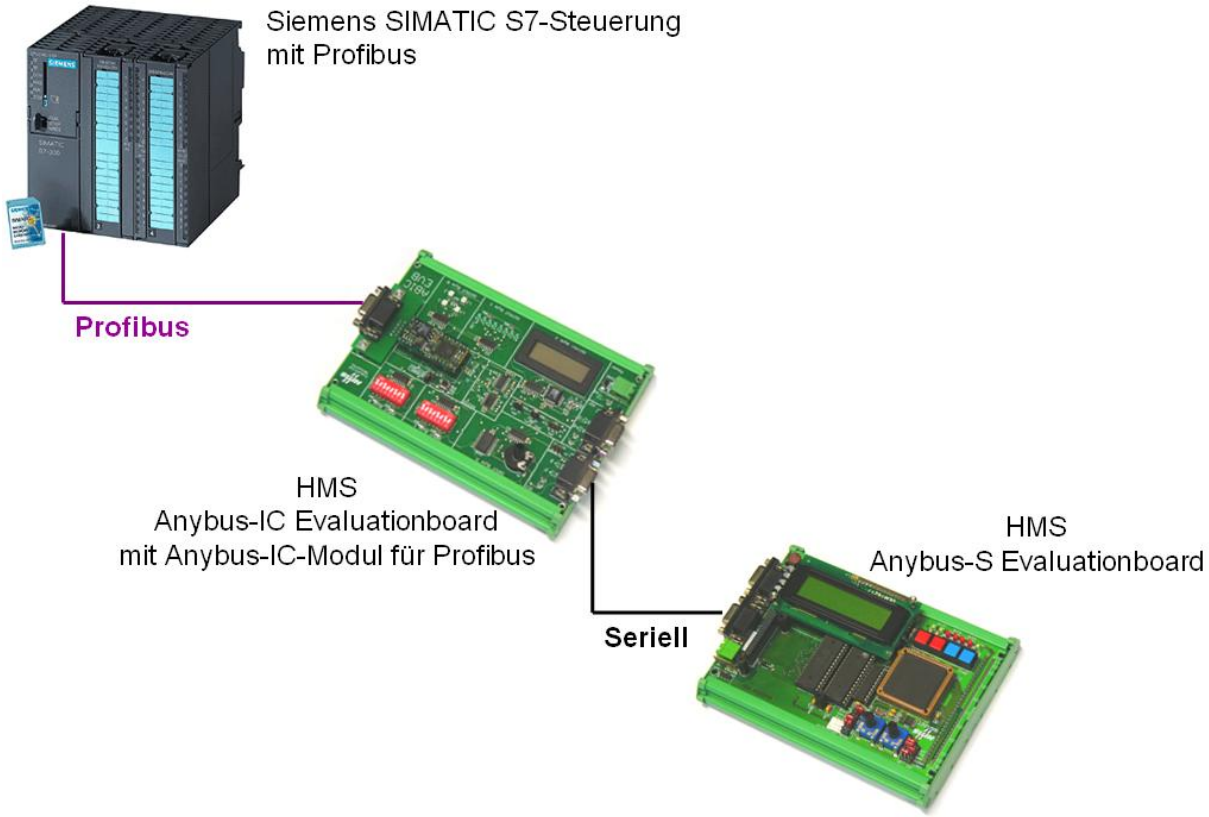
1. Aufgabenstellung

Herstellern und Entwicklern von Automatisierungsgeräten stellt sich häufig die Frage, wie sie möglichst einfach eine Anbindung an ein Automatisierungssystem – z.B. eine Siemens S7 mit Profibus Master - realisieren.

2. Lösung

HMS stellt mit seinen Anybus-Produktfamilien verschiedene Möglichkeiten zur Anbindung von Geräten an die gebräuchlichsten Feldbus- und Industrial Ethernet-Systeme zur Verfügung.
Gerade bei Mikrocontrollern mit geringen Hardware-Ressourcen bietet sich das Anybus-IC an. Zur Kommunikation zwischen Anybus-IC (nachfolgend auch ABIC genannt) und dem Mikrocontroller wird lediglich eine UART-Schnittstelle benötigt, über die nahezu jeder Mikrocontroller verfügt.
Das nachfolgend erläuterte Applikationsbeispiel beschreibt, wie eine Host-Applikation (in unserem Beispiel ein Evaluationboard für die Anybus-S-Module, nachfolgend ABS-Evalboard genannt) über das ABIC Daten mit der übergeordnete SPS austauschen kann. Das ABIC wird dabei auf einem Evaluationboard für die ABIC-Module betrieben und kommuniziert über eine serielle Leitung mit dem Controller des ABS-Evalboards.
Die beiden Evalboards können separat bei HMS erworben werden.
Für dieses Beispiel wurde die Variante mit Profibus gewählt.

2.1 Systemaufbau



2.2 Funktionsprinzip

2.2.1 Wie läuft die Kommunikation zwischen ABIC und Host Applikation?

Die Host-Applikation kommuniziert über das so genannte SCI-Interface mit dem ABIC. Dieses ist ein serielles asynchrones Interface. Als Protokoll wird Modbus RTU verwendet.
Im vorliegenden Beispiel wird das serielle Signal über RS232-Treiber auf die entsprechenden Pegel gebracht. In einer Kundenapplikation ist dies nicht notwendig. Dort kann die Kommunikation direkt mit dem UART des Mikrocontrollers auf TTL-Pegeln stattfinden.

2.2.2 Wie ist das verwendete Modbus Protokoll aufgebaut?

Eine vollständige Spezifikation des Modbus-Protokolls kann von der Internet-Seite der Modbus IDA (www.mobus.org) geladen werden.
An dieser Stelle seien lediglich einige Rahmeninformationen zusammengefasst.
Modbus ist ein Master-Slave-basiertes Netzwerk. Hierbei stellt ein Slave Informationen zur Verfügung, die von einem Master gelesen werden, bzw. nimmt Informationen entgegen, die von einem Master geschrieben werden. Dazu werden verschiedene Schreib- und Lesefunktionen anhand so genannter Funktionscodes unterschieden.
Die Zugriffe erfolgen registerweise, wobei ein Register aus 2 Bytes besteht.
Jedes intakte Telegramm des Masters muss vom Slave beantwortet werden.
Die einzelnen Telegramme haben eine einheitliche Struktur:

Telegrammteil	**Länge**
Slave Adresse	1 Byte
PDU (protocol data unit)	mindestens 2 Byte
CRC Checksumme	2 Byte

Der Aufbau der PDU ist abhängig vom verwendeten Funktionscode.

2.2.3 Übersicht über verwendbare Modbus-Kommandos

Hier wird der unterschiedliche Aufbau der Telegramme für die Modbus-Funktionscodes, die vom ABIC unterstützt werden, aufgezeigt.

2.2.3.1 03 (0x03) Read Multiple Registers

Dieser Funktionscode heißt laut Definition der Modbus IDA „Read Holding Register“ und ist dazu gedacht, rücklesbare Ausgänge zurückzulesen.

Master-Telegramm

0x01	Slave Adresse
0x03	Funktionscode
0x00	Startadresse High
0x00	Startadresse Low
0x00	Anzahl Register High
N	Anzahl Register Low
0xXX	CRC Low
0xXX	CRC High

Slave-Antwort

0x01	Slave Adresse
0x03	Funktionscode
2*N	Anzahl Datenbytes
0x23	Datenregister 1 High
0xd3	Datenregister 1 Low
0xff	Datenregister 2 High
0x3e	Datenregister 2 Low
...	…
0x79	Datenregister N High
0x6a	Datenregister N Low
0xXX	CRC Low
0xXX	CRC High

2.2.3.2 04 (0x04) Read Input Registers

Dieser Funktionscode ist dazu gedacht, Eingangsdaten eines Slaves einzulesen.

Master-Telegramm

0x01	Slave Adresse
0x04	Funktionscode
0x00	Startadresse High
0x00	Startadresse Low
0x00	Anzahl Register High
N	Anzahl Register Low
0xXX	CRC Low
0xXX	CRC High

Slave-Antwort

0x01	Slave Adresse
0x04	Funktionscode
2*N	Anzahl Datenbytes
0x23	Datenregister 1 High
0xd3	Datenregister 1 Low
0xff	Datenregister 2 High
0x3e	Datenregister 2 Low
...	…
0x79	Datenregister N High
0x6a	Datenregister N Low
0xXX	CRC Low
0xXX	CRC High

2.2.3.3 06 (0x06) Write Single Register

Dieser Funktionscode schreibt ein einzelnes Register an die angegebene Adresse.

Master-Telegramm

0x01	Slave Adresse
0x06	Funktionscode
0x12	Registeradresse High
0x56	Registeradresse Low
0xfe	Registerwert High
0xc2	Registerwert Low
0xXX	CRC Low
0xXX	CRC High

Slave-Antwort

0x01	Slave Adresse
0x06	Funktionscode
0x12	Registeradresse High
0x56	Registeradresse Low
0xfe	Registerwert High
0xc2	Registerwert Low
0xXX	CRC Low
0xXX	CRC High

2.2.3.4 16 (0x10) Write Multiple Registers

Dieser Funktionscode schreibt mehrere Register ab einer übergebenen Startadresse.

Master-Telegramm

0x01	Slave Adresse
0x10	Funktionscode
0x00	Startadresse High
0x00	Startadresse Low
0x00	Anzahl Register High
N	Anzahl Register Low
2*N	Anzahl Datenbytes
0x23	Datenregister 1 High
0xd3	Datenregister 1 Low
0xff	Datenregister 2 High
0x3e	Datenregister 2 Low
...	…
0x79	Datenregister n High
0x6a	Datenregister n Low
0xXX	CRC Low
0xXX	CRC High

Slave-Antwort

0x01	Slave Adresse
0x10	Funktionscode
0x00	Startadresse High
0x00	Startadresse Low
0x00	Anzahl Register High
N	Anzahl Register Low
0xXX	CRC Low
0xXX	CRC High

2.2.3.5 23 (0x17) Read/Write Multiple Registers

Dieser Funktionscode schreibt und liest mehrere Bytes ab einer einstellbaren Startadressen.

Master-Telegramm

0x01	Slave Adresse
0x17	Funktionscode
0x00	Lese-Startadresse High
0x00	Lese-Startadresse Low
0x00	Lese-Registerzahl High
N1	Lese-Registerzahl Low
0x00	Schreib-Startadresse High
0x00	Schreib-Startadresse Low
0x00	Schreib-Registerzahl High
N2	Schreib-Registerzahl Low
2*N2	Anzahl Schreibbytes
0x23	Datenregister 1 High
0xd3	Datenregister 1 Low
0xff	Datenregister 2 High
0x3e	Datenregister 2 Low
...	…
0x79	Datenregister N2 High
0x6a	Datenregister N2 Low
0xXX	CRC Low
0xXX	CRC High

Slave-Antwort

0x01	Slave Adresse
0x17	Funktionscode
2*N1	Anzahl Lesebytes
0x23	Datenregister 1 High
0xd3	Datenregister 1 Low
0xff	Datenregister 2 High
0x3e	Datenregister 2 Low
...	…
0x79	Datenregister N High
0x6a	Datenregister N Low
0xXX	CRC Low
0xXX	CRC High

3. Erläuterungen zum Beispielcode

Der Beispielcode ist für das ABS-Evalboard in Verbindung mit einem Anybus-IC-Evaluationboard entwickelt worden.

Der Beispielcode beinhaltet folgende Funktionalität:

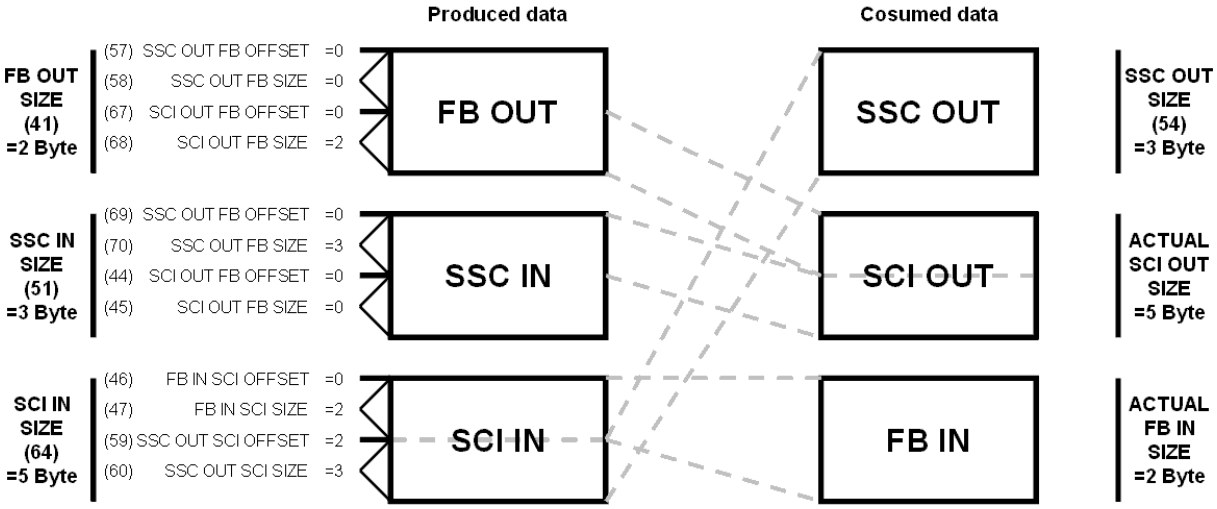
- Kommunikation über das SCI-Interface mit dem ABIC
- Berechnung der CRC Prüfsummen
- Unterstützung für die automatische Baudratendetektion des ABIC
- Zurücksetzen des ABIC auf Werkseinstellungen
- Neu-Parametrieren des ABIC
- Starten des ABIC
- Datenaustausch mit dem ABIC

Über Compilerschalter können die einzelnen Teile des Codes aktiviert oder deaktiviert werden. Dazu werden die #defines in der Datei „abic.h“ verwendet (siehe Kapitel 3.3). Nach einer Änderung muss der Code neu kompiliert werden.

3.1 Realisierte Applikation

Das ABS-Evalboard verfügt über eine serielle Schnittstelle, die über die Applikationssoftware frei programmierbar ist. Außerdem steht ein zweizeiliges Display zur Verfügung.
Um einen Controller mit geringen Hardware-Ressourcen zu simulieren, wird davon ausgegangen, dass dieser außer dem UART keine weiteren Portpin als I/O zur Verfügung stellt. Daher wird die Möglichkeit des ABIC genutzt, auch Schieberegisterein- und ausgänge an die Host-Applikation durchreichen zu können.

Im Detail sieht das Daten-Mapping des ABIC wie folgt aus:



Über die "Config Bits" (Parameter #8) wird festgelegt, dass die Node ID über das SCI-Interface festgelegt wird (FBNP=1 und NA=1). Weiterhin wird der Feldbusstatus nur über das SCI-Interface auslesbar sein (FBLP=1). Die Baudrate wird bei Profibus immer über den Feldbus festgelegt.
Im normalen Datenaustausch wird das Lese-Telegramm, das der Mikrocontroller zum ABIC schickt, in der oberen Zeile auf dem LCD Display dargestellt. Die Antwort dazu wird in der unteren Zeile dargestellt:

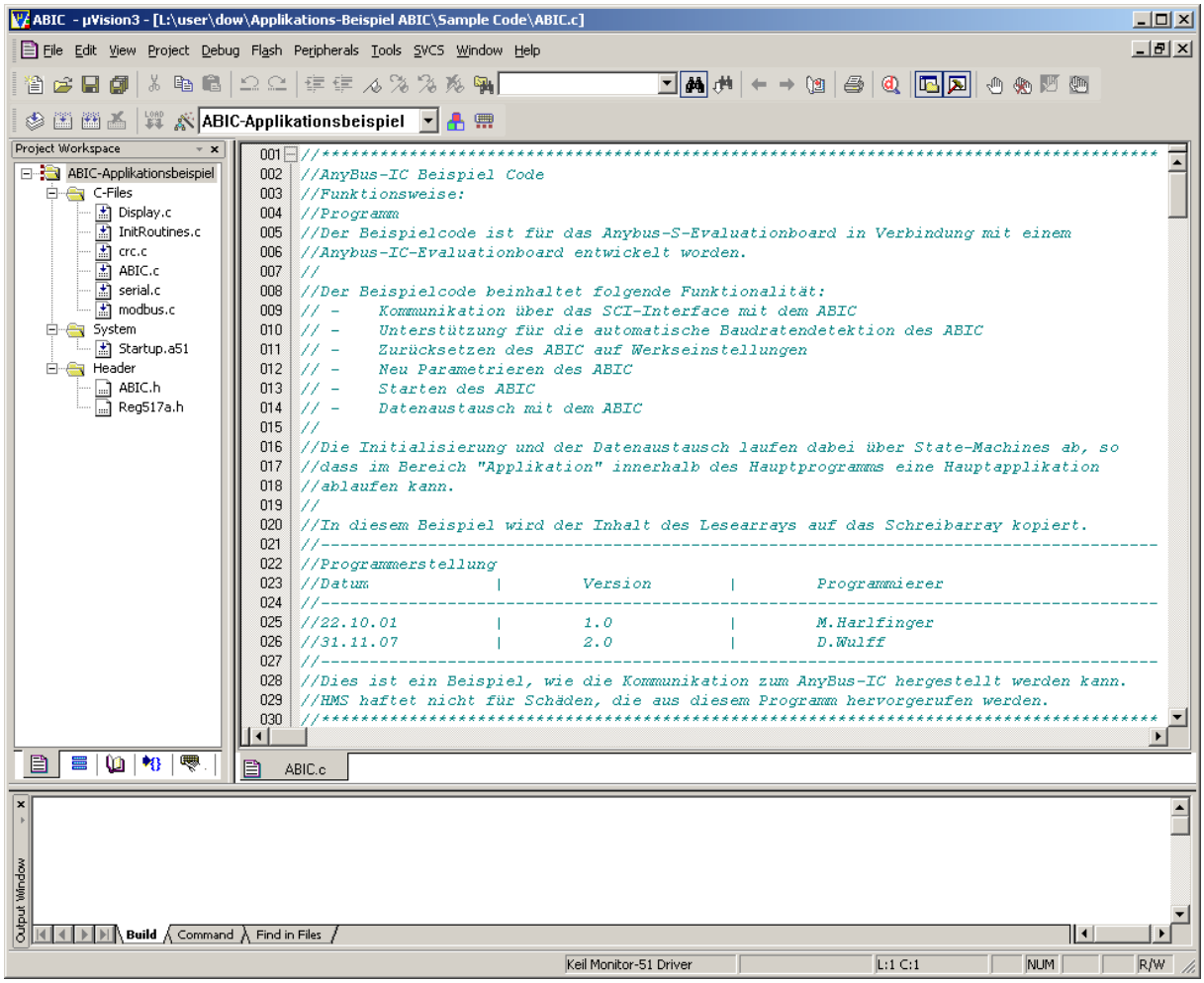
Wird der „PROG“-Knopf des ABS-Evalboards gedrückt, wird Selbiges für die Schreibtelegramme getan.
Anmerkung: Wenn die darzustellenden Telegramme länger als 8 Byte sind wird nur das Nutzdatenpaket angezeigt, da das Display nur über 16 Stellen pro Zeile verfügt.

3.2 Struktur des Beispiel-Codes

Der Beispielcode wurde in Form eines Keil-C51-Projektes für die Keil-Entwicklungsumgebung µ-Vision 2 bzw. 3 angelegt.
Diese Entwicklungsumgebung ist als Demoversion auch im Lieferumfang eines ABS-Evalboards enthalten. Deren Beschränkung auf 2 kB Code macht es leider unmöglich, den hier vorliegenden Code zu kompilieren. Daher ist er in diesem Fall nur als Beispiel für eine mögliche Umsetzung zu sehen.

Der Beispielcode sollte in ein Verzeichnis kopiert werden, auf das der Anwender Schreibrechte hat.

Durch Doppelklicken auf die Datei ABIC.uv2 wird das Projekt geladen; es sollte sich dann folgendes Bild ergeben:



Die Hauptroutine **`main()`** befindet sich in dem Modul ABIC.c.

Innerhalb dieser Hauptroutine wird das Modul in der Initialisierungs-State-Machine initialisiert und anschließend der Datenaustausch über die entsprechende State-Machine gestartet.

3.3 Standardeinstellungen und deren Veränderungen

Die nachfolgend genannten Standardeinstellungen befinden sich in der **Datei *abic.h***:

```
//*************** Modbus-Funktionscodes ***************//
#define READ_MULT_REGISTER      0x03
#define READ_INPUT_REGISTER     0x04
#define WRITE_SINGLE_REGISTER   0x06
#define WRITE_MULT_REGISTER     0x10
#define READ_WRITE_REGISTERS    0x17

//*************** Programm-Parameter ***************//
#define INIT                    1
          // Initialisierung über die Applikation. Andernfalls muss
          // Initialisierung über MIF Interface vorgenommen werden.

#define HWSELFTEST         1
          // Hardwaretest durchführen

#define RESET_PARAMETER    1
          // ABIC beim Start und vor der Parametrierung auf
          // Standardwerte setzen

#define SET_PARAMETER      1
          // Parametrierung des ABIC über die Applikation
          // vornehmen. Andernfalls muss Parametrierung über das
          // MIF Interface vorgenommen werden.

#define READ                    1
          // Daten werden über SCI vom ABIC gelesen

#define WRITE              1
          // Daten werden über SCI zum ABIC geschrieben


#define INITTIME      20000
#define READTIME      40000
#define WRITETIME     40000
          // Definition der Timeouts für Modbuskommunikation.
```

3.4 Überblick über den Initialisierungsablauf

Die Initialisierung des ABIC läuft in folgenden Schritten ab:

1. automatische Baudratendetektion
2. Prüfen, ob Parameter im IC den gewünschten entsprechen
3. Zurücksetzen der gespeicherten Parameter im ABIC
4. erneute automatische Baudratendetektion
5. Durchführen eines Hardwareselbsttests
6. Parameter im ABIC auf definierte Werte setzen
7. ABIC auf Normalbetrieb schalten

Die einzelnen Schritte werden im Folgenden näher erläutert.

3.4.1 Automatische Baudratendetektion

Die Automatische Baudratendetektion innerhalb der Funktion **`autobaud()`** läuft in folgenden Schritten ab:

1. Absenden von „Read Multiple Register“ auf Adresse 0x5001 (Modul Status)
2. Warten auf Antwort
3. Prüfen der Antwort
4. a) Antwort vollständig und CRC ok: Baudrate wurde vom ABIC erkannt.
   b) Antwort unvollständig oder CRC fehlerhaft: Einstieg bei 1.

3.4.2 Prüfen, ob Parameter im IC den gewünschten entsprechen

In der Funktion **`CheckParameter()`** werden die Parameter des ABIC, die im Array „parameters“ mit Registeradresse und Sollwert hinterlegt sind, mit mehreren Lesezugriffen mit den im ABIC Gespeicherten abgeglichen.
Die Letzte Zeile des Arrays ist komplett mit 0x00 vorbelegt, um ein Erkennen des Array-Endes zu ermöglichen.

3.4.3 Zurücksetzen der gespeicherten Parameter im ABIC

Wenn die Prüfung der Parameter ergibt, dass diese nicht den Sollwerten entsprechen, werden über die Funktion **`ResetParameter()`** alle Parameter auf Werkseinstellungen gesetzt.
Dies erfolgt, indem der Modul Mode (Register 0x5001) auf 4 gesetzt und im Anschluss 200ms bis zum Abschluss des Rücksetzens gewartet wird.

3.4.4 Durchführen eines Hardwareselbsttests

Der Hardwareselbsttest wird gestartet, indem der Modul Mode (Register 0x5001) auf 5 gesetzt wird (**`SetModuleMode(0x05)`**). Der Test ist abgeschlossen, wenn der Modul Mode wieder auf 0 (Init State) steht.
Das Ergebnis des Tests wird mittels Überprüfung der drei höchstwertigen Bits des Module Status Registers (Register 0x5002) ausgewertet. Der Test verlief fehlerfrei, wenn keines dieser Bits gesetzt ist.

3.4.5 Parameter im ABIC auf definierte Werte setzen

Mit der Funktion **`SetParameter()`** werden die Parameter des ABIC, die im Array „parameters“ mit Registeradresse und Sollwert hinterlegt sind, mit mehreren Schreibzugriffen zum ABIC übertragen.
Die Parameter werden vom ABIC nur dann in den nichtflüchtigen Speicher geschrieben, wenn sie von den dort gespeicherten abweichen.
Die letzte Zeile des Arrays ist komplett mit 0x00 vorbelegt, um ein Erkennen des Array-Endes zu ermöglichen.

3.4.6 ABIC auf Normalbetrieb schalten

Das Umschalten geschieht durch einen Schreibzugriff auf das Register Module Mode (Register 0x5001). Dieses wird auf 1 (Normal Operation) gesetzt (**`SetModuleMode(0x01)`**).

Weitere Informationen können dem Quelltext entnommen werden.

3.5 Ablauf des Datenaustauschs

Der Datenaustausch findet in folgenden Schritten statt:

1. „Read Input Registers ab Adresse 0x1000 für 3 Register“ senden
2. Warten auf Antwort
3. Prüfen der Antwort (bei Fehler Meldung auf Display)
4. Daten aus korrekter Antwort in Eingangsarray schreiben
5. Daten aus Ausgangsarray in Datenpuffer kopieren
6. „Write Multiple Registers ab Adresse 0x0000 für 3 Register“ aus Datenpuffer senden
7. Warten auf Antwort
8. Prüfen der Antwort (bei Fehler Meldung auf Display)

Im vorliegenden Beispielcode wird bei jedem Programmdurchlauf das Eingangsarray auf das Ausgangsarray kopiert.

3.6 Modbus RTU CRC-Berechnung

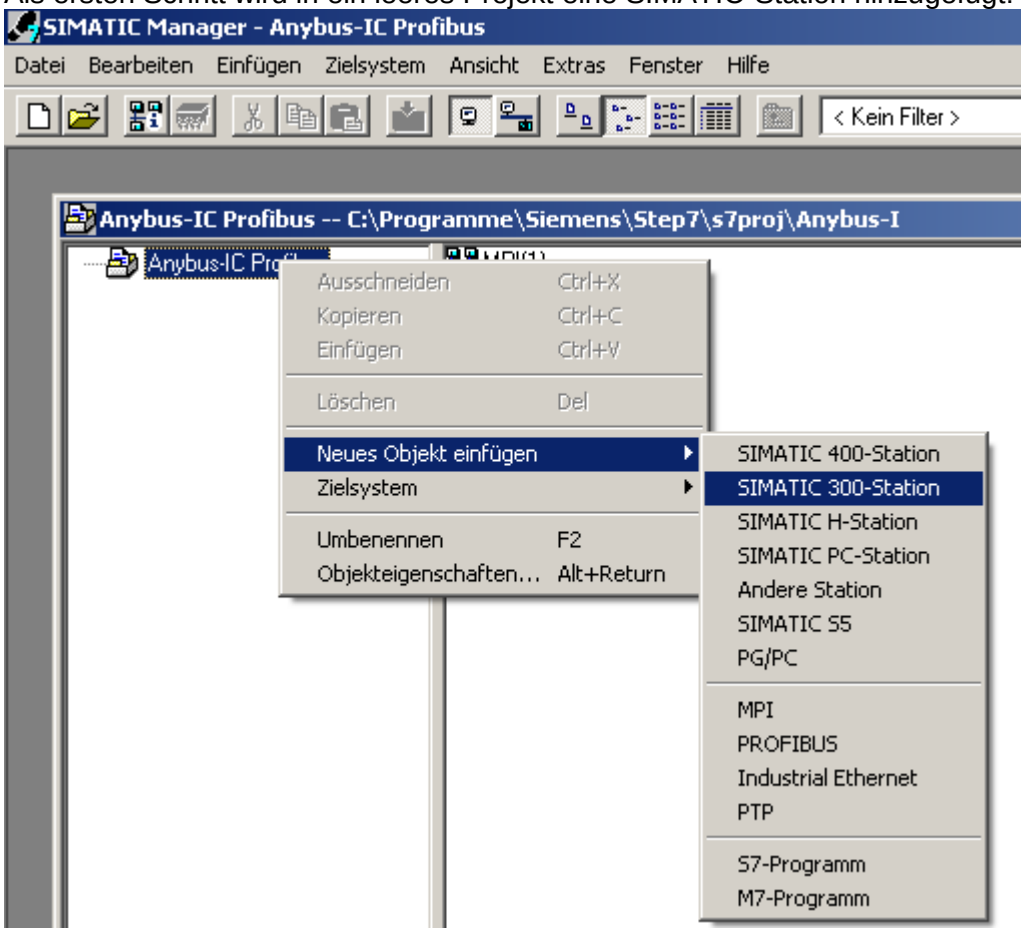
Die CRC-Prüfsumme wird innerhalb der Funktion „GenerateCrc“, die Bestandteil der Datei crc.c ist, berechnet. Es handelt sich dabei um eine CRC16 Prüfsumme nach Modbus-Spezifikation. Diese ist im Dokument „MODBUS over Serial Line“ – „Specification & Implementation guide“ der Modbus IDA beschrieben und verwendet das Generatorpolynom $1+x_2+x_{15}+x_{16}$ (=> Berechnungspolynom 1010 0000 0000 0001b = 0xA001h)
Das Vorgehen ist wie folgt:

1. Laden eines 16-Bit-Registers mit 0xFFFFh. Dies ist das CRC-Register (iCRCReg).
2. Exklusiv-Oder des ersten Bytes der Nachricht mit dem Low-Byte des CRC-Registers. (Bei der in crc.c gewählten Pointer-Schreibweise muss die Code-Zeile je nach verwendeter IDE verändert werden. Dazu bitte den Kommentar beachten)
3. Das CRC-Register ein Bit in Richtung LSB verschieben (MSB wird mit null aufgefüllt). Dabei das LSB extrahieren (bCarryFlag) und untersuchen.
4. Wenn LSB=0: Schritt 3 wiederholen
   Wenn LSB=1: CRC-Register mit dem Ergebnis einer Exklusiv-Oder-Verknüpfung von CRC-Register mit Polynomwert 0xA001h (1010 0000 0000 0001) laden
5. Wiederholen von Schritt 3 und 4 bis 8 Verschiebungen durchgeführt wurden. Damit ist die Verarbeitung eines Bytes abgeschlossen
6. Wiederholen von Schritt 2 bis 5 für das nächste Byte der Nachricht bis alle Bytes bearbeitet wurden
7. Nachdem alle Bytes der Nachricht bearbeitet wurden, ist der Inhalt des CRC-Registers der CRC-Wert
8. High- und Low-Byte des CRC-Wertes werden vor der Rückgabe getauscht, da das Low-Byte zuerst übertragen werden muss.

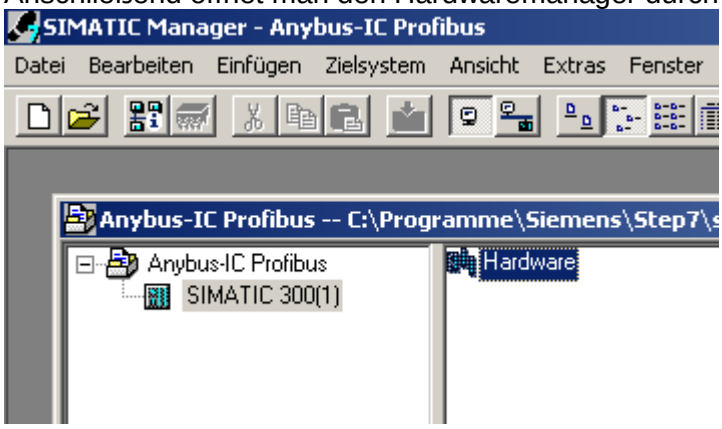
4. Einbinden des Anybus-IC in ein Step7-Projekt

Um das ABIC in ein Projekt für eine Siemens-S7-Steuerung einzufügen, benötigt man einen PC mit einem installierten SIMATIC-Manager.

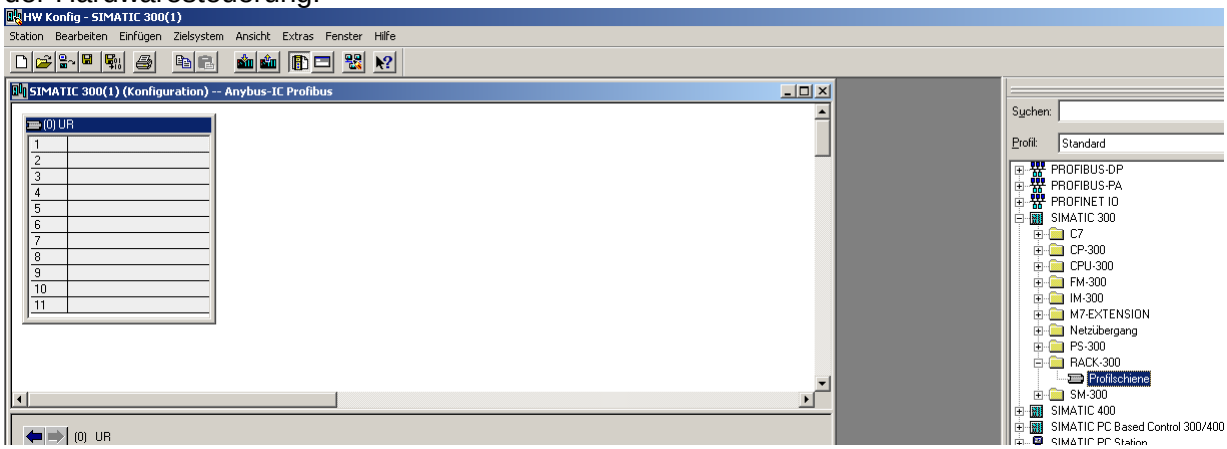
Als ersten Schritt wird in ein leeres Projekt eine SIMATIC-Station hinzugefügt:



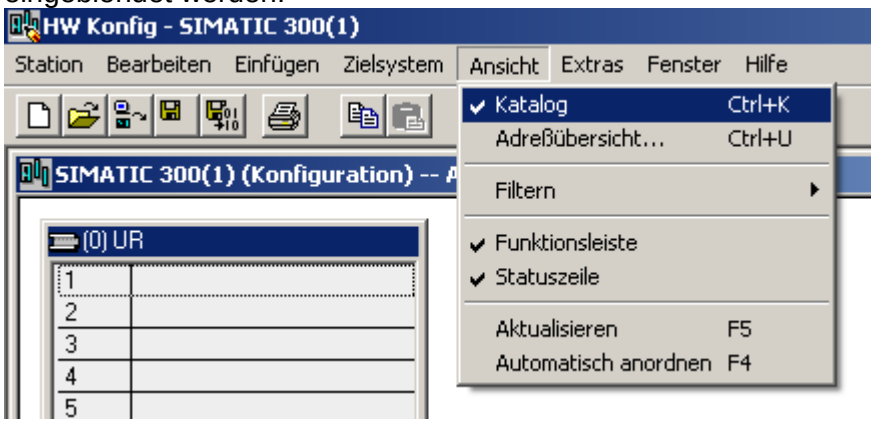
Anschließend öffnet man den Hardwaremanager durch einen Doppelklick auf ***Hardware***:



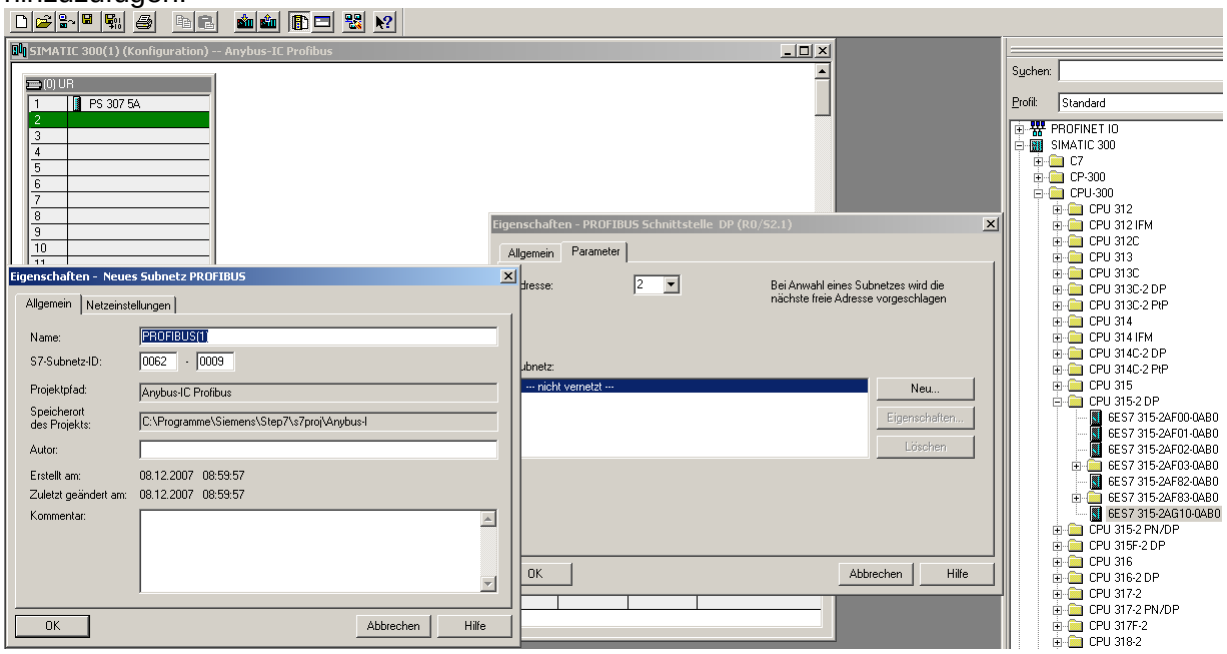
Im Hardwaremanager wird zuerst eine Profilschiene durch einen Doppelklick auf das Symbol im Hardwarekatalog eingefügt. Dies ermöglicht den Aufbau der Steuerung des Netzwerkes in der Hardwaresteuerung:



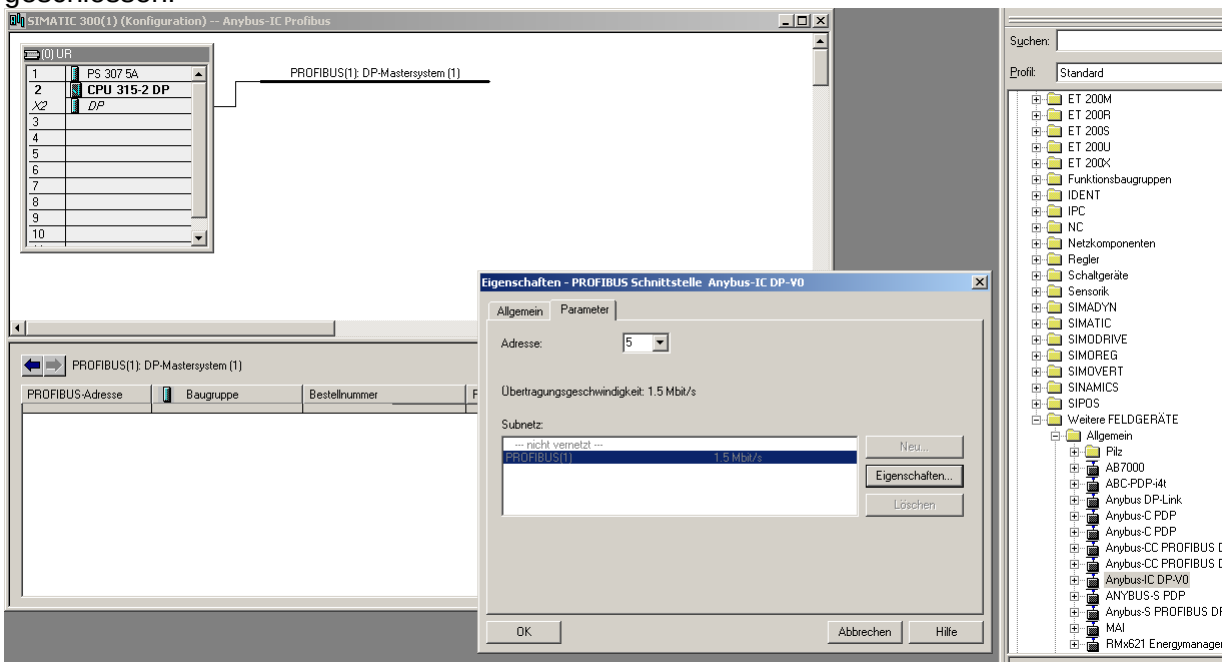
Falls der Hardwarekatalog nicht eingeblendet ist, kann er über einen Klick im Menü ***Ansicht*** eingeblendet werden.



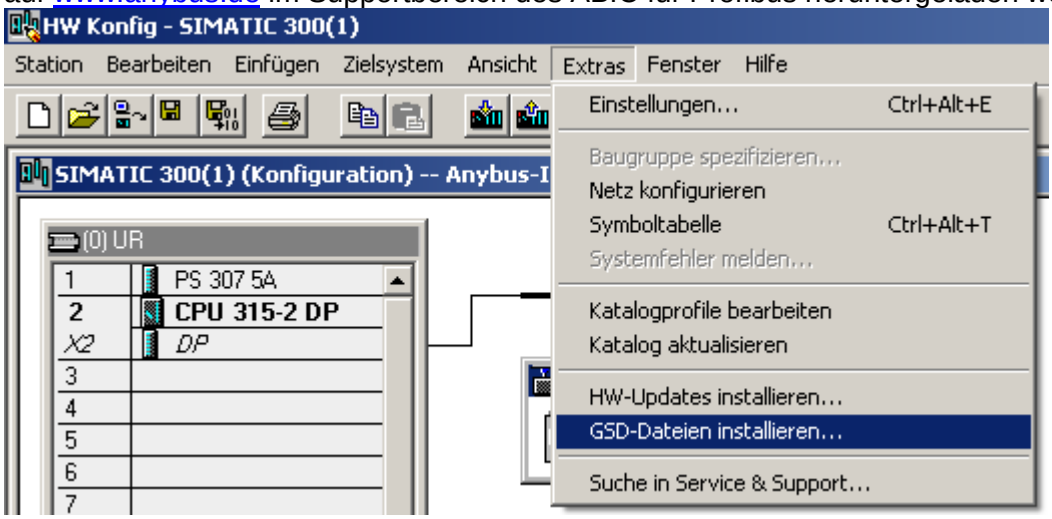
Anschließend wird durch einen Doppelklick die verwendete Steuerung eingefügt. Dabei öffnet sich automatisch das Konfigurationsmenü zur Steuerung. Im Bereich ***Subnetz*** gelangt man durch Klick auf den Button ***Neu*** in das Profibus-Konfigurationsmenü. Nachdem man die Baudrate eingestellt hat, wird auf ***OK*** geklickt. Im Eigenschaften-Menü der Profibus-Schnittstelle kann außerdem die Profibus-Adresse des Masters eingestellt werden. Nach dem erneuten Klick auf den Button ***OK*** ist das Netzwerk vorbereitet, um Slaves hinzuzufügen.



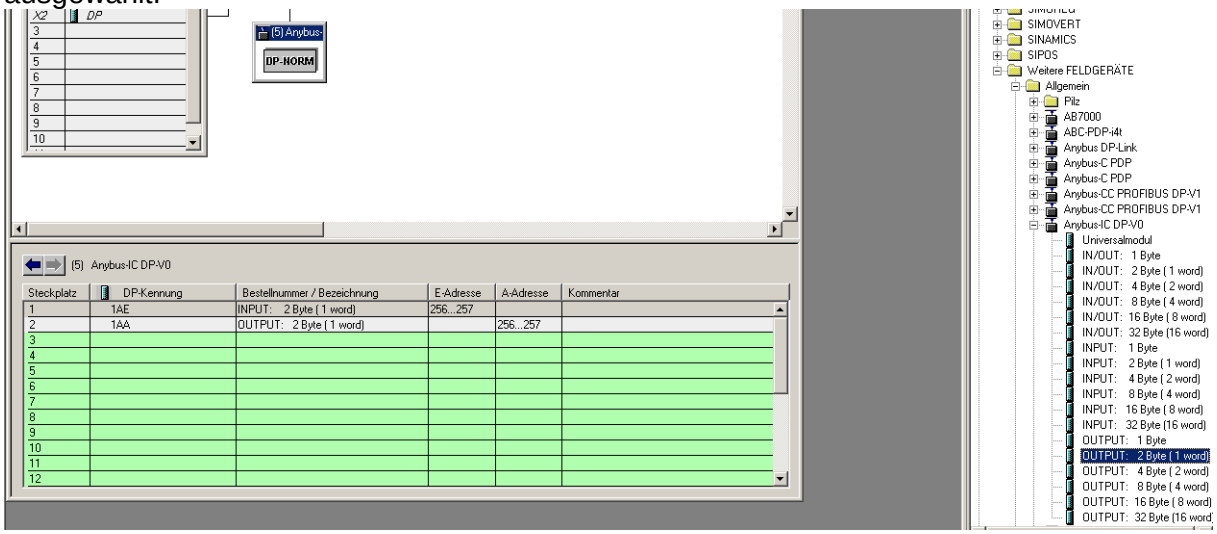
Markiert man nun das Profibus-Mastersystem, fügt ein Doppelklick auf „***Anybus-IC DP-V0***“ im Bereich „***weitere Feldgeräte > Allgemein***“ ein ABIC als Slave hinzu und öffnet sein Konfigurationsmenü. Nach dem Einstellen der Profibus-Adresse wird dieses Fenster geschlossen.



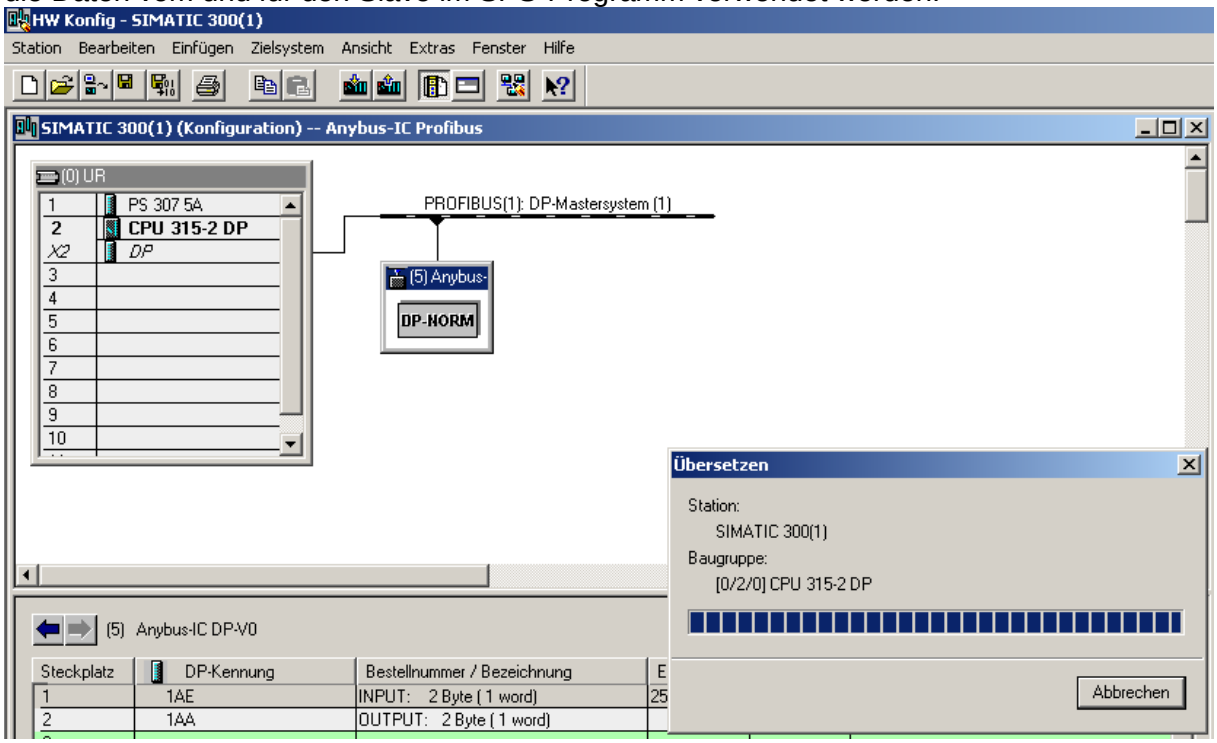
Sollte das ABIC noch nicht im Hardwarekatalog vorhanden sein, muss es über „***Extras > GSD-Dateien installieren…***“ hinzugefügt werden. Hierzu wird die GSD-Datei verwendet, die auf www.anybus.de im Supportbereich des ABIC für Profibus heruntergeladen werden kann.



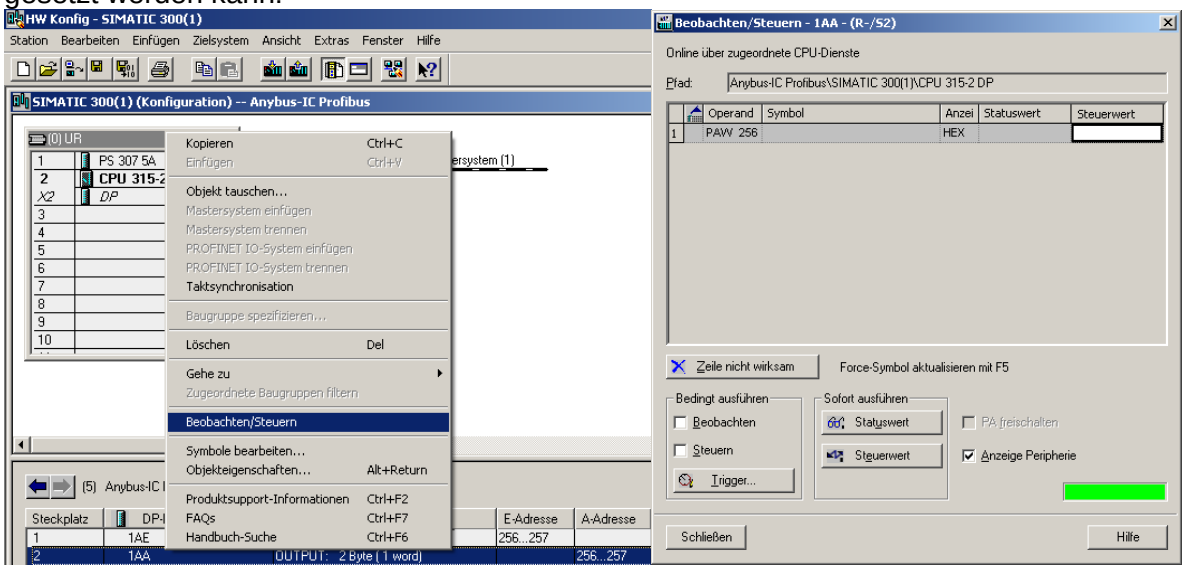
Nachdem der Slave eingefügt ist, wird dessen I/O-Größe durch Auswahl der entsprechenden Module festgelegt. Hier werden zum Beispiel ein 2Byte-In- und ein 2Byte-Out-Modul ausgewählt.



Nachdem die Hardwarekonfiguration übersetzt und in die Steuerung geladen wurde, können die Daten vom und für den Slave im SPS-Programm verwendet werden.



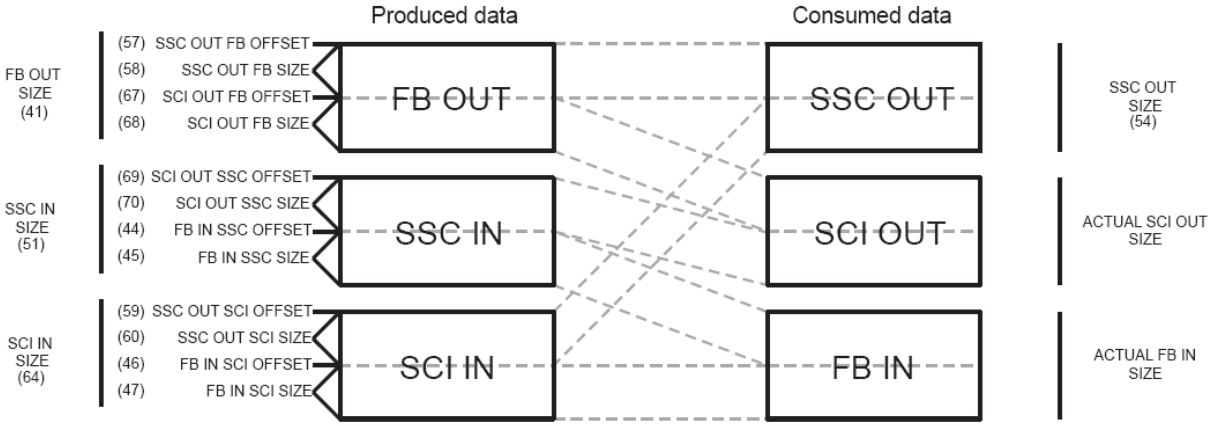
Über den Punkt ***Beobachten/ Steuern*** im Kontextmenü eines I/O-Moduls kann ein Fenster geöffnet werden, über das der Eingangs- bzw. Ausgangswertwert eines Moduls geprüft bzw. gesetzt werden kann:



5. Anhang

5.1 Überblick über Mapping-Parameter

Ergänzend zu den Informationen in Kapitel 5 „I/O Mapping“ des Design Guide Anybus-IC finden Sie im Folgenden eine Skizze, die die Mapping-Parameter visualisiert:



5.2 Weitere Dokumente

Die komplette Dokumentation zur Anybus-IC-Familie finden Sie im Internet (www.anybus.de).
Insbesondere sind für die Implementierung nötig:

- Design Guide Anybus-IC
  (beschreibt die feldbusübergreifenden Funktionen und Parameter)
- Design Appendix Anybus IC Profibus
  (beschreibt Profibus-spezifische Funktionen und Parameter)

HMS Industrial Networks GmbH
Emmy-Noether-Straße 17
D-76131 Karlsruhe
Tel: +49 (0) 721 989777 - 000
Fax: +49 (0) 721 989777 - 010
E-Mail: info@hms-networks.de
Internet: www.anybus.de