Otto-von-Guericke-Universität Magdeburg
Fakultät Elektrotechnik
Institut für Prozeßmeßtechnik und Elektronik (IPE)

Laborpraktikum Nachrichtentechnik

Versuch P-NET

NT 20

Institut für Prozeßmeßtechnik und Elektronik (IPE)
Lehrstuhl Nachrichtentechnik
Prof. Dr.-Ing. habil. F. Güttler

Postfach 4120
39016 Magdeburg
Tel.: (0391) 67-146 55

1. Versuchsziel

Mit diesem Versuch sollen Kenntnisse über die Parametrierung, Konfigurierung und Handhabung des seriellen Bussystems P-NET vermittelt werden. Aufbauend auf den Kenntnissen über serielle Bussysteme, werden zunächst die Adressierung der Busmodule, die Gatewaystrukturen und das Prinzip des Tokenmanagements untersucht. Daran anschließend werden Telegrammanalysen und Zeitmessungen durchgeführt. Abschließend erfolgt die applikationsbezogene Konfigurierung eines P-NET-Systems.

2. Theoretische Grundlagen

2.1. Eigenschaften und Anwendungsgebiete des P-NET

P-NET ist ein multimaster- und multinetfähiges Feldbussystem für den vorrangigen Einsatz in verfahrenstechnischen Prozessen mit mittleren zeitlichen Anforderungen. Einsatzgebiete sind u.a. Lebensmittelproduktion und Landwirtschaft, Grundstoff- und Textilindustrie sowie Fahrzeugausrüstung. Eine typische P-NET-Installation weist Antwortzeiten bis zu einigen Millisekunden und eine Buslänge bis zu einem Kilometer auf. Für Applikationen, bei denen die Antwortzeiten im Mikrosekundenbereich liegen (schnelle Steuerungstechnik), ist P-NET dagegen nicht geeignet.

Die verfügbaren Busmodule basieren auf Mikrocontrollern und enthalten vorprogrammierte Funktionen (Interface-Module) oder sind in einer Hochsprache (PASCAL-Dialekt) programmierbar (Controller). Die Echtzeitdaten liegen in den jeweiligen Modulen in skalierter Form vor. Ein konzentriertes Prozeßabbild in einem Busmodul, wie z.B. beim INTERBUS-S, ist bei P-NET nicht prinzipbedingt vorhanden. Der Zugriff auf die Daten in den Busmodulen erfolgt über Funktionen der Schicht 7 des OSI-Modells. Es existieren standardisierte Datenformate und Funktionsbeschreibungen (Channels) innerhalb des P-NET-Standards. Die europäische Feldbusnorm EN 50170 enthält neben PROFIBUS und WorldFIP auch P-NET als standardisierte Implementierungsvariante.

2.2. Topologie

Das P-NET ist ein Multi-Master-System, das entsprechend **Bild 2.1** eine zum Ring geschlossene Linienstruktur aufweist. Die elektrischen Eigenschaften basieren auf der Norm RS 485. Eine spezielle Anschaltelektronik in den Busmodulen erlaubt in Verbindung mit dem zum Ring geschlossenen geschirmten zweiadrigen Buskabel die Nutzung von bis zu 125 Busmodulen (davon bis zu 32 Mastermodule) je Segment. Abschlußwiderstände am Buskabel sind nicht notwendig.

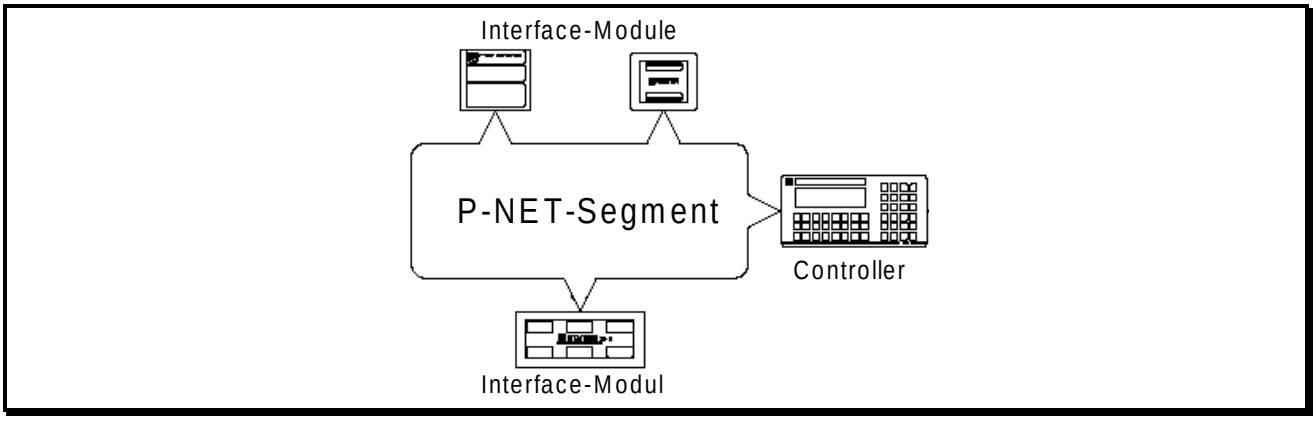

Bild 2.1: Topologie eines P-NET-Segments
Interface-Modul: Prozeßinterface (Universalmodul, Digitalmodul, Wägemodul usw.)

Eine Besonderheit bei P-NET ist die im Protokoll (Schichten 3 und 4) implementierte Multinetzfähigkeit. Mittels multiportfähiger Master können so einzelne P-NET-Segmente zu einem komplexen

Netzwerk verknüpft werden (**Bild 2.2**). Damit ist die Realisierung von Systemen mit mehr als 125 Teilnehmern ebenso möglich, wie eine Verteilung der Buslast auf spezialisierte Segmente (Prozeßsteuerung - Netze 1 bis 3 und Leitsystem - Netz 4) oder eine Verbesserung der Fehlertoleranz des Gesamtsystems. Das Multinetz-Konzept erlaubt schließlich den Aufbau hierarchischer Systeme.

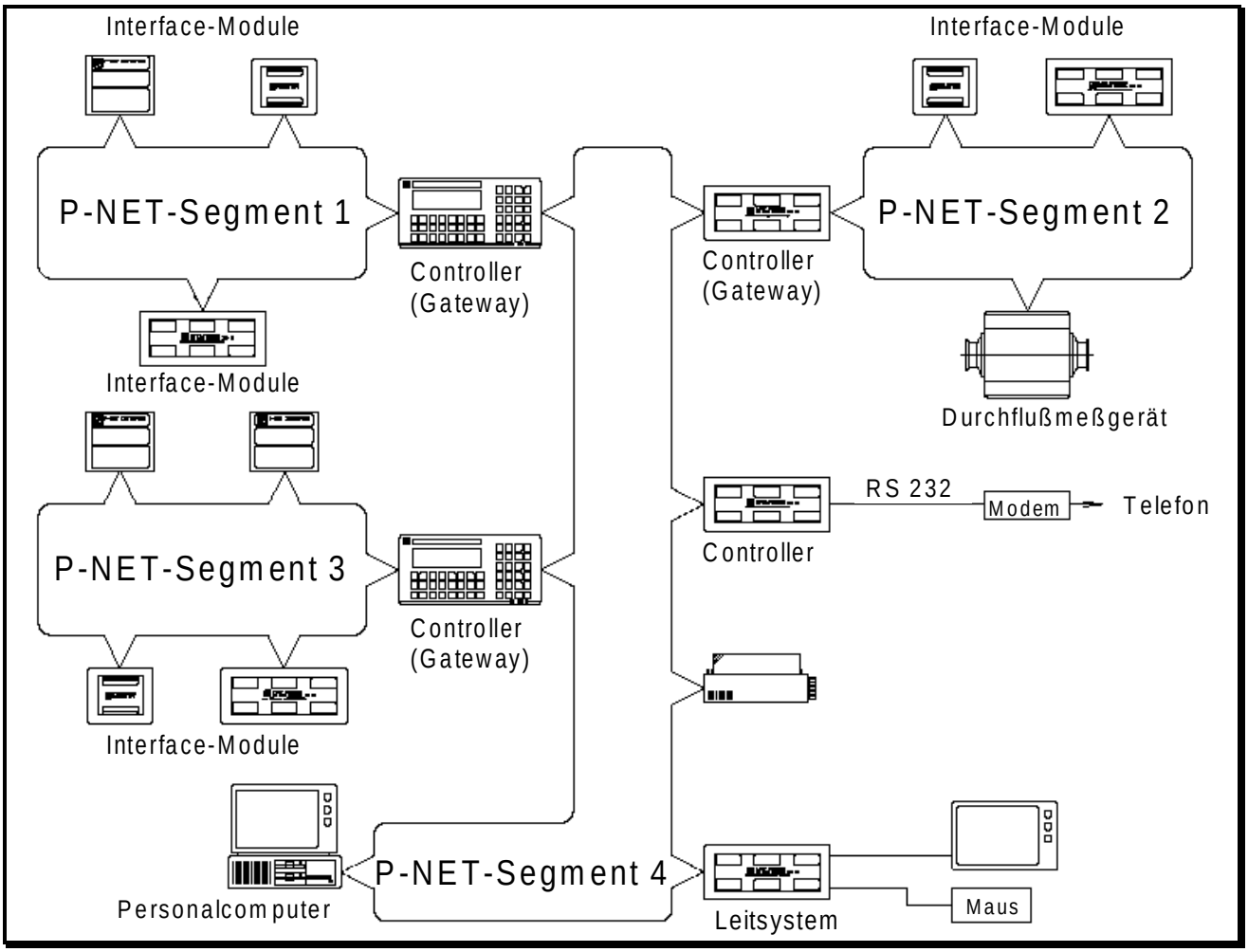

Bild 2.2: Komplexes P-NET-System durch Ausnutzung der Multinet-Fähigkeit

2.3. Das P-NET-Protokoll

2.3.1. Telegrammaufbau

Die P-NET-Telegramme werden mittels NRZ-codierter UART-Zeichen asynchron übertragen. Der Aufbau sowohl eines einzelnen Zeichens als auch eines kompletten Telegramms ist im **Bild 2.3** dargestellt. Zusätzlich zum Startbit und zu den acht Datenbits jedes Zeichens wird vor dem Stopbit noch ein Bit (A/D) eingefügt, das der Erkennung des ersten Adreßbytes dient (A/D=1) und damit gleichzeitig den Beginn eines Frames kennzeichnet. Die Zeichen werden mit einer festen Datenrate von 76,8 kbit/s übertragen. Eine zweite Variante des Protokolls erlaubt die Nutzung von Punkt-zu-Punkt-Verbindungen mittels RS 232 bei Datenraten von 1,2 bis 38,4 kbit/s.

Das Telegramm beginnt immer mit einem Adreßfeld von 2 bis 24 Byte Länge, das Ziel- und Quelladresse beinhaltet. Daran schließt sich ein Control/Status-Feld von einem Byte Länge an. Das Control/Status-Feld beinhaltet im Request-Frame die Kodierung des durchzuführenden Schicht-7-Dienstes (Bit 0 bis Bit 2). Im Response-Frame wird das Control/Status-Feld zur Übermittlung von Fehlerinformationen benutzt.

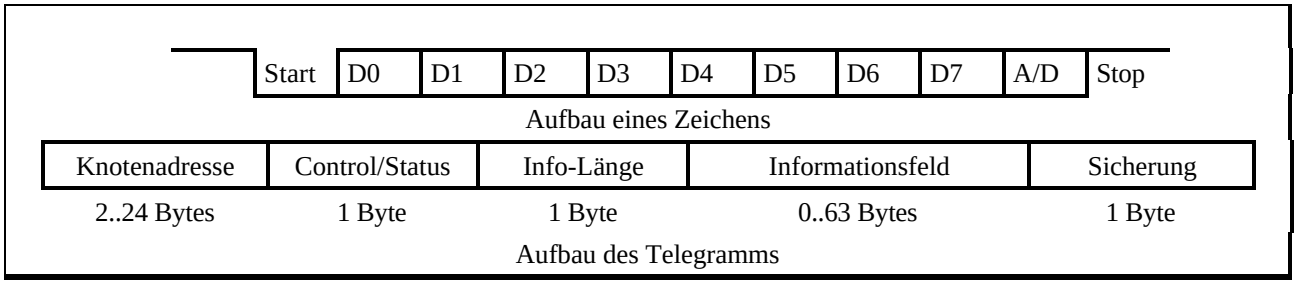

Bild 2.3: Aufbau von UART-Zeichen und Telegramm bei P-NET

Das Informationslängenfeld (1 Byte) enhält in den Bits 0 bis 5 die Längenangabe des sich anschließenden Informationsfeldes (0..63 Bytes). Das Informationsfeld selbst enthält den Verweis auf die anzusprechende Variable sowie die zu schreibenden bzw. die gelesenen Werte.

Die Datensicherung erfolgt mit einem Prüfsummenverfahren. Das Sicherungsfeld enthält das Zweierkomplement der errechneten Prüfsumme. Für spezielle Anwendungen existiert eine Protokollvariante mit 2-Byte-CRC-Sicherung.

2.3.2. Tokenmanagement

Die Vergabe des Buszugriffsrechts unter den Master-Modulen wird bei P-NET mittels eines Token-Verfahrens gesteuert. Das P-NET-Protokoll verwendet einen virtuellen Token-Mechanismus. Er basiert auf einer Zeitsteuerung und benötigt keine Telegramme zur Token-Weitergabe, wie z.B. Profibus. Dadurch wird eine Reduzierung der Buslast erreicht.

Wesentliche Bedeutung für die Funktionsweise der Tokenweitergabe besitzt das Zeitregime des P-NET-Protokolls (**Bild 2.4**). Seine Grundlage bildet das Prinzip der „immediate response“, das am Ablauf einer Datenübertragung nach dem Client-Server-Modell veranschaulicht werden soll. Der Client muß dabei ein Master-Modul sein, während der Server Master oder Slave sein kann. Ein Client beginnt innerhalb von 2 bis 7 Bitdauern nach Erhalt des Tokens mit dem Senden eines Request-Telegramms. Das Prinzip der immediate response verlangt nun vom Server, innerhalb von 11 bis 30 Bitdauern (390 µs) nach dem vollständigen Empfang des Request-Telegramms mit dem Senden der Response zu beginnen. Diese kurze Reaktionszeit wird durch protokollbegleitende Auswertung des Request-Frames möglich. Stehen die Daten zu diesem Zeitpunkt noch nicht zur Verfügung, sendet der Server ein Quittungstelegramm mit der Kodierung „busy“ im Control/Status-Feld. Der Client muß den ausstehenden Service beim nächsten Tokenbesitz vervollständigen. Bei Netzübergängen sendet das Gateway „answer comes later“ und schickt die eigentliche Response bei deren Komplettierung und eigenem Tokenbesitz an den Client zurück.

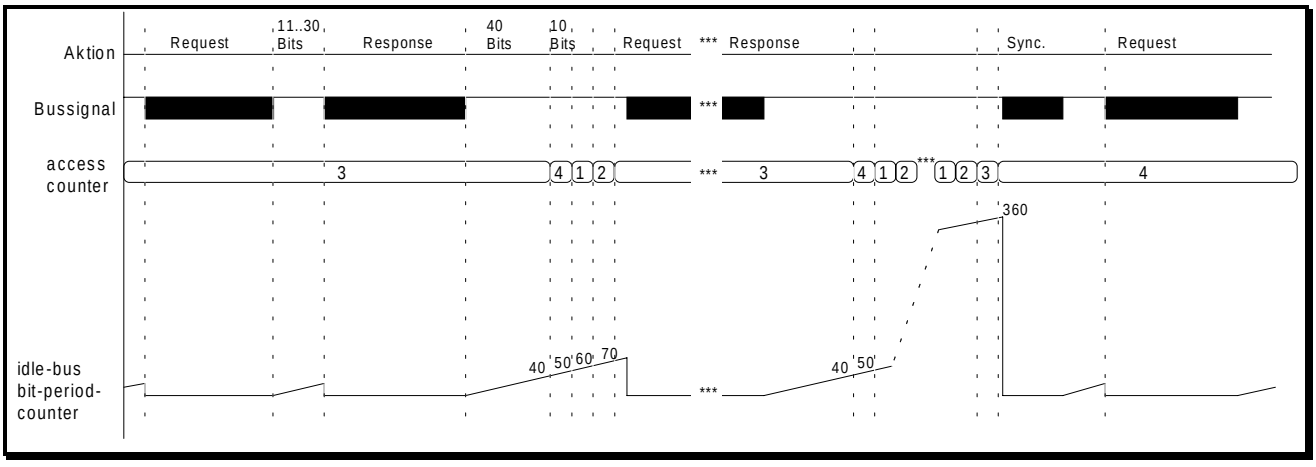

Bild 2.4: Zeitregime des P-NET-Protokolls

Ein Master darf bei Tokenbesitz lediglich ein Request-Telegramm aussenden. In Verbindung mit dem Prinzip der immediate response läßt sich so das Zeitverhalten einer P-NET-Installation gut berechnen. Die Tokenweitergabe wird durch zwei Zähler gesteuert, die für jedes P-NET-Port eines Mastermoduls implementiert sind. Ein Zähler zählt mit der Bitfrequenz, wenn der Bus frei ist (idle-bus bit-period-counter). Bei Datenverkehr auf dem Bus wird dieser Zähler rückgesetzt. Erreicht der Zähler den Wert 40, so wird ein weiterer Zähler, der access counter, inkrementiert. Ein Master hat genau dann das Senderecht - den Token - wenn der Zahlenwert des access counters seiner eigenen Knotenadresse entspricht.

Mit Beginn des Request-Telegramms des sendeberechtigten Masters werden die idle-bus-Zähler aller Mastermodule rückgesetzt und der beschriebene Vorgang vollzieht sich erneut. Hat der sendeberechtigte Master keinen Sendewunsch (oder ist er eventuell überhaupt nicht am Bus), so zählt der idle-bus-Zähler weiter, da kein Datenverkehr auftritt. Im Abstand von jeweils 10 Bitdauern wird der access counter inkrementiert (Zählerstände 50, 60, usw.) und damit das Senderecht an die anderen Master weitergegeben. Auf diese Weise wird das Token alle 10 µs weitergegeben, falls ein Master keinen Sendewunsch hat.

Der access counter zählt bis zu einem innerhalb eines Netzes (Bussegments) eindeutigen Maximalwert, der die höchste mögliche Adresse für einen Master in diesem Netz angibt. Dieser Wert ist projektierbar zwischen 1 und 32 (der maximal möglichen Anzahl von Mastern an einem Netz) und wird in der Variablen NoOfMasters eines jeden Ports eines Mastermoduls gespeichert. Er sollte so projektiert werden, daß er etwas größer als die tatsächliche Anzahl der Mastermodule an einem Netz ist, um Reserven für Erweiterungen bereitzustellen (z.B. Inbetriebnahmeterminals, PCs usw.). Erreicht der access counter den Wert NoOfMasters, so wird er beim nächsten Zählimpuls durch den idle-bus-Zähler wieder auf 1 gesetzt. Ein Master, der eine höhere Knotenadresse als NoOfMasters des Netzes besitzt, wird durch den beschriebenen Mechanismus niemals das Senderecht erhalten. Es ist sinnvoll, die Master im Adreßbereich von 1 bis 32 anzuordnen, während die Slavemodule höhere Adressen erhalten sollten. Außerdem sollten die Slaveadressen oberhalb der NoOfMasters des jeweiligen Netzes angeordnet sein.

Jeder Master vergleicht beim Empfang eines Request-Telegramms dessen Quelladresse mit dem Wert des access counters. Stimmen beide Werte nicht überein, so ist der Master „out of sync“ und sperrt seine eigene Sendefunktion. Tritt beim nächsten Empfang eines Request-Telegramms der gleiche Offset zwischen der Quelladresse im Telegramm und dem Wert des access counters auf, so wird der access counter mit dem Wert aus der Quelladresse gesetzt. Damit ist die Synchronisation wieder erreicht.

Sollte keiner der Master einen Übertragungswunsch haben, so wird der idle-bus-Zähler immer weiter inkrementiert. Nach 360 Bitdauern sendet der Master mit dem Token seine eigene Knotenadresse, um somit die idle-bus-Zähler in allen Mastern infolge Datenverkehrs auf dem Bus zurückzusetzen. Auf diese Weise werden Taktdifferenzen in den einzelnen Mastern ausgeglichen und es wird eine Synchronisation sämtlicher Master eines Netzes auch dann erreicht, wenn keine Übertragungsanforderungen in den Mastern bestehen.

2.3.3. Adressierung

Das Adreßfeld eines P-NET-Telegramms enthält die Adresse eines Busmoduls, die einfach oder komplex aufgebaut sein kann (**Bild 2.5)**. Die Datenbits 0 bis 6 enthalten dabei die eigentliche Adresse. Bit 7 dient der Unterscheidung von Ziel- und Quelladressen. In Request-Telegrammen kennzeichnen rückgesetzte Bit 7 die Bytes der Zieladresse, während bei denen der Quelladresse Bit 7 gesetzt ist. In Response-Telegrammen ist Bit 7 bei Zieladreßbytes gesetzt und bei Quelladreßbytes rückgesetzt. Als Adressen sind Werte von 1 bis 125 zugelassen. Die Adressen 0 und 127 sind reserviert, Adresse 126 dient als Rundspruchadresse.

einfache Adresse, Request

Bit7	Bit6..Bit0
0	Zieladresse
1	Quelladresse

komplexe Adresse, Request

Bit7	Bit6..Bit0
0	Zieladresse Byte 1
0	Zieladresse Byte 2
0	Länge der restlichen Adreßinformationen
0	Zieladresse Byte 3
1	Quelladresse Byte 1

Response (immer einfache Adresse)

Bit7	Bit6..Bit0
1	Zieladresse
0	Quelladresse

Bild 2.5: Aufbau des Adreßfelds im P-NET-Telegramm

Eine einfache Adresse kennzeichnet ein Busmodul im gleichen Bussegment (Netz) und ist als ein Byte kodiert. Der empfangende Knoten (Server) entfernt seine eigene Adresse (Zieladresse, Bit 7=0) aus dem Adreßfeld und fügt sie hinter der Quelladresse als neue Zieladresse wieder ein. Somit ist die Adresse für das Response-Telegramm zusammengestellt. Das **Bild 2.6** stellt das Prinzip der Adreß-bildung bei einfachen Adressen dar.

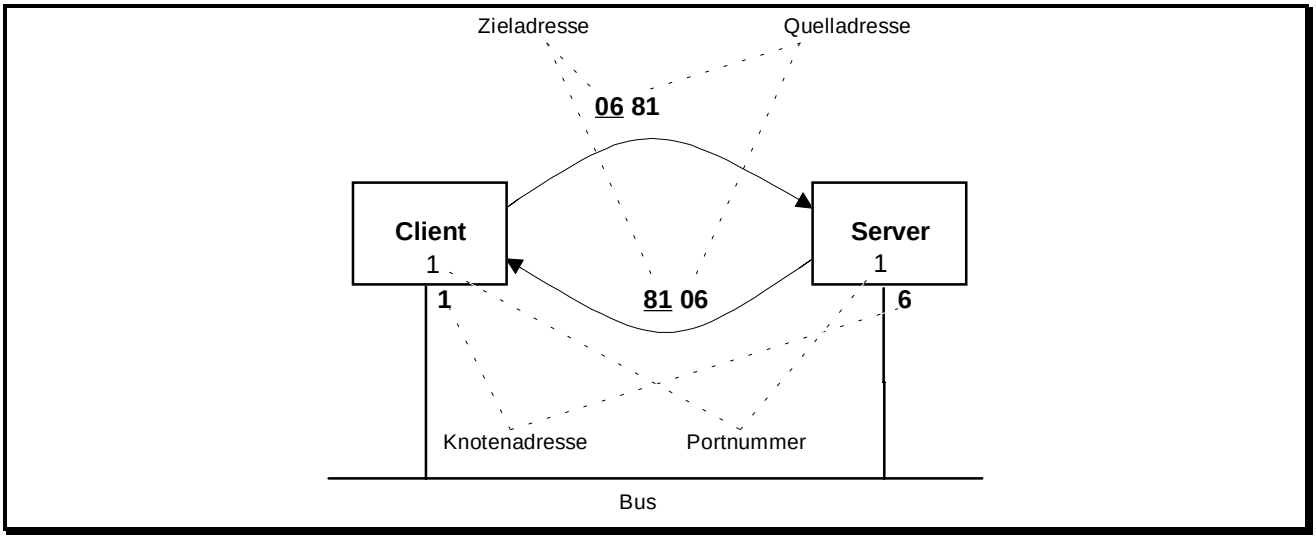

Bild 2.6: Prinzip der Adreßbildung bei einfachen Adressen

Eine komplexe Adresse wird zur Adressierung von Busmodulen in entfernten Netzen genutzt und ist aus einer Folge von Adreßbytes aufgebaut. Dabei stellt das erste Adreßbyte eine einfache Adresse für das mit dem aufrufenden Modul im gleichen Netz befindliche Gateway dar, über das der Zugang zum nächsten Netz erfolgt. Das folgende Byte spezifiziert die Nummer des Ports im Gateway, das zum Zugriff auf das nächste Netz verwendet wird. Das nächste Byte ist die einfache Adresse des Zielknotens innerhalb des zweiten Netzes. Da der Zielknoten wiederum ein Gateway sein kann,

können weitere Paare von Portnummern und Adreßbytes folgen. Auf diese Weise können bis zu zehn Gateways durchlaufen werden. Das dritte Byte einer komplexen Adresse enthält die Anzahl der noch folgenden Bytes des Adreßfeldes, damit das Protokoll dessen Ende erkennen kann.

Jedes Gateway sendet nach Empfang eines Request-Telegramms, das an ein anderes Netz adressiert ist, ein Response-Telegramm mit der Kodierung „answer comes later“ als Quittung zurück, um dem Prinzip der immediate response zu genügen. Danach entfernt es sein eigenes Adreßbyte (Knotenadresse) aus dem Telegramm. Die der Knotenadresse zugehörige Portnummer wird als erstes Byte der Quelladresse (Bit 7=1) eingefügt. Das Telegramm wird dem durch das erste Byte der Zieladresse spezifizierten Zielport zugeleitet. Am Zielport wird das erste Byte entfernt und die Knotenadresse an diesem Port mit gesetztem Bit 7 als erstes Byte der Quelladresse eingefügt. Das Telegramm wird ausgesandt. Dieser Vorgang wird in eventuellen weiteren Gateways wiederholt, bis letztlich die Zieladresse nur noch ein Byte lang ist. Das letzte Gateway speichert die Quelladresse in einer Warteschlange und sendet ein Request-Telegramm mit der einfachen Adresse. Das Antworttelegramm wird im Server generiert und als Response-Telegramm an das Gateway zurückgeschickt. Das Gateway generiert die eigentliche Response an den Client, indem es die gespeicherte Adresse als Zieladresse einsetzt und die Server-Adresse als erstes Byte der Quelladresse verwendet. Zusätzlich fügt es ein Null-Byte am Ende der Quelladresse ein. Dieses Telegramm durchläuft die anderen Gateways bis zurück zum Client. Das Null-Byte am Ende der Quelladresse verhindert die Generierung von Quittungen. Das **Bild 2.7** erläutert das Prinzip der Behandlung von komplexen Adreßfolgen durch das Protokoll.

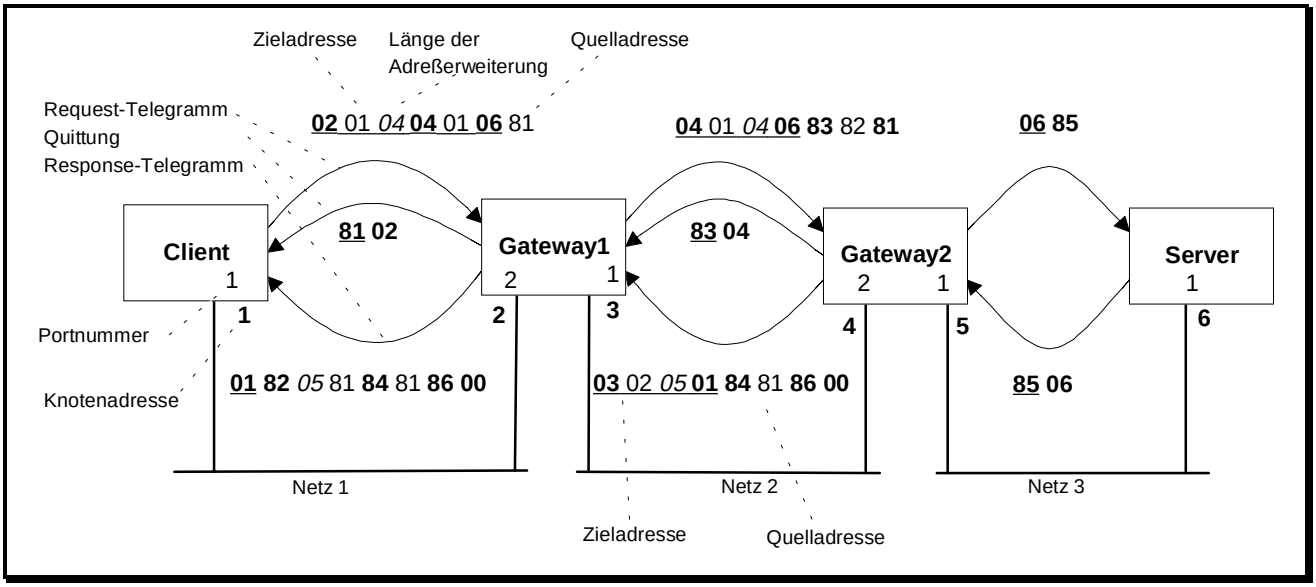

Bild 2.7: Adreßbildung bei komplexen Adressen

Die Adresse eines Busmoduls ist per Software einstellbar. Jedes P-NET-Modul verfügt über eine eindeutige Seriennummer. Zur Adreßvergabe wird die gewünschte Adresse zusammen mit der Seriennummer des Moduls im Informationsfeld eines Telegramms mit der Rundspruchadresse 126 (7EH) ausgesandt. Dieses Telegramm wird nur von dem Modul ausgewertet, dessen Seriennummer mit der ausgesandten übereinstimmt. Dieses Modul trägt dann die gewünschte Adresse in der dafür vorgesehenen Variablen ein und ist ab sofort unter dieser Adresse ansprechbar.

2.3.4. Variablen und Softwire-Nummern

Die Abbildung der Prozeßdaten erfolgt bei P-NET auf Variablen mit standardisierten Datentypen. Dabei sind sowohl einfache wie auch komplexe Datentypen (Strukturen) definiert. Einfache Datentypen sind z.B. Boolean, Byte, Char, Word, Integer, Longinteger, Real und Longreal, während komplexe Datentypen Array, Record und Buffer sein können. Verschachtelungen komplexer Datentypen

sind zulässig. Die Datendarstellung erfolgt nach dem Motorola-Format, d.h. das höchstwertige Byte wird zuerst übertragen. Der Zugriff auf einzelne Elemente eines komplexen Datentyps ist durch Angabe eines Byte-Offsets möglich. Datenstrukturen mit einer Länge größer als ein Byte müssen an geraden Adressen beginnen. Das bedeutet, daß bei Records und Arrays gegebenenfalls Füllbytes (Dummies) eingeschoben werden. Dies ist bei der Berechnung des Offsets zu berücksichtigen. Beim Zugriff auf eine Variable ist ihre Datenlänge im Request-Telegramm anzugeben.

Eine Variable kann in einem P-NET-Modul in verschiedenen physikalischen Speichermodulen abgelegt sein (RAM, ROM, EEPROM). Es existieren unterschiedliche Zugriffsrechte auf die Variablen (read only - ro, read write - rw, read protected write - rpw). Als read protected write gekennzeichnete Variablen können nur nach Setzen eines speziellen modulweiten Bits (write enable) geändert werden. Darüber hinaus sind die Initialisierung von Variablen im RAM durch Auslesen eines EEPROM-Wertes sowie die automatische zyklische Speicherung einer RAM-Variablen im EEPROM implementiert.

Die Adresse jeder Variablen wird in einer Liste abgelegt. Jedem Eintrag dieser Liste ist eine eindeutige Nummer zugeordnet, die Softwire-Nummer (SWNo). Der Zugriff auf eine Variable erfolgt unter Nutzung ihrer Softwire-Nummer. Das hat den Vorteil, daß die physikalische Adresse einer Variablen nicht bekannt sein muß. Die Referenzierung wird durch das Busprotokoll mittels einer Liste (softwire-list) vorgenommen und ist für den Anwender unsichtbar. Das Konzept der Softwire-Nummern erzeugt Implementationsunabhängigkeit, d.h. die interne Struktur eines Moduls kann geändert werden, ohne daß Applikationsprogramme geändert werden müssen. Das Zuordnungsprinzip der softwire-list ist im **Bild 2.8** dargestellt. Über eine spezielle Kodierung im Informationslängenfeld eines Telegramms ist der Zugriff auf absolute Adressen unter Umgehung der softwire-list möglich, besitzt aber nur für Servicezwecke Relevanz.

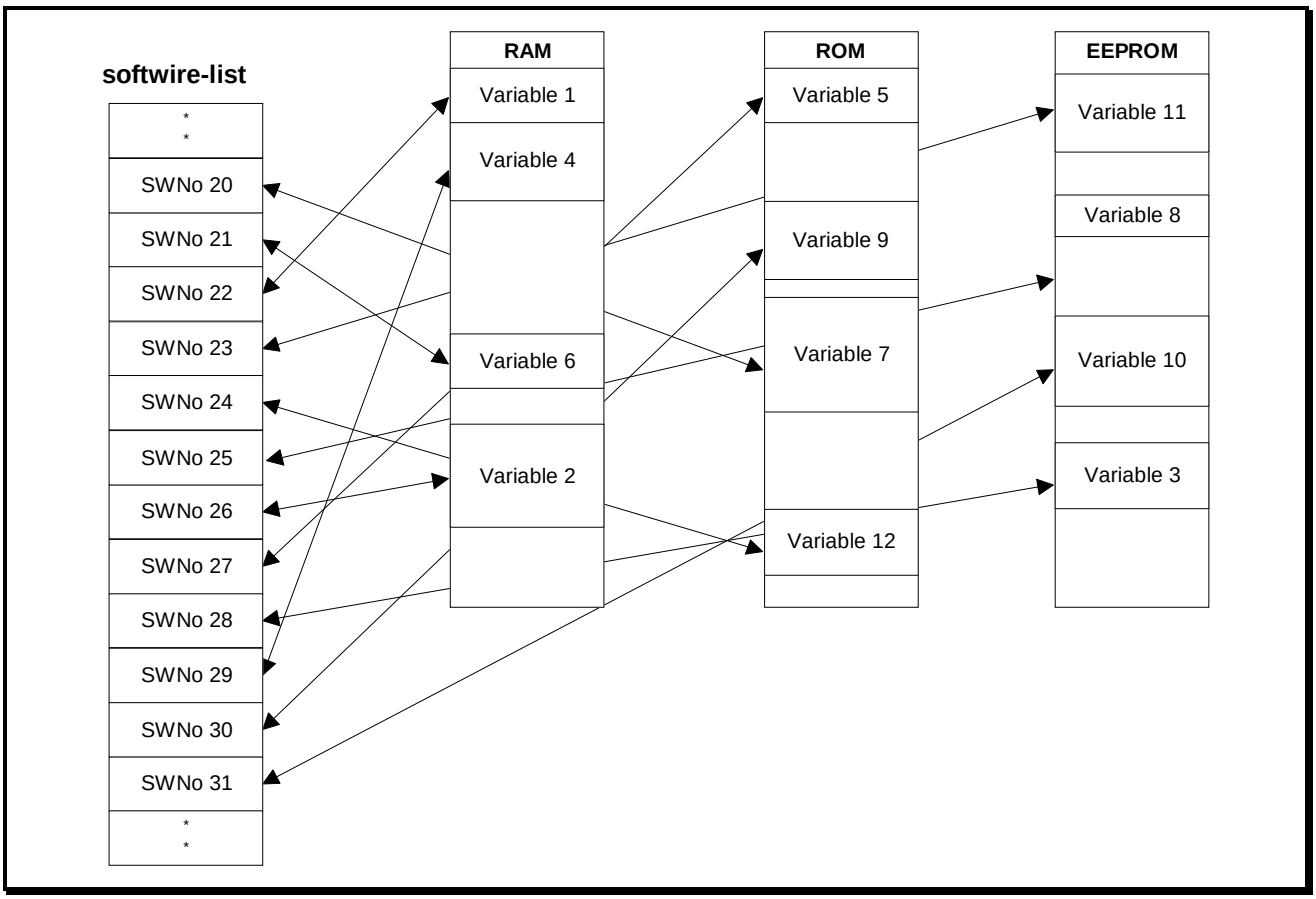

Bild 2.8: Abbildung von Variablen auf Softwire-Nummern

2.3.5. Kommunikationsfunktionen

Die Schicht 7 des P-NET-Protokolls definiert die Applikationsdienste, die auf die Variablen (Softwire-Nummern) angewandt werden können. **Tabelle 2.1** enthält die Kodierungen der sieben Dienste, wie sie im Control/Status-Feld eines Request-Telegramms kodiert werden.

Bit2..Bit0	Dienst	Funktion
001	Store	Schreiben auf eine Variable
010	Load	Lesen einer Variablen
011	And	bitweise logische UND-Verknüpfung zwischen Variable und Daten im Telegramm, Ergebnis wird in Variable geschrieben
100	Or	bitweise logische ODER-Verknüpfung zwischen Variable und Daten im Telegramm, Ergebnis wird in Variable geschrieben
101	Test-And-Set	Ressourcensteuerung (Test auf Verfügbarkeit, Belegung oder Freigabe von Ressourcen)
110	Long Load	Lesen eines großen Datenblocks (mehr als 56 Bytes, max 64 KBytes), automatische Blockbildung
111	Long Store	Schreiben eines großen Datenblocks (mehr als 56 Bytes, max. 64 Kbytes), automatische Blockbildung

Tabelle 2.1: Dienste der Applikationsschicht bei P-NET

In Response-Telegrammen beinhaltet das Feld Control/Status umfangreiche Fehlerinformationen (**Tabelle 2.2**). Damit ist sichergestellt, daß bei jedem Response-Telegramm zumindest eine Signalisierung auch solcher Fehlerzustände erfolgt, die unabhängig von der aktuell angesprochenen Variablen sind (z.B. Busfehler). Das Applikationsprogramm kann auf diese Weise feststellen, daß im Modul ein Fehler vorliegt und in eine entsprechende Behandlungsroutine verzweigen.

Bit2..Bit0 <>000	Service des Request-Telegramms
Bit6..Bit4	***Applikationsfehler***
000	fehlerfrei (OK)
001	beschäftigt (busy)
010	aktueller Fehler (actual data error)
100	warten (wait)
101	Antwort wird verzögert (answer comes later)
Bit7	historischer Fehler (historical data error)
Bit2..Bit0 =000	
Bit7..Bit3	***Busfehler***
00000	keine Antwort (no response)
00001	Zeitüberschreitung (time out)
00010	zu beschäftigt (too busy)
00011	zu langes Warten (wait too long)
00100	Puffer voll bzw. leer (buffer full/empty)
00101	Datenformatfehler (data format error)
00110	falsche Softwire-Number (SWNo error)

*
*
*

*
*
*

00111	falsche Knotenadresse (node address error)
01000	Schreibschutz (write protection)
01001	falsche Längenangabe des Informationsfeldes (info length error)
01010	falsche Dienstkodierung (instruction error)
10000	Problem bei Fehlererkennung (error detect failure)
10001	Formatfehler/Überlauf (overrun/framing error)
10010	Buskurzschluß (net short circuit)
10011	Port ist kein Master (port not master)
10100	Synchronisationsverlust (out of sync)
10101	RS 232 Handshakefehler (RS-232 handshake error)

Tabelle 2.2: Bedeutung der Fehlercodes im Control/Status-Feld

Das **Bild 2.9** zeigt zusammengefaßt am Beispiel des Lesens einer einfachen Variablen vom Datentyp word den Aufbau von Request- und Response-Telegramm. Der Client hat die Knotenadresse 2, der Server 6. Die Variable wird als Softwire-Nummer 23 (17H) angesprochen und besitzt den Wert 85 (55H). Das Control/Status-Feld des Request-Telegramms enthält den Kode für den Dienst LOAD (02H), es werden 3 Bytes im Informationsfeld übertragen. Dies sind High- und Low-Byte der Softwire-Nummer (00H, 17H) sowie die Datenlänge der Variablen (02 für word-Variablen). Das letzte Byte des Telegramms bildet das Sicherungsfeld. Das Response-Telegramm enthält im Control/Status-Feld den Kode „fehlerfrei“ (Bit 6..Bit 4=0, Bit 2..Bit 0<>0, Bit 7=0). Das Informationsfeld enthält High- und Low-Byte des Variablenwertes (00H, 55H). Folglich ist das Informationslängen-Feld mit dem Wert 2 gesetzt.

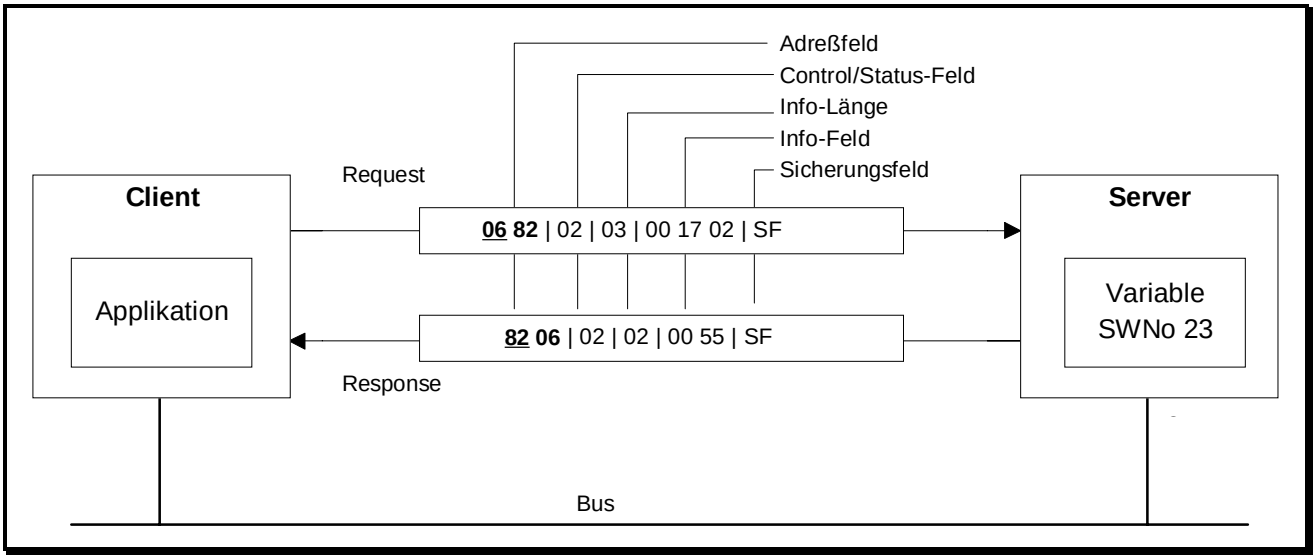

Bild 2.9: Aufbau von Request- und Response-Telegramm beim Lesen einer word-Variablen

2.4. Das Channel-Konzept

2.4.1. Aufbau und Funktion

Typische P-NET-Geräte sind Sensoren, Aktoren oder Interface-Module. Sie können über ein oder mehrere Prozeßsignale wie z.B. digitale Ausgänge oder analoge Eingänge verfügen. Jedem Prozeßsignal sind neben dem reinen Signalzustand bzw. -wert zusätzliche Informationen zugeordnet. Diese Informationen werden in Variablen gespeichert und betreffen spezielle Funktionen wie Konfigura-

tion, Umrechnung, Skalierung, Filterung, Fehlermeldungen etc. Im P-NET wird die Gesamtheit der auf ein Prozeßsignal bezogenen Variablen und Funktionen als Prozeßobjekt behandelt und trägt den Namen Channel (Kanal). Ein Channel beinhaltet alle notwendigen Daten, um die für ein Prozeßobjekt verfügbaren Steuerfunktionen auszuführen. Er enthält außerdem Daten, die die Wartung und das technische Management der Anlagenausrüstung unterstützen.

Ein Channel ist strukturiert als Folge von 16 Registern (Variablen). Diese 16 Variablen werden als Softwire-Nummern angesprochen und können strukturierte Datentypen aufweisen, die in verschiedenen Speichermedien abgelegt werden. Die Abbildungsvorschrift zwischen Register, Channel und SWNo legt fest, daß das niederwertigste Halbbyte (Nibble) einer Softwire-Nummer die Registernummer enthält, während die höherwertigen Nibbles die laufende Nummer des Channels im Modul festlegen. So addressiert die SWNo 0149H das Register 9 des Channels 14H eines Moduls.

Ziel bei der Einführung von Channels war der Gedanke, Automatisierungsfunktionen immer nach dem gleichen Prinzip ansprechen zu können. Dazu wurde der Aufbau eines Channels im Standard festgeschrieben. So enthält jeder Channel im Register 0 den aktuellen Ausgangswert, im Register 0FH Fehlerinformationen sowie im Register 9 Konfigurationsinformationen. Außerdem sind im Register 0EH die Channel-Identifikation sowie die Nutzung optionaler Register vermerkt. Diese Standardisierung ermöglicht neben dem einheitlichen Zugriff auf die wichtigsten Informationen jedes beliebigen Kanals die Erstellung herstellerunabhängiger Applikationssoftware sowie allgemeingültige Konfigurierungsprogramme. **Bild 2.10** stellt die obligatorischen Register eines jeden Channels dar.

Register (hex.)	Name	Speicher	Datenformat	Bedeutung
0	AnalogOut			Ausgangswert
9	ChConfig	EEPROM rpw	record	Konfiguration
D	Maintenance	EEPROM rpw	longint	Wartungsinformationen
E	ChType	ROM ro	record	Typ- u. Registerinfo.
F	ChError	RAM ro	record	Fehlerstatus

Bild 2.10: Obligatorische Register jedes Channels

Ein Channel muß nicht notwendigerweise mit Prozeßsignalen korrespondieren, er kann sich auch auf Strukturen, wie interne Funktionsblöcke mit ihren entsprechenden Parametern, beziehen. Ein Beispiel eines derartigen Channels ist ein PID-Regler, bei dem die Ausgangswerte Resultat eines Algorithmus sind und dessen Parameter in entsprechenden Variablen abgelegt sind.

Neben den globalen Festlegungen zum Aufbau von Channels werden Channels mit bestimmten Funktionen (Analogeingang, Digitalausgang, PID-Regler, Kommunikations-, Programm-Channel usw.) bereits im P-NET-Standard definiert. Die spezifische Ausprägung eines Channels ist dem Hersteller überlassen, er muß jedoch die im Standard als obligatorisch festgelegten Register in der vorgeschriebenen Form (Datenstruktur) implementieren. **Bild 2.11** stellt die Registerstruktur eines Analogausgang-Channels dar.

Kanäle werden durch die International P-NET User Organization standardisiert und durch eindeutige Nummern (in Register 0EH vermerkt) gekennzeichnet.

2.4.2. Der Service-Channel

Der Service Channel ist als Channel 0 in jedem Modul implementiert und nimmt modulbezogene Informationen auf. Dazu gehören neben Hersteller- und Versionsnummer unter anderem die Seriennummer des Moduls und seine Knotenadresse. Der Service-Channel ist auch für einfachste Busknoten in der beschriebenen Form obligatorisch. Er ist somit in jedem Modul immer in der gleichen

Form ansprechbar und kann beispielsweise zur Ermittlung des Modultyps und der darin implementierten Channels genutzt werden. **Bild 2.12** stellt die Registerstruktur des Service-Channels dar.

Register (hex.)	Name	Speicher	Datenformat	Bedeutung
0	AnalogOut	RAM rw	real	Ausgangswert, 0..100%
1				unbenutzt
2				unbenutzt
3				unbenutzt
4				unbenutzt
5				unbenutzt
6	Sollwert	RAM rw, Init EEPROM	real	Ausgangswert, skaliert
7*	HighLevel	RAM rw, Init EEPROM	real	oberer Grenzwert
8*	LowLevel	RAM rw, Init EEPROM	real	unterer Grenzwert
9	ChConfig	EEPROM rpw	record	Konfiguration
A				unbenutzt
B	Fullscale	EEPROM rpw	real	Endwert, skaliert
C	Zeropoint	EEPROM rpw	real	Anfangswert, skaliert
D	Maintenance	EEPROM rpw	longint	Wartungsinformationen
E	ChType	ROM ro	record	Typ- u. Registerinfo.
F	ChError	RAM ro	record	Fehlerstatus

Bild 2.11: Registerstruktur eines Analogausgangs-Channels
*: optionale Register

Register (hex.)	Name	Speicher	Datenformat	Bedeutung
0	NumberOfSWNo	ROM ro	integer	höchste SWNo im Modul
1	DeviceID	ROM ro	record	Gerätenummer, Programm-Version, Herstellerinfo.
2				unbenutzt
3*	Reset	RAM rw	byte	Modulreset
4	PnetSerialNo		record	Knotenadresse, Serienr.
5				unbenutzt
6*	TimeDate		record	Datums- und Zeitangabe
7*	FreeRunTimer	RAM ro	word	modulweite Zeitbasis
8*	WDTimer	RAM rw	real	Watchdog-Zähler
9	ModuleConfig	EEPROM rpw	record	Modul-Konfiguration
A*	WDPreset	EEPROM rpw	real	Watchdog-Voreinstellwert
B*	MailFilter	RAM rw, Init EEPROM	string	Message-Filter
C*	MailBox	RAM rw	buffer	Message-Puffer
D	WriteEnable	RAM rw	boolean	Schreibfreigabe f. EEPROM
E	ChType	ROM ro	record	Typ- u. Registerinfo.
F	CommonError	RAM rw	record	Modul-Fehlerstatus

Bild 2.12: Registerstruktur des Service-Channels
*: optionale Register

2.5. Programmierung von Mastermodulen

Mastermodule sind freiprogrammierbare Controller unterschiedlicher Ausprägung. Sie können in der Hochsprache PROCESS-PASCAL /**5**/ programmiert werden, einem um Funktionen zur Handhabung von Tasks und Netzwerkvariablen erweiterten Dialekt von Standard-Pascal. Ein Master kann verschiedene Tasks zyklisch oder ereignisgesteuert abarbeiten. Zur Handhabung der Programme (Download, Start, Stop usw.) werden MMS-konforme Services verwendet. Die Kommunikation zwischen unterschiedlichen Tasks erfolgt über globale Variablen, die als Softwire-Nummern deklariert sein können und somit auch über den Bus beeinflußbar sind.

3. Literatur- und Abkürzungsverzeichnis

3.1. Literatur

/1/ *Schnell, G.; u.a.:* **Bussysteme in der Automatisierungstechnik.**
Verlag Vieweg, Braunschweig/Wiesbaden, 1994.

/2/ *Bonfig, K.W.; u.a.:* **Feldbussysteme.**
expert Verlag, Ehningen bei Böblingen, 1992.

/3/ *International P-NET User Organization:* **P-NET Standard.**
1994.

/4/ *International P-NET User Organization:* **P-NET standardized general purpose channel types.**
Manual, 1992.

/5/ *Process Data:* **PROCESS PASCAL.**
Manual, 1995.

/6/ *Process Data:* **Universal Process Interface PD 3221.**
Manual, 1995.

/7/ *Process Data:* **Digital I/O PD 3120.**
Manual, 1993.

3.2. Abkürzungen

MMS	Manufacturing Message Specification
OSI	Open System Interconnection, Schichtenmodell von Kommunikationssystemen
ro	read only, Zugriffsart auf Variable
rpw	read protected write, Zugriffsart auf Variable, Speicherung im EEPROM
rw	read write, Zugriffsart auf Variable
SWNo	Softwire-Nummer

4. Beschreibung des Versuchsaufbaus

4.1. Versuchsaufbau P-NET

Der Versuchsaufbau besteht entsprechend **Bild 4.1** aus zwei P-NET-Segmenten (Netzen), die über einen Master (PD 3010) miteinander gekoppelt sind. Das Netz 1 enthält neben dem Master PD 3010 ein Digitalmodul PD 3120 (Slave) sowie den Testmaster, der vom PC aus gesteuert werden kann. Das Netz 2 verfügt über ein Universalmodul PD 3221 (Slave) sowie einen PC (Master), der mittels einer Einsteckkarte PD 3920 an das P-NET angeschlossen ist. Die Busmodule sind mit geeigneten Peripheriekomponenten verbunden (LED, Schalter, Meßpunkte). Der speziell für den Versuchsplatz entwickelte Testmaster ermöglicht den Zugang zu internen Zuständen und Ereignissen des Busprotokolls sowie die Erzeugung spezieller Testsequenzen, die für die Versuchsdurchführung erforderlich sind. Eine im PC befindliche Logikanalysatorkarte gestattet die Aufzeichnung der für den Versuch relevanten Signale und Ereignisse (access counter, E/A-Daten, Kommunikationsfunktionen). Zur Auswertung stehen vorbereitete Konfigurationsdateien mit den entsprechenden Trigger- und Kanaleinstellungen sowie Beschriftungen zur Verfügung.

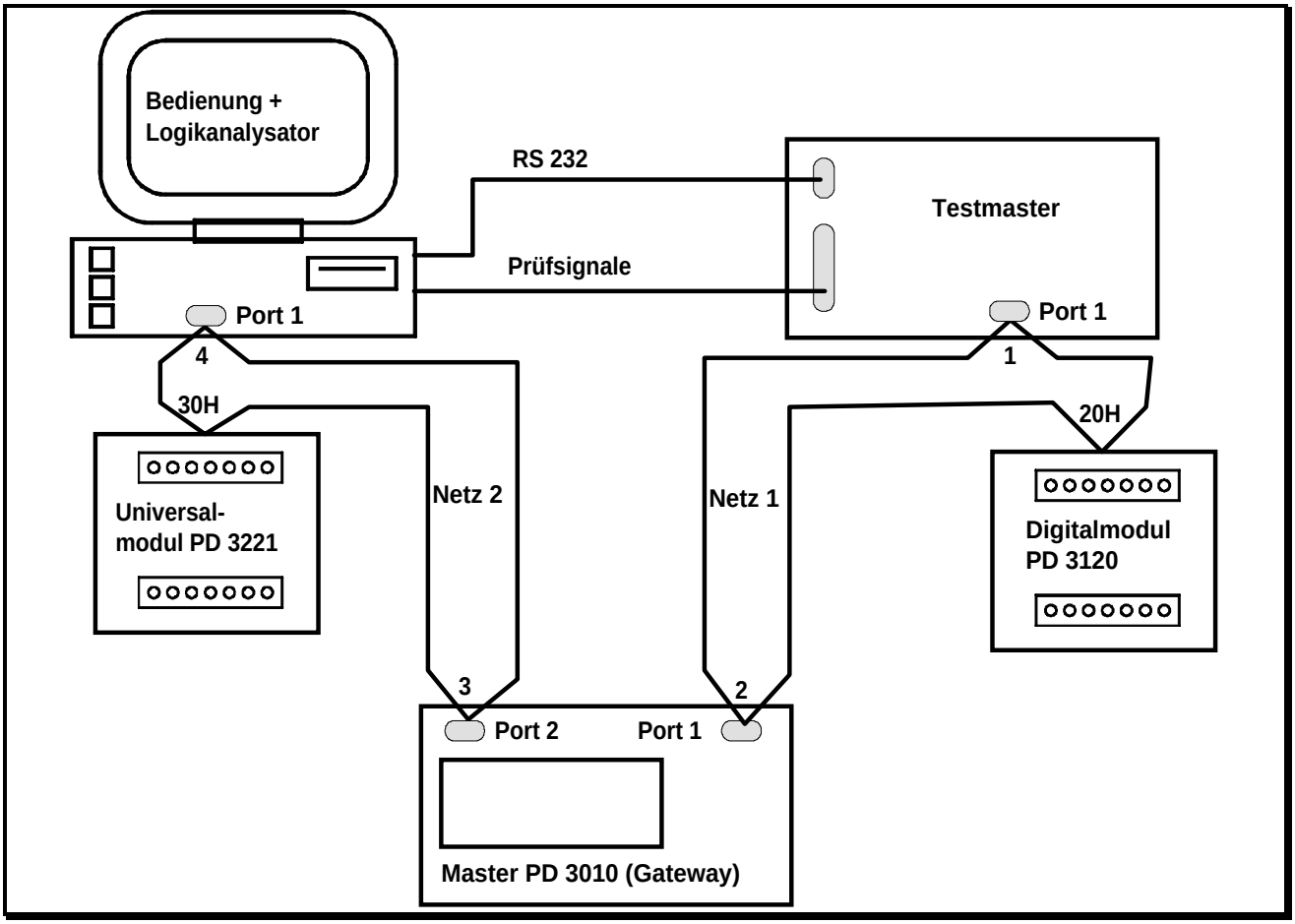

Bild 4.1: Versuchsaufbau P-NET

Die gesamte Bedienung des Versuchsaufbaus erfolgt am PC unter Windows™. Der PC und die Bedienereingaben stellen dabei den Anwendungsprozeß dar, der über die RS-232-Schnittstelle mit dem P-NET-Testmaster sowie über die P-NET-Interfacekarte mit den Modulen kommuniziert. Die Bedienung erfolgt im wesentlichen über Dialoge und ist selbsterklärend, so daß auf eine weitergehende Programmbeschreibung verzichtet werden kann. Spezielle Hinweise werden bei den einzelnen Versuchsschritten gegeben.

4.2. Busmodule

4.2.1. Universalmodul PD 3221

Das Universalmodul PD 3221 ist ein in P-NET-Installationen sehr häufig anzutreffendes Slavemodul. Es beinhaltet die folgenden Channels:

1	Service-Channel	Channel 0
4	Digital-E/A-Channels	Channel 1-4
2	Digitaleingangs-Channels	Channel 5-6
1	gemeinsamer Digital-E/A-Channel	Channel 7
2	Analogeingangs-Channels	Channel 8-9
1	Stromausgangs-Channel	Channel 0AH
1	PID-Regler-Channel	Channel 0BH
1	Programm-Channel (Calculator)	Channel 0CH
1	Pulse-Processor-Channel	Channel 0DH

Das **Bild 4.2** stellt das Blockschaltbild des Universalmoduls dar. Die Bezeichnungen und Datentypen für die einzelnen Register der Channels sind der am Versuchsplatz ausliegenden Beschreibung /**6**/ zu entnehmen.

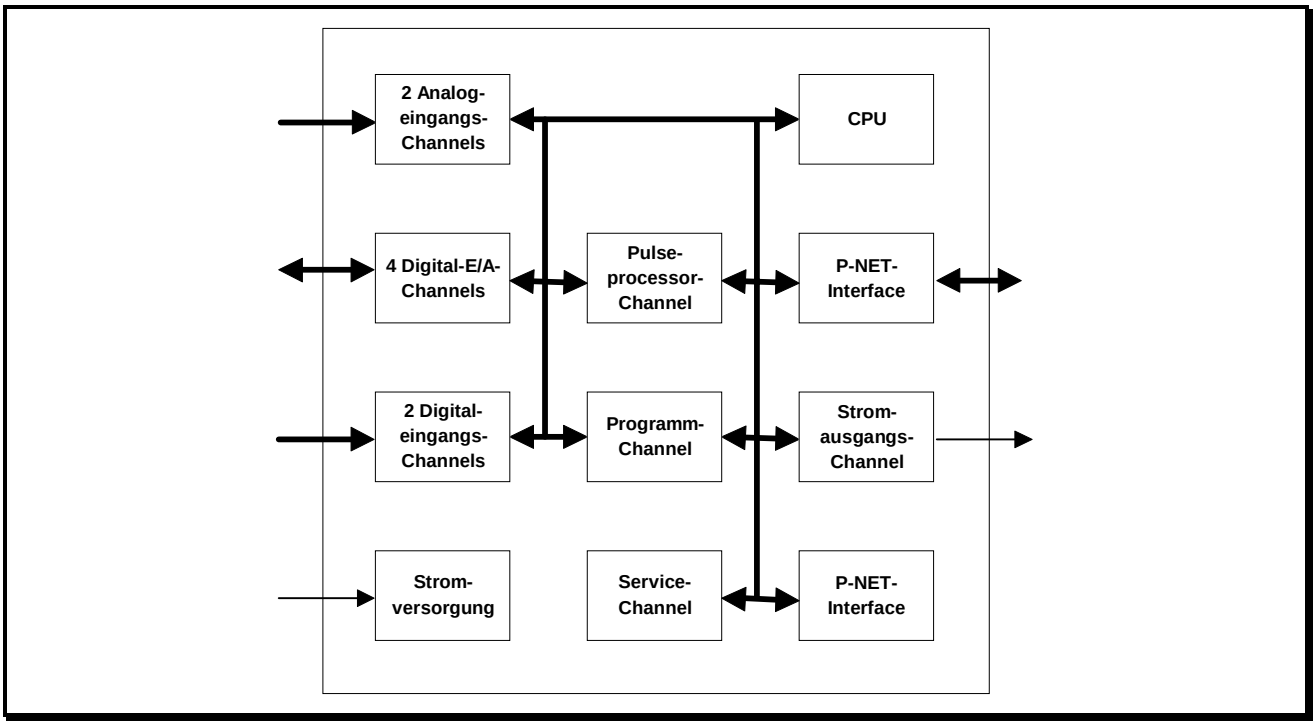

Bild 4.2: Blockschaltbild des Moduls PD 3221

4.2.2. Digitalmodul PD 3120

Das Digitalmodul PD 3120 wird zur Verarbeitung digitaler Ein- und Ausgangssignale eingesetzt und ist mit folgenden Channels ausgestattet:

1	Service-Channel	Channel 0
16	Digital-E/A-Channels	Channel 1-10H
1	gemeinsamer Digital-E/A-Channel	Channel 11H
1	Programm-Channel (Calculator)	Channel 12H
1	Daten-Channel	Channel 13H

Das **Bild 4.3** stellt das Blockschaltbild des Digitalmoduls dar. Die Bezeichnungen und Datentypen für die einzelnen Register der Channels sind der am Versuchsplatz ausliegenden Beschreibung /**7**/ zu entnehmen.

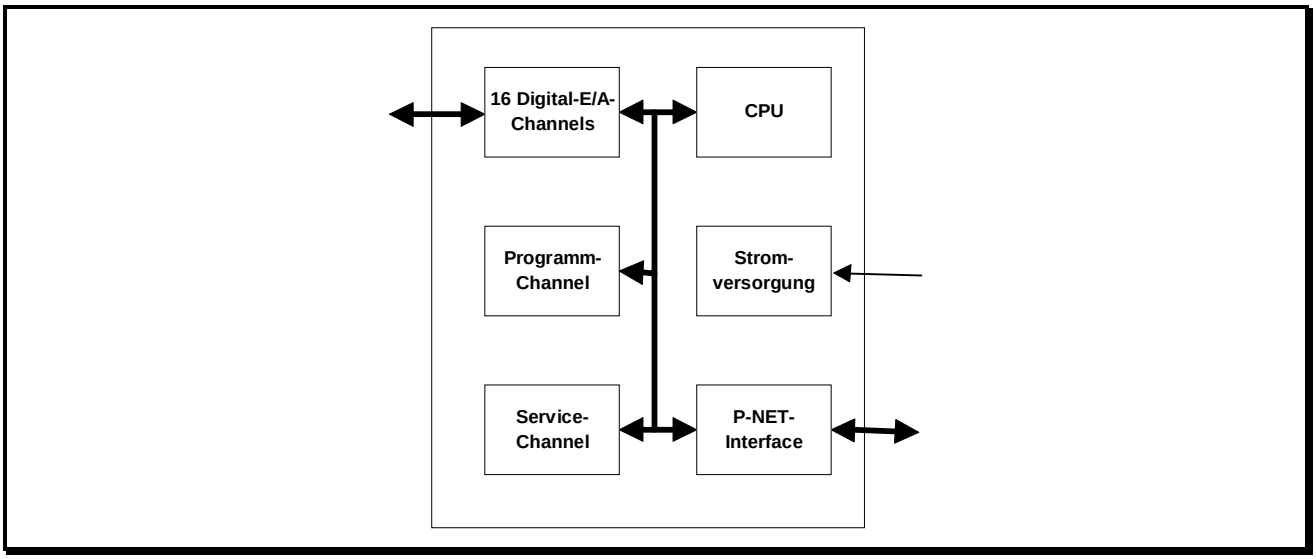

Bild 4.3: Blockschaltbild des Moduls PD 3120

4.2.3. Mastermodul PD 3010

Das Mastermodul PD 3010 ist ein freiprogrammierbarer Controller mit einer Tastatur- und Display-Einheit. Dieser Master verfügt über zwei P-NET-Ports und wird im Rahmen des Versuchs als Gateway zwischen Netz 1 (Port 1) und Netz 2 (Port 2) genutzt, seine Programmierung ist nicht Gegenstand des Versuches.

4.3. Monitorprogramm

Zur Durchführung von Zugriffen auf P-NET-Module vom PC aus steht ein Monitorprogramm zur Verfügung (**Bild 4.4**). Es ist Bestandteil der herstellerseitig mitgelieferten Werkzeuge zur Konfiguration und Diagnose. Die einzelnen Felder einer Anzeigezeile sind durch Doppelclick ansprechbar. Im Adreßfeld ist eine P-NET-Adresse, gefolgt von der anzusprechenden Softwire-Nummer, anzugeben. Der Zugriff vom PC aus (Portnummer auf der P-NET-Karte ist immer 01H) auf SWNo 90H eines Moduls mit der Adresse 45H ist z.B. als 01 45.90 einzutragen. Nach Einstellung des Byte-Offsets für Elemente von Strukturen, des Datentyps und des Anzeigeformats wird durch Einschalten der Checkbox zyklisch eine Datenübertragung durchgeführt.

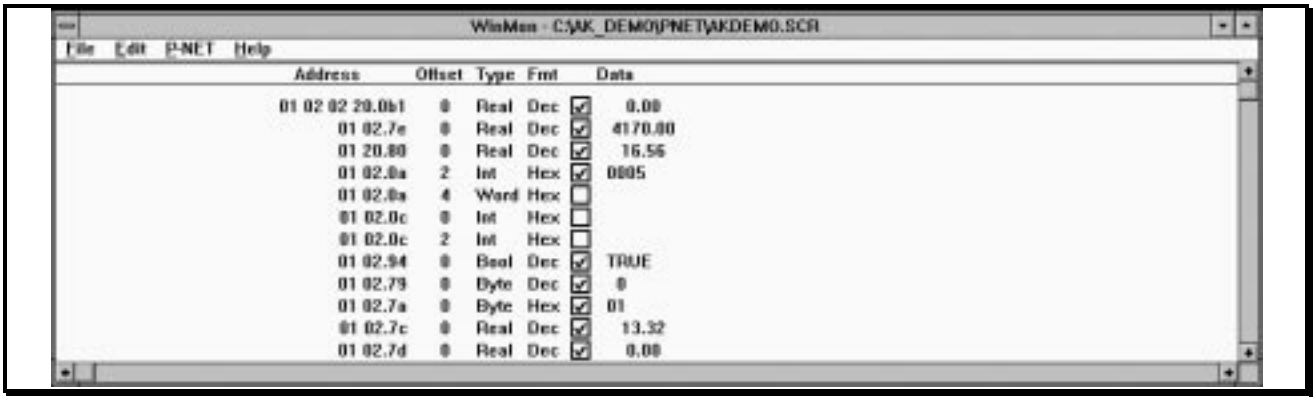

Bild 4.4: Screenshot des P-NET-Monitorprogramms

5. Aufgaben zur Versuchsvorbereitung

Vorbereitung und Durchführung des Versuches setzen grundlegende Kenntnisse über das Schichtenmodell der offenen Kommunikation, über Busstrukturen und Buszugriffsverfahren, Adressierungsverfahren, das Client-Server-Modell sowie Kommunikationsdienste und -objekte voraus. Informieren Sie sich darüber anhand der angegebenen Literatur und der Erläuterungen im Abschnitt 2. Der Standard und die Handbücher **/3/-/7/** können beim Versuchsbetreuer eingesehen und ausgeliehen werden.

5.1. Charakterisieren Sie die in Automatisierungssystemen auftretenden Daten und leiten Sie daraus Anforderungen an des Übertragungssystem ab!

5.2. Beschreiben Sie die Topologie des P-NET und die sich daraus ableitenden Eigenschaften! Für welche Einsatzfälle ist P-NET gut und für welche weniger gut geeignet?

5.3. Was versteht man unter Echtzeitsicherheit, Zykluszeit, Antwortzeit, Reaktionszeit, Datensicherheit, Datenkonsistenz, Zuverlässigkeit und Verfügbarkeit? Charakterisieren Sie das P-NET anhand dieser Begriffe!

5.4. Was besagen Konformität und Interoperabilität? Welchen Zweck verfolgt die Definition von Channels? Vergleichen Sie Channels mit Geräteprofilen!

5.5. Wie ist das Protokoll des P-NET aufgebaut? Was versteht man unter Telegrammeffizienz?

5.6. Berechnen Sie die Telegrammeffizienz und die Zeit für den Datenzyklus bei einem Lesezugriff auf eine real-Variable eines Moduls im gleichen Netz!

5.7. Erläutern Sie das Prinzip des Tokenmanagements bei P-NET. Charakterisieren Sie dessen Vor- und Nachteile!

5.8. Wie werden Netzübergänge im P-NET realisiert? Welche Vorteile bietet die Multinetzfähigkeit? Wie erfolgt die Routenwahl?

5.9. Skizzieren Sie zur Realisierung der Aufgabe 6.5 geeignete Blockschaltbilder auf der Basis von Channels! Wie können Werte zwischen Slaves ausgetauscht werden?

5.10. Was ist beim Austausch von defekten Geräten einer P-NET-Installation notwendig, um die Anlage wieder in Betrieb nehmen zu können? Kann der Austauschvorgang bei laufendem Betrieb erfolgen?

6. Aufgabenstellungen zum Versuch

Schalten Sie die Betriebsspannung ein und starten Sie am PC die Programme „P-NET“ (Bedienprogramm) und „LA“ (Logikanalysator) aus der Programmgruppe „Praktikum P-NET“. In jedem der Dialoge ist angegeben, welche neue Konfigurationsdatei für den Logikanalysator benötigt wird. Die dort eingestellten Triggerbedingungen sind beim Logikanalysator jeweils vor dem Starten der Messung mit Run➢Run zu aktivieren. Sie können die aufgezeichneten Signale ausdrucken, um die Auswertung zu vereinfachen. Die ursprüngliche Systemkonfiguration kann durch Wahl des Menüpunkts „Bus➢Ausgangskonfiguration“ wieder hergestellt werden.

6.1. Einstellung von Moduladressen

Wechseln Sie zum Dialog „Bus➢Moduladresse“.

a) Ermitteln Sie die Seriennummern von Digital- und Universalmodul.

b) Vergeben Sie für beide Module neue Adressen durch Eintragung der benötigten Werte im Dialog und Absenden des Telegramms.

c) Überprüfen Sie die Funktionsfähigkeit der Module unter ihren neuen Adressen.

d) Können Sie die Adressen beider Module auf den gleichen Wert setzen?

6.2. Untersuchung von Gatewaystrukturen

Wechseln Sie zum Dialog „Bus➢Gateway“.

a) Ermitteln Sie die Adressen, die der Testmaster zum Zugriff auf Digital- und Universalmodul benötigt. Vergleichen Sie diese mit den Adressen, die Sie bei 6.1. genutzt haben!

b) Stellen Sie die Informationen zusammen, die Sie zum Zugriff auf die Gerätenummer der Slavemodule vom Testmaster aus benötigen.

c) Starten Sie einen Datenzyklus zur Ermittlung der Gerätenummer des Universalmoduls vom Testmaster aus. Werten Sie die Meldungen des Testmasters sowie die mittels Logikanalysator gemessenen Zeiten aus.

d) Starten Sie einen Datenzyklus zur Ermittlung der Gerätenummer des Digitalmoduls vom Testmaster aus. Werten Sie die Meldungen des Testmasters sowie die mittels Logikanalysator gemessenen Zeiten aus, und vergleichen Sie sie mit denen aus Aufgabe c).

e) Wie können Sie vom Testmaster aus mittels Gateway auf das Universalmodul zugreifen? Welche Zeiten messen Sie?

6.3. Untersuchung des Tokenmanagements

Wechseln Sie zum Dialog „Bus➢Tokenmanagement“.

a) Beobachten und interpretieren Sie den Wert des access counters des Testmasters.

b) Ermitteln Sie die Adressen der Mastermodule sowie die in den einzelnen Netzen eingestellten Werte für NoOfMasters. Stellen Sie die benötigten Informationen zusammen, um die Gerätenummer des Testmasters vom PC aus lesen zu können.

c) Starten Sie unter Nutzung des Monitorprogramms einen zyklischen Zugriff auf die Gerätenummer des Testmasters.

d) Ändern Sie den Wert von NoOfMasters im Testmaster und interpretieren Sie die Meldungen des Monitors bzw. des Testmasters.

e) Stellen Sie den Originalwert von NoOfMasters beim Testmaster wieder ein und beenden Sie den Zugriff durch den Monitor.

f) Ermitteln Sie die Zeiten zwischen Synchronisationstelegrammen.

6.4. Analyse von Telegrammen

Wechseln Sie zum Dialog „Kommunikation➢Wert schreiben".

a) Stellen Sie das benötigte Telegramm zusammen, um vom Testmaster aus einen Wert von 50 in Softwire-Nummer 0A0H des Universalmoduls zu schreiben. Führen Sie die Datenübertragung durch. Werten Sie das Response-Telegramm aus, und überprüfen Sie das Ergebnis des Schreibvorgangs mit Hilfe des Monitors.

b) Schreiben Sie vom Testmaster aus einen Wert von 50 in Softwire-Nummer 0ABH des Universalmoduls. Werten Sie das Response-Telegramm aus, und überprüfen Sie den Wert mit Hilfe des Monitors.

c) Schreiben Sie vom Testmaster aus einen Wert von 40H in Softwire-Nummer 10H des Digitalmoduls. Werten Sie die vom Testmaster empfangenen Telegramme aus, und interpretieren Sie diese.

d) Bauen Sie mittels Monitor eine zyklische Verbindung vom PC zum Testmaster auf, und lesen Sie dessen Gerätenummer. Provozieren Sie Datenübertragungsfehler im Testmaster, und werten Sie die Fehlermeldungen im Monitor aus.

6.5. Applikationsbezogene Konfigurierung mittels Channels

a) Realisieren Sie mit Hilfe des Digitalmoduls eine Konfiguration, bei der durch Betätigung des Schaltes an dessen Eingang 1 eine Leuchtdiode am Ausgang 9 geschaltet wird. Ermitteln Sie anhand des vorbereiteten Blockschaltbildes aus der Vorbereitungsaufgabe 5.9. die notwendigen Registerbelegungen der jeweiligen Channels.

b) Schreiben Sie die Werte in das Digitalmodul, und überprüfen Sie die Funktionsfähigkeit der Konfiguration. Wie läßt sich eine Schaltung mit invertierendem Ausgang realisieren?

c) Realisieren Sie mit Hilfe des Universalmoduls eine Konfiguration, bei der eine durch Betätigung des Potentiometers am Analogeingang 1 angelegte Spannung als entsprechender Strom am Analogausgang ausgegeben wird. Die Eingangsspannung des Analogeingangs ist mit dem Potentiometer zwischen 0 und 5 V einstellbar. Der Meßbereich des Channels ist so einzustellen, daß sein Ausgangswert der angelegten Spannung entspricht. Der Stromausgang ist so zu parametrieren, daß der Bereich des Spannungssignals genau auf das life-zero-Signal des Stromausgangs abgebildet wird. Ermitteln Sie anhand des vorbereiteten Blockschaltbildes aus der Vorbereitungsaufgabe 5.9. die notwendigen Registerbelegungen der jeweiligen Channels.

d) Schreiben Sie die Werte in das Universalmodul, und überprüfen Sie die Funktionsfähigkeit der Konfiguration. Vergleichen Sie den durch das Meßgerät angezeigten Strom mit dem angelegten Spannungswert. Ändern Sie die Skalierung so, daß der Vollausschlag des Stromsignals einer Spannung von 10 V entspricht. Überprüfen Sie die Wirksamkeit der Änderung.

e) Erweitern Sie die Funktionalität des Universalmoduls dahingehend, daß Alarme bei Eingangsspannungen von 1 V bzw. 4 V generiert werden. Führen Sie durch Änderung der Eingangsspannung mit dem Potentiometer Alarmsituationen herbei, und werten Sie mit Hilfe des Monitors die entsprechenden Alarminformationen aus. Wie ist die Anzeige der Alarminformationen mittels LED an Digitalausgängen realisierbar?