Herausgegeben von der International P-NET User Organization 590 005 01

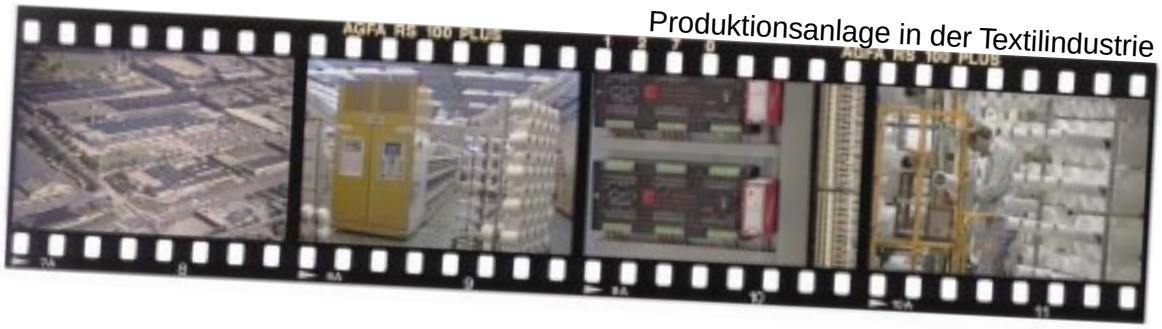
Der P-NET Feldbus

für die Prozeßautomation

P-NET
- der Europäische Feldbusstandard
EN 50170, Band 1

P-NET-Applikationen in der Praxis

Produktionsanlage in der Textilindustrie



Endtestlinien in der Geschirrspülerfertigung bei Bosch-Siemens

Fortgeschrittene Gebäudeautomation

Tierfütterung & Klimasteuerung

Allgemeines zu P-NET

P-NET wurde entwickelt, um verteilte Prozeßkomponenten wie Prozeßcomputer, intelligente Sensoren, Aktoren, Ein-/Ausgabemodule, Feld- und Zentralcontroller, Speicherprogrammierbare Steuerungen etc. über eine gemeinsame Zweidrahtleitung, wie in Abb. 1 gezeigt, zu verbinden.

Dies Feldbustechnik ersetzt die konventionelle Verdrahtung mit ihrem sehr hohen Verdrahtungsaufwand.

Prozeßdaten (wie Meßwerte, Ventilsignale) werden digital übertragen. P-NET wird darüber hinaus für Datenerfassung, für die Konfiguration von Modulen/Sensoren und für den Download von Programmen benutzt.

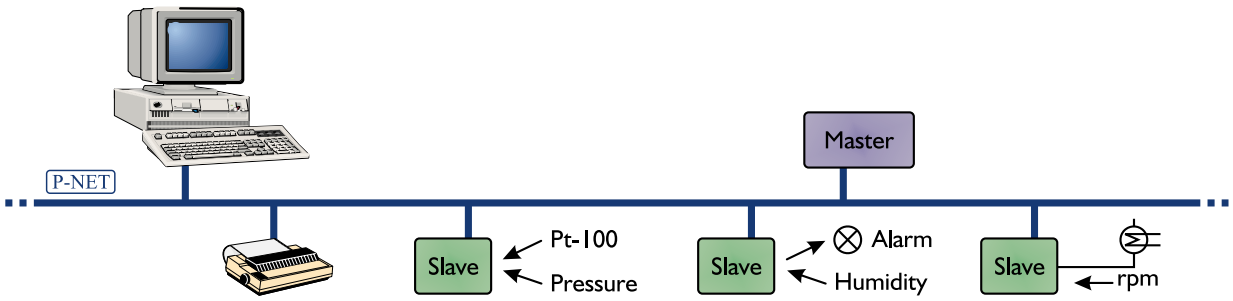


Abb. 1: P-NET mit verteilten Prozeßkomponenten

Unabhängig von der reinen Übertragung von Meßwerten und Statusinformationen erlaubt der Bus den bidirektionalen Austausch zusätzlicher Informationen wie Grenzwerte, Aktorpositionen und -feedback-Signale, Fehlermeldungen und interne Systemdaten.

P-NET kann auch zum Download von Parametern und Programmen in Module herangezogen werden, um dezentral den Prozeß zu steuern. Die Anwendung intelligenter P-NET-Sensoren und -Aktoren ermöglicht darüber hinaus wesentlich verbesserte Diagnoseeigenschaften als bei konventioneller Verkabelung.

Vergleiche mit herkömmlicher Verdrahtung zeigen, daß die Anwendung von P-NET noch einige weitere nachweisbare Vorteile in der Automatisierung industrieller Prozesse mit sich bringt.

Dies sind insbesondere eine Vereinfachung der Planung und Installation, eine Reduzierung des Aufwandes und der Kosten beim Verkabeln, eine Reduzierung der Installations- und Wartungskosten und eine deutliche Abnahme von Installationsfehlern, was eine zielgerichtete Lösung zukünftiger Automatisierungsaufgaben ermöglicht. Wichtige Informationen über Feldgerätefehler und Fehler am Kabel können automatisch vom Busprotokoll detektiert werden.

P-NET-Applikationen sind durch niedrige Kosten gekennzeichnet. Diese steigen nur linear mit der Größe des Systems.

P-NET eignet sich sowohl für kleine Anlagen als auch für große, die über viele Controller, Sensoren und Interface-Module verfügen. Zusätzlich ist jedes dieser Systeme jederzeit für eine erforderlich werdende Erweiterung bereit.

Die Geschichte von P-NET

P-NET wurde 1983 eingeführt. Das erste Produkt, das diesen Multimaster-Feldbus nutzte, wurde 1984 auf den Markt gebracht.

Die Multi-Network und Multi-Port-Funktionen wurden zur Protokollspezifikation 1986 hinzugefügt. 1987 wurde das erste P-NET Multi-Port-Produkt produziert.

Der P-NET-Standard wurde 1989 zu einem offenen und kompletten Standard, der weltweit adaptiert werden kann.

Im Zuge des steigenden Interesses an P-NET wurde die International P-NET User Organization ein Jahr später ins Leben gerufen.

Anwendungsbereiche

P-NET wird seit vielen Jahren benutzt. Weltweit sind mehr als 5.000 Installationen zur Zeit in Betrieb. Die Anwendungen reichen von einfachen Installationen mit einigen wenigen I/O-Punkten zu sehr großen und komplexen Installationen mit mehreren tausend I/O-Punkten.

P-NET-Applikationen finden sich sowohl in der Prozeßtechnik als auch der Stückgutfertigung.

Die folgenden typischen Beispiele zeigen, wo P-NET zur Zeit installiert ist: Molkereien, Brauereien, Umweltmonitoring in der Agrarwirtschaft, Tierfütterungssysteme, Asphalt- und Betonproduktion, Textilindustrie, Milch-/Öl-/Schmierstoff-Verteilerfahrzeuge, Qualitätskontrollsysteme, Kraftwerke, Solaranlagen, Plastikformung, Schiffsmaschinen-Überwachung, Tankmanagement- und -alarmsysteme (zugelassen von German Lloyd, Bureau Veritas, Norske Veritas, Lloyds Register of Shipping), Datenerfassung, Wasserversorgung, Gebäudeautomatisierung, Brennstoff-Füllanlagen (zugelassen für den amtlichen Handel von PTB, NMI, NWML, ...).

Eine typische P-NET Anwendung weist Antwortzeiten bis hinunter zu einigen ms und eine Buslänge bis zu einem oder mehren km auf. Für Applikationen, bei denen die Antwortzeiten im Mikrosekunden-Bereich liegen, ist P-NET dagegen nicht geeignet.

Grundprinzipien von P-NET

Die elektrische Spezifikation von P-NET basiert auf dem RS-485-Standard und benutzt eine geschirmte Zweidrahtleitung. Dies erlaubt eine Kabellänge von bis zu 1.200 m ohne Repeater. Daten werden asynchron im NRZ-Code übertragen.

P-NET-Schnittstellen sind galvanisch isoliert, wobei aufgrund spezieller Clamp-Schaltkreise bis zu 125 Geräte innerhalb eines Bussegmentes ohne den Einsatz von Repeatern angeschlossen werden können.

P-NET ist ein sehr effizientes Feldbusprotokoll, was dazu führt, daß bis zu 300 bestätigte Datentransfers pro Sekunde - auch mit 300 voneinander unabhängigen Adressen - durchgeführt werden können.

Daten können sowohl als komplette Prozeßwerte (Gleitpunktzahl) wie Temperatur, Druck, Strom, Spannung etc. als auch als Blöcke von typ. 32 unabhängigen binären Signalen, die zum Beispiel Ventilzustände oder Schalterpositionen anzeigen, übertragen werden.

Dies führt zu einer Leistungsfähigkeit von bis zu 9.600 binären Signalen pro Sekunde, die von jeder beliebigen Stelle im gesamten System adressiert werden können.

Diese hohe Rate an voll bestätigten Datentransfers kann erreicht werden, da P-NET-Slaves die Verarbeitung von Daten und die Übertragung von Datenrahmen parallel handhaben. Die

Bearbeitung einer Anfrage wird vom Slave gestartet, sobald das erste Datenbyte empfangen wurde. Dies unterscheidet sich wesentlich von dedizierten Chiplösungen, bei denen der komplette Datenrahmen eingelesen sein muß, bevor die Verarbeitung beginnen kann. Unter diesem Gesichtspunkt ist die bei P-NET standardmäßig verwendete Bitrate von 76.800 bit/s kein die Performance limitierender Faktor.

Die Leistungsfähigkeit kann verglichen werden mit herkömmlichen Systemen, die Datenraten bis zu 500.000 bit/s benutzen. Vgl. dazu auch die detaillierte Beschreibung auf Seite 17 „P-NET im Vergleich mit dedizierten Feldbus-Chip-Lösungen“.

P-NET ist ein Multi-Master-Bus, der bis zu 32 Master pro Bussegment erlaubt. Jegliche Kommunikation basiert auf dem Prinzip, daß ein **Master** eine Anfrage sendet und der adressierte **Slave** sofort eine Antwort zurückgibt. Anfragen können vom Typ „Lesen“ oder „Schreiben“ sein. Typische Masters und Slaves sind in Abb. 2 aufgeführt.

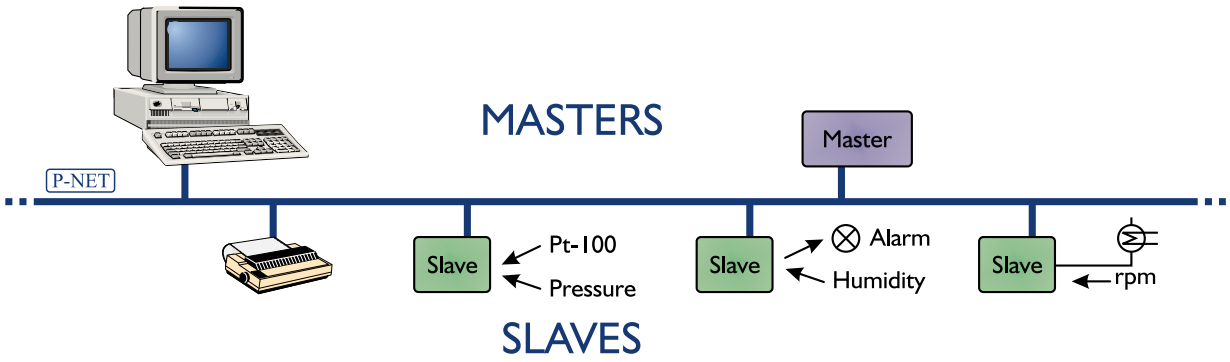


Abb. 2: Master und Slaves in einem P-NET-System

Auf dem Bus übertragene Daten können einen einfachen oder komplexen Datentyp aufweisen, um den jeweiligen Anforderungen beim Messen und Regeln gerecht zu werden. Einfache Datentypen sind boolean, byte, char, word, integer, long integer, real, long real und timer. Komplexe Datentypen sind array, string, record und buffer.

Das Datenformat ist Bestandteil des P-NET-Standards.

Das Recht, auf den Bus zuzugreifen, zirkuliert von einem Master zum nächsten unter Verwendung eines Tokens. P-NET benutzt dabei eine „virtuelles token passing“ genannte Methode, die keine über den Bus zu versendenden Nachrichten benötigt.

Wenn ein Master seinen Buszugriff beendet hat, wird der Token automatisch an den nächsten Master durch einen zyklischen, Zeit-basierten Mechanismus weitergereicht. Diese Methode unterscheidet sich von der in anderen Multi-Master-Systemen verwendeten.

Andere Busse wie Profibus beispielsweise benutzen echte Nachrichtentelegramme für die Weitergabe des Token. Dies führt zu einer Zunahme der Verarbeitungszeiten im Master und reduziert die Kapazität des Busses.

Das Verfahren des virtuellen Tokens toleriert auch den Fall, daß ein Master gerade nicht angeschaltet ist. In dieser Situation arbeiten alle Geräte, eingeschlossen der anderen Master, normal weiter. Vgl. Seite 16 „Virtuelles Token Passing“ für eine genauere Beschreibung.

Multi-Net-Strukturen

Der früher übliche Weg, eine Netzwerk-Architektur für eine Fabrik zu entwerfen, bestand darin, den Feldbus direkt mit den Sensoren und Aktoren zu verbinden. Der Feldbus wurde dann mit einem Zellen-Controller verbunden und eine bestimmte Anzahl von Zellen-Controllern wurden zu einem Zellen-Netzwerk zusammengeschaltet und so weiter, bis nach mehreren hierarchischen Ebenen schließlich ein Hochgeschwindigkeits-Backbone-Netzwerk alles miteinander verknüpft. Die benötigte Datenrate wuchs dabei beim Übergang von einer Netzwerkebene auf die nächst höhere um etwa eine Größenordnung.

Dies war allenfalls in der Vergangenheit eine akzeptable Philosphie, bei der alle Daten gegebenenfalls auch in einem leistungsfähigen Computer auf der höchsten Ebene verfügbar sein mußten. Die Technik für heute und morgen dagegen baut auf eine verteilte Intelligenz zwischen den Zell-Controllern, den Interfaces und den Sensoren. Auf jeder Ebene werden die Daten konzentriert und Regelschleifen werden typischerweise innerhalb desselben Busses geschlossen.

Die Erfordernis einer höheren Datenrate auf den höheren Ebenen nimmt nun ab, da mehr verteilte Intelligenz zur Verfügung steht. Das ist der Grund, wieso P-NET auf verschiedenen Ebenen innerhalb eines kompletten Fabrikautomationssystems benutzt werden kann.

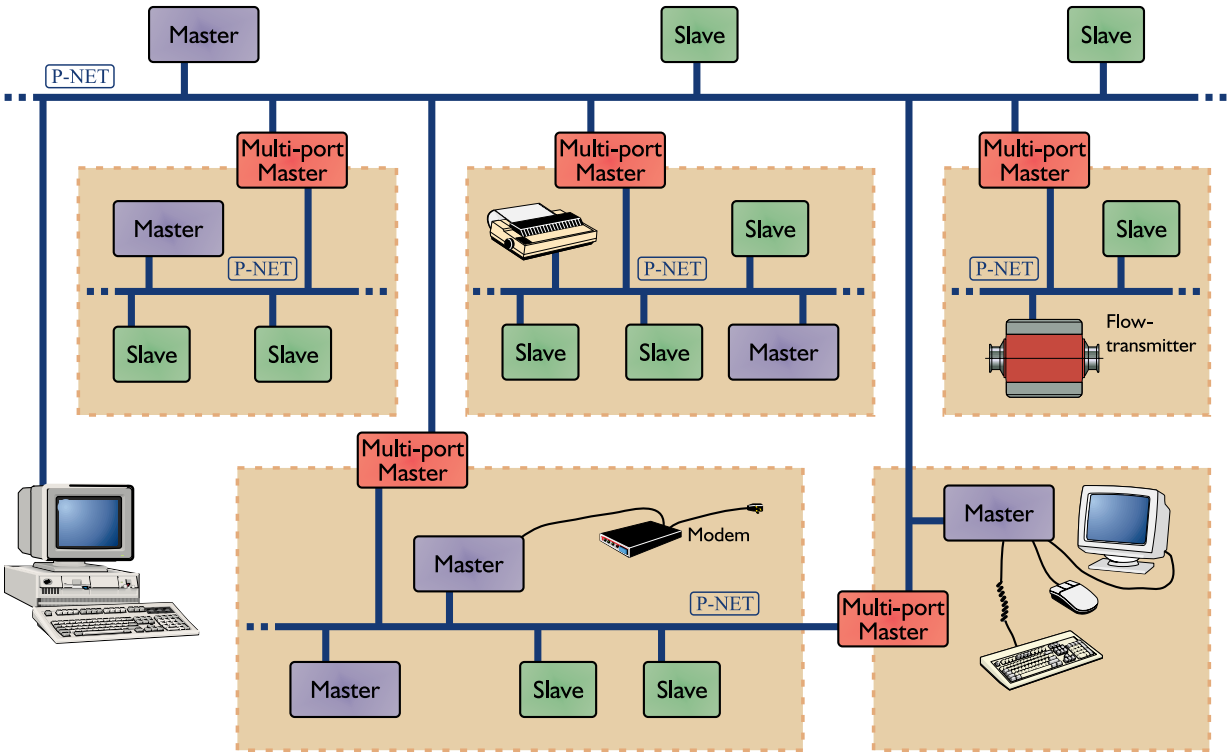


Abb. 3: Eine Multi-Net-Struktur mit P-NET

Indem ein System in einzelne Zellen, die jeweils bestimmten Abschnitten einer Anlage entsprechen, unterteilt wird, kann durchaus eine Zelle ausfallen, ohne die anderen in ihrer Funktion zu beeinflussen. Die Programmausführung kann innerhalb einer Zelle auf einen oder mehrere Prozessoren verteilt werden.

Ein Software- oder Hardware-Fehler in einer Zelle beeinflußt die anderen Zellen nicht. Der Datenaustausch zwischen einzelnen Zellen beschränkt sich zum Beispiel auf den Start und das Abbrechen von Prozessen, das Laden von Rezepten, die Übertragung von Produktionsdaten etc.

In Systemen mit einer echten Verteilung der Intelligenz kann zusätzliche Rechenleistung immer durch einfaches Hinzufügen zusätzlicher Master-Controller erzeugt werden. Ein System dieser Struktur kann deshalb jederzeit erweitert werden.

Unter den verfügbaren Feldbussystemen erlaubt nur P-NET die direkte Adressierung zwischen verschiedenen Bussegmenten, was auch unter dem Begriff Multi-Net-Struktur bekannt ist. Diese Eigenschaft ist spezifizierter Bestandteil des P-NET-Protokolls and kann in das Standard-Betriebssystem von Multi-Port-Mastern integriert werden. Eine Multi-Net-Struktur ist in Abb. 3 dargestellt.

Kommunikationsdaten werden von Knoten mit zwei oder mehreren P-NET-Schnittstellen durch die verschiedenen Bussegmente gereicht. Dies bedeutet, daß jeder Master eines Bussegments transparent auf jeden Knoten in einem beliebigen anderen Bussegment zugreifen kann, ohne daß in den Multi-Port-Mastern spezielle Programme ablaufen müssen (vgl. Abb. 4).

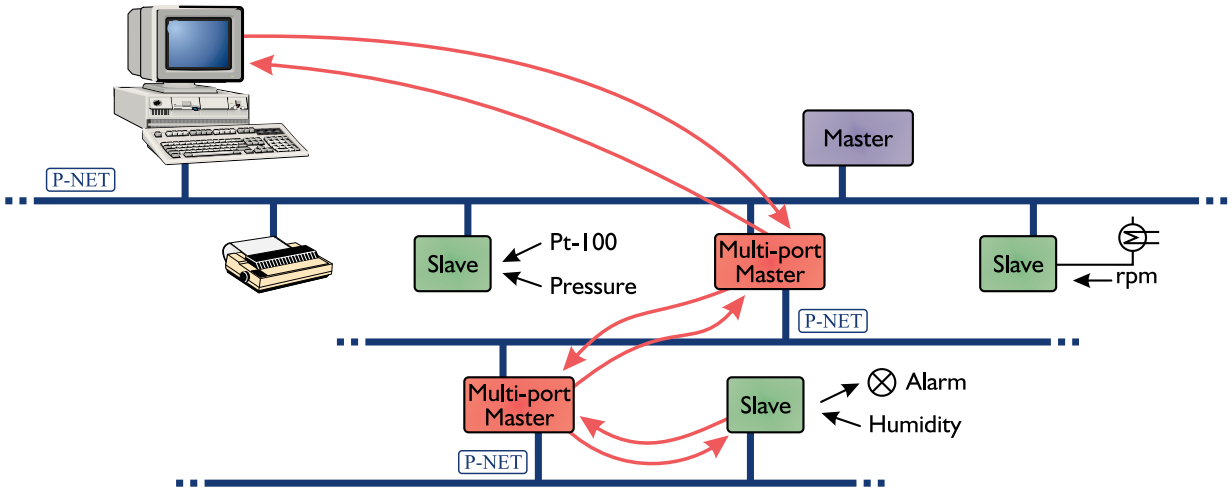


Abb. 4: Transparenter Zugriff auf andere Bussegmente über Multi-Port-Master

Die Segmentierung erlaubt auch einen unabhängigen lokalen Datenverkehr in jedem Bussegment, was die Zykluszeiten und den Datendurchsatz im gesamten System optimiert.

Die Vorteile, die man durch Unterteilung eines Systems in kleinere Abschnitte erhält, sind insbesondere deshalb sehr signifikant, weil dies die Folgen eines Fehlers auf ein einzelnes Segment begrenzt, was eine höhere Systemsicherheit zur Folge hat. Weiterhin implementieren diese Multi-Net-Strukturen eine natürliche Redundanz, die das gesamte Netzwerk sehr robust in bezug auf Fehler macht (vgl. auch Abb. 3). Ein weiterer wichtiger Vorteil der P-NET-Multi-Net-Topologie ist, daß eine hierarchische Strukturierung der Bussegmente nicht notwendig ist. Dies ist ein großerVorteil, wenn existierende P-NET-Installationen erweitert bzw. an andere Netzwerke angekoppelt werden.

Der Versuch, zwei Segmente innerhalb eines Knotens zu koppeln, erfordert bei Bussystemen **ohne** diese Multi-Net-Eigenschaft ein spezielles Programm im Knoten. Ein derartiges Programm muß alle Daten von allen Geräten eines Segments einlesen, um sie für das andere Segment verfügbar zu machen, eine Technik, die als „Prozeßabbild“ bekannt ist.

Bei der hohen Anzahl der heute in intelligenten Knoten verfügbaren Daten ist es beinahe unmöglich, ein komplettes „Prozeßabbild“ für ein Bussegment ständig aktuell zu halten und zu pflegen. Eine derartige Prozedur benötigt einen signifikanten Anteil der gesamten Buskapazität und einen großen Speicherplatz. Weiterhin ist es sehr teuer, dedizierte Programme für jede Segmentverbindung zu entwickeln und zu testen.

Mit P-NET müssen derartige komplexe „Prozeßabbilder“ nicht erzeugt werden.

Vorteile des P-NET-Protokolls

Alle Knoten, die dem P-NET-Standard entsprechen, können direkt an den Bus angekoppelt werden und sofort miteinander kommunizieren, da P-NET nur eine Datenrate benützt und für jede Kommunikationsebene nur eine Variante zulässig ist.

Dies unterscheidet sich von anderen Standards, die viele Varianten auf jeder Ebene zulassen, was in vielen Kombinationen, die untereinander nicht kommunizieren können, endet.

Jeder P-NET-Knoten, eingeschlossen die Master, kann abgeschaltet werden oder an den Bus an- bzw. von ihm abgeklemmt werden, ohne das verbleibende Bussystem zu beeinflussen.

Als Konsequenz daraus können Module während des Betriebs ausgetauscht werden und ein System kann erweitert werden, während das verbleibende Produktionssystem den Betrieb aufrecht erhält.

Die Erfordernis, Kommunikationsparameter zu konfigurieren, ist bei P-NET imVergleich zu anderen Systemen stark reduziert worden. In Slave-Modulen muß der P-NET-Systemintegrator nur die Moduladresse setzen. In Master-Modulen sind lediglich die Knotenadresse und die Gesamtzahl der Master zu setzen.

Daher ist nur ein kurzes Training erforderlich, damit ein qualifizierter Techniker P-NET-Systeme verstehen und installieren kann.

Die verteilte Rechenleistung eines Systems kann durch einfaches Hinzufügen zusätzlicher Master erhöht werden.

Es wurden spezielle Prozeduren in den P-NET-Standard aufgenommen, die es ermöglichen, die Adresse eines einzelnen Knotens im Netzwerk unter Verwendung seiner speziellen Seriennummer zu ändern. Dies erlaubt, daß individuelle P-NET-Knoten-Adressen geändert werden, während das System weiterläuft.

DIP-Schalter und andere mechanische Mechanismen sind nicht erforderlich, weshalb hermetisch dichte P-NET-Knoten gebaut werden können (z.B. IP-67).

Während der Entwicklung eines neuen P-NET-Gerätes kann es sich als Vorteil erweisen, daß über P-NET jede logische und physikalische Adresse innerhalb des Gerätes - je nach Bedarf des Herstellers - angesprochen werden kann. Wenn in einem Gerät P-NET implementiert wird, können sowohl die Testprozeduren, die innerhalb der Entwicklung des Anwenderprogrammes durchgeführt werden müssen, als auch Kalibrierungs- und Wartungsprozeduren in der Zukunft vereinfacht werden. P-NET kann daher benutzt werden, um in die Geräte hineinzusehen und dabei Programmvariable zu überwachen.

Das Ergebnis einer von einem Slave durchgeführten Messung wird einem Master in einer vordefinierten Form, nämlich in metrischen SI-Einheiten übermittelt. Der Vorteil ist offensichtlich, da vom Master keine wiederholten Skalierungs- und Umwandlungsprozeduren durchgeführt werden müssen, was zu beträchtlichen Einsparungen an Rechenleistung führt. So führt beispielsweise eine Temperaturmessung direkt zu einer Gleitpunktzahl (IEEE 754-Standard) im Slave und wird gegenüber allen Mastern in Grad Celsius dargestellt.

Sogenannte Identifiers, die für den Zugriff auf physikalische Netzwerkvariablen benutzt werden, sind in einer „SOFTWIRE“-Liste aufgeführt. Diese Liste wird automatisch generiert, wenn Applikationsprogramme compiliert werden. Daher sind keine Echtzeit-Übersetzungen notwendig, was zu einem sehr schnellen Datenzugriff führt.

Um Echtzeit-Datenerfassung zu garantieren, ist jeder Datenrahmen, der über das Netzwerk übertragen wird, auf 56 Datenbytes begrenzt. Wenn die angefragte Datenlänge größer als 56 Bytes ist, werden die Daten automatisch auf mehrere nachfolgende Übertragungen aufgeteilt.

Intelligente P-NET-Module

Typische P-NET-Slave Module ermöglichen dem Systemintegrator mehr als eine reine Ein- und Ausgabe von Daten. Sie verfügen sehr oft über zusätzliche prozeßnahe Funktionen, von der einfachen Grenzwertabfrage über PID-Regler bis zu Programmkanälen, die dem Systemintegrator die Implementierung lokaler Regelschleifen oder spezieller Prozeßschritte erlauben.

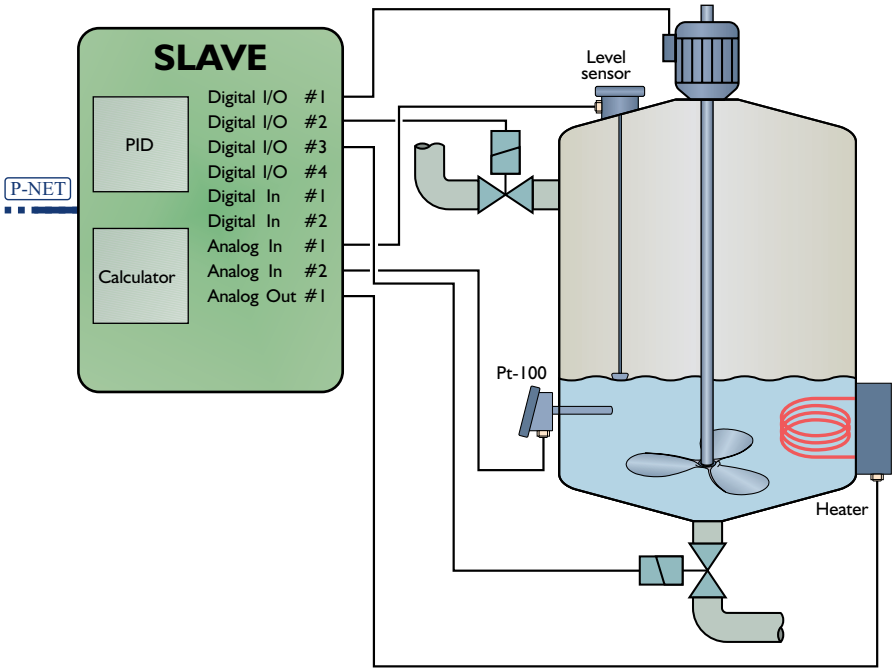


Abb. 5: Ein intelligentes P-NET-Modul in einer chemischen Anlage

Das Diagramm in Abb. 5 zeigt, wie ein Standard-P-NET-I/O-Modul die Temperaturregelung und Produktzu- und abfuhr für einen Heizkessel in einer chemischen Anlage übernimmt.

In diesem Beispiel übernehmen die internen Prozeßfunktionen des Moduls die Temperatur- und Füllstandsregelung sowie die Steuerung des Befüllvorganges. Lediglich die Sollwerte für Temperatur und Füllstand werden von einem Master vorgegeben.

Ein weiteres Beispiel für ein Slave-Modul ist ein Wägetransmitter, bei dem das analoge Signal von der Wägezelle kontinuierlich umgewandelt, skaliert und im Speicher des Slaves abgelegt wird. Wenn vom Master eine Anfrage getätigt wird, antwortet der Slave sofort mit dem letzten gespeicherten Wert. Weiterhin wird durch den Slave eine kontinuierliche Fehlererkennung durchgeführt. Tritt ein Fehler auf, wird der Master durch einen speziellen Code innerhalb des Antworttelegramms, das der nächsten Masteranfrage folgt, darüber informiert.

„Layer 8“: P-NET-Channel-Struktur

Typische P-NET-Geräte sind Sensoren, Aktoren oder Interface-Module. Sie können über ein oder mehrere Prozeßsignale wie digitale Ausgänge oder analoge Eingänge verfügen. Jedem Prozeßsignal sind neben dem reinen Signalzustand bzw. -wert zusätzliche Informationen zugeordnet. Diese den Prozeßsignalen zugeordneten Variablen betreffen spezielle Funktionen wie Konfiguration, Umrechnung, Skalierung, Filterung, Fehlermeldungen etc.

In P-NET wird die Gesamtheit der auf ein Prozeßsignal bezogenen Variablen und Funktionen als **Prozeßobjekt** behandelt und trägt den **Namen Kanal** (Channel).

Ein Channel beinhaltet alle notwendigen Daten, um die für ein Prozeßobjekt verfügbaren Steuerfunktionen auszuführen. Er enthält außerdem Daten, welche die Wartung und das technische Management der Anlagenausrüstung unterstützen.

Ein Channel ist strukturiert als Folge von 16 Registern, wobei jedes seine eigene logische Adresse, genannt SOFTWIRE number (SWNo), hat.

Diese 16 Variablen bzw. Konstanten eines Channels können beliebige, auch komplexere Datentypen aufweisen, und in verschiedenen Speichertechnologien ausgelegt werden.

Um ein spezielles Beispiel eines standardisierten Interface-Channels zu geben, zeigt Abb. 6 einen digitalen I/O-Channel.

Ein derartiger Channel kann für verschiedene Funktionalitäten einschließlich automatischer Funktionen konfiguriert werden. Diese Funktionen sind beispielsweise einfaches Einlesen, Ausgabe, One Shot Output, Timer Output etc.

Die jeweilige Funktion wird durch Setzen eines Codes im ChConfig-Register ausgewählt. Wird der Ausgang für timer-Funktionen konfiguriert, werden zusätzlich die Preset-Register SWNo x7 und x8 benutzt.

Wenn der Input/Output-Pin aktiv ist, mißt das OperatingTime-Register die Zeit. Ein Zustandswechsel am Pin inkrementiert das Counter-Register, um Ein- bzw. Ausgangsaktivitäten aufzuzeichnen.

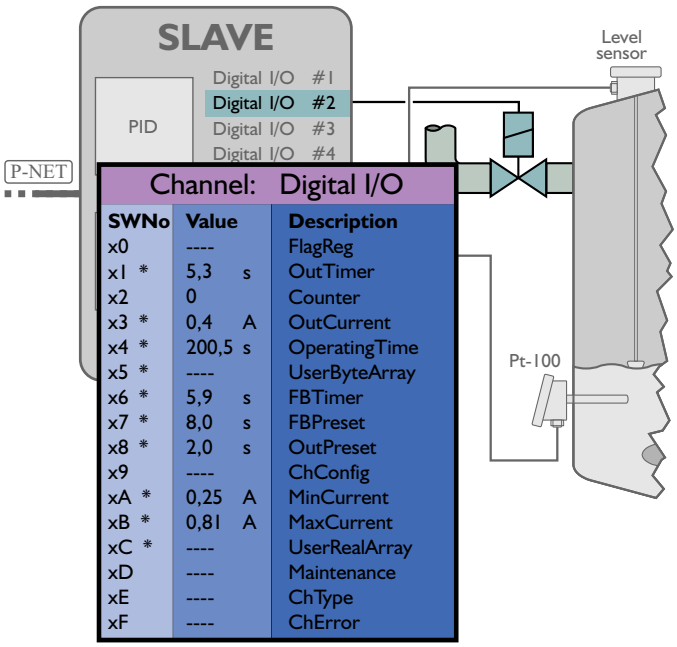


Abb. 6: Die Channel-Struktur eines digitalen I/O-Signals

Der Strom am Ausgang wird gemessen und kann von SWNo x3 eingelesen werden.

MinCurrent und MaxCurrent können als eine Art Feedback-Signale benutzt werden, um zu erkennen, ob ein Verbraucher angeschaltet ist und um den Ausgang sowie den Verbraucher vor Überlast zu schützen.

Das Maintenance-Register enthält Informationen darüber, wann und mit welchem Ergebnis die letzte Wartung beispielsweise des angeschlossenen Ventils durchgeführt wurde. Das Register mit dem Namen ChType muß in allen Channels vorhanden sein. Es stellt einen Record dar, der eine Zahl, die den Channel-Typ wiedergibt, sowie ein logisches Array, welches die tatsächlich vorhandenen Register des Channels anzeigt, enthält.

Die Register, die mit einem "★" markiert sind, müssen nicht immer implementiert werden sondern können auch als nicht benutzt gekennzeichnet werden.

Eine der wichtigen Eigenschaften von P-NET ist die Behandlung von Fehlermeldungen. Jeder Channel verfügt über ein Fehlercode-Register, das ChError genannt wird. Dieses Register beinhaltet auf den Channel und die Werte in seinen Registern bezogene Fehlerinformation.

Beispiele für Fehler sind Überlast, Fühlerbruch etc.

Ein Channel muß nicht notwendigerweise mit Prozeßsignalen korrespondieren, er kann sich auch auf jegliche andere Art von Daten wie zum Beispiel interne Funktionsblöcke mit entsprechenden Parametern beziehen.

Ein Beispiel eines derartigen Channels ist ein PID-Regler, bei dem die Ausgangswerte Resultat eines Algorithmusses sind. Andere standardisierte Channel-Typen sind Drucker-Channel, Kommunikations-Channel, Programm-Channel etc.

Dies bedeutet, daß Knoten unterschiedliche I/O-Strukturen aufweisen können. Beispielsweise kann ein Knoten über 16 digitale I/O- und 2 analoge I/O-Channels verfügen, während ein anderer Knoten 8 digitale I/O- und 4 analoge I/O-Channels hat. In jedem Fall wird jeder einzelne Ein-/ Ausgang jedoch vom Master aus in genau gleicher Weise betrachet, unabhängig davon, zu welchem Typ von Knoten er gehört.

Der **Service Channel** ist ein weiterer sehr wichtiger standardisierter Channel, der in alle Knoten implementiert werden muß, egal ob es sich um eine komplexe Ansammlung verschiedenster Channels oder um einen einfachen Sensor handelt.

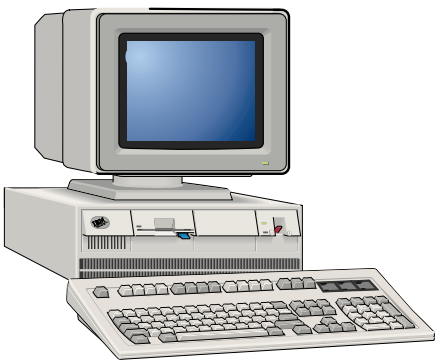
Dieser Channel beinhaltet Informationen über die Knotenadresse, die Seriennummer, Herstellerangaben, auf das Gesamtmodul bezogene Fehlerdaten sowie weitere, das gesamte Modul betreffende Angaben. Die SWNo dieses Channels beginnen immer mit einer 0, weshalb der Zugriff auf den Service Channel für alle Knoten stets derselbe ist. Dieser Channel wird auch benutzt, um unbekannte Knoten zu identifizieren.

Der aus dieser Philosphie der standardisierten Channels resultierende Vorteil ist, daß vom Master aus gesehen, jeder Channel in derselben Art und Weise behandelt werden kann, unabhängig davon, wer das Gerät gefertigt hat oder in welchem Knoten er sich befindet.

Diese Standardisierung ermöglicht außerdem die Entwicklung allgemeingültiger Programme, die zur Konfiguration derartiger Channel-Typen bzw. zum Lesen oder Beschreiben der Register benutzt werden können, unabhängig vom Ort des Channels.

PC-Zugriff auf P-NET

PCs werden in P-NET-Installationen häufig als einer der Master benutzt. Sie werden im Normallfall über Einsteckkarten an P-NET angekoppelt.



Speziell für diese Belange wurde ein Produkt mit dem Namen VIGO entwickelt.

VIGO ist ein PC-basiertes Feldbus-Management-System. Es erlaubt die Beschreibung einer prozeßtechnischen Anlage in Form von Daten, bezogenen Datenstrukturen und Datenorten.

VIGO stellt außerdem ein Kommunikationssystem dar, das die Datensicherheit und die Datenintegrität für Datenanfragen innerhalb einer Anlage sicherstellt.

VIGO zieht den Faden zwischen physikalischen Objekten innerhalb einer Anlage und den zugeordneten Feldbusknoten. Es enthält auch ein System von Dateien, das zugeordnete Steuerprogramme, Konfigurations- und Kalibrierungsparameter, aber auch Tools für die Konfiguration, den Backup, den Download etc. enthält.

Die Weitervermittlung und Behandlung verschiedener gleichzeitiger Informationspakete für dieselben oder für verschiedene Netzwerke werden ebenfalls von VIGO durch einen Echtzeit-Kommunikationskern übernommen. Wenn verschiedene Applikationen zur gleichen Zeit den gleichen Bus benutzen wollen, entstehen in üblichen Windows-Multitasking-Umgebungen Zugriffsprobleme. Diese werden durch VIGO gelöst, indem es sicherstellt, daß Kommunikationspakete und -nachrichten nicht miteinander vermischt werden.

Eine Einrichtung, die den Echtzeit-Datenaustausch zwischen MS-Windows-Applikationen ermöglicht, heißt „OLE2 Automation“ (Object Linking and Embedding).

VIGO ist ein OLE2 Automation-Server, der eine konsistente und transparente Schnittstelle zwischen Benutzerprogramm (Applikation) und physikalischen Elementen (Objekten) innerhalb einer Anlage erzeugt.

Auf diese Weise stellt VIGO ein einfaches Interface für Standard-Programm-Pakete wie Visual Basic und Visual C++, Tabellenkalkulationen, Datenbanken, Mensch-Maschine-Schnittstellen und anderen Visualisierungsprogrammen bzw. SCADA-Anwendungen zur Verfügung.

Im folgenden ist ein Visual Basic-Beispiel (EXCEL-Makro Sprache) aufgeführt, das aus drei Schritten besteht:

Schritt 1: ***set AA = CreateObject(„VIGO“)***

VIGO erzeugt ein virtuelles Objekt AA, das daraufhin Teil der Visual Basic-Programmierumgebung wird.

Schriit 2: ***AA.PhysID = „Setpoint“***

Das virtuelle Objekt AA zeigt nun auf das physikalische Objekt, indem der physikalische Identifier einem Eigenschaftsfeld mit dem Namen PhysId zugewiesen wird.

Schritt 3: ***X = AA.ExFloat*** (Get Setpoint)

oder ***AA.ExFloat = 37,0*** (Set Setpoint)

Jede Manipulation am physikalischen Objekt wird über das virtuelle Objekt durchgeführt (vgl. Abb. 7).

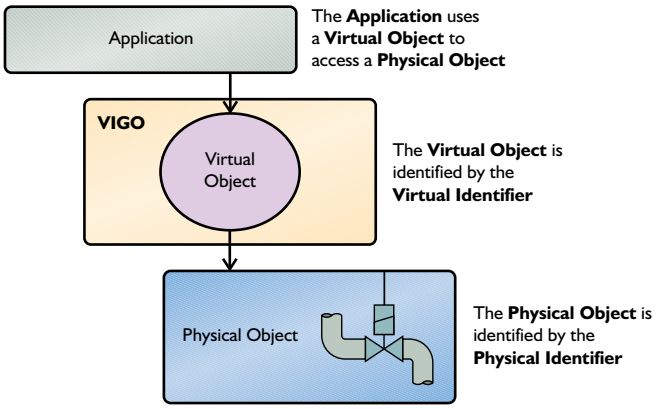


Abb. 7: Virtuelles und physikalisches VIGO-objekt

Um mit einem Objekt zu arbeiten, muß die Typeigenschaft belegt werden, um den Typ der Variable im Knoten anzugeben. ExFloat zeigt dabei an, daß die Objektvariable vom Typ real (Gleitpunkt) ist, ExBool würde einen logischen Typ angeben etc.

Das Objekt kann innerhalb normaler Anweisungen wie set- oder get-Funktionen benutzt werden. Es können viele Objekte für verschiedene unabhängige Applikationen erzeugt werden.

VIGO ist eigentlich eine Ansammlung verschiedener Programmelemente. Es ist ein offenes System, das die Hinzunahme von Netzwerkelementen anderer Hersteller erlaubt. Alle diese Elemente werden von VIGO gehandhabt und auch in dieses integriert, was zu einer sehr einfachen und durchgehend definierten Schnittstelle für jegliche Feldbusdaten führt. Die Elemente von VIGO werden in Abb. 8 gezeigt.

Application: Ein unabhängiges Benutzerprogramm, das über VIGO kommuniziert.

VIGOSERV: Ein OLE2 Automation-Server, der die Schnittstelle zwischen VIGO und den Applikationen darstellt.

IDC: Ein Instruction/Data Converter, der eine VIGO-Anfrage in eine spezifische Netzwerk-Anweisung umwandelt. Weiterhin initialisiert er die Daten in der für den Zielknoten erforderlichen Syntax. Für das Einlesen von Daten gilt das Umgekehrte.

HUGO2: Ein Echtzeit-Kommunikationskern, der die parallele Ausführung verschiedener Feldbus-Kommunikations-Programme erlaubt.

Manager Information Base: Die MIB beinhaltet einen Satz von Datenstrukturen, welche das pysikalische System beschreiben. Die MIB übersetzt einen Variablennamen in eine spezifische Knotenadresse, Variablenadresse, Typspezifikation etc.

Drivers: Diese kümmern sich um das Senden und Empfangen von Informationen über spezifische Netzwerke.

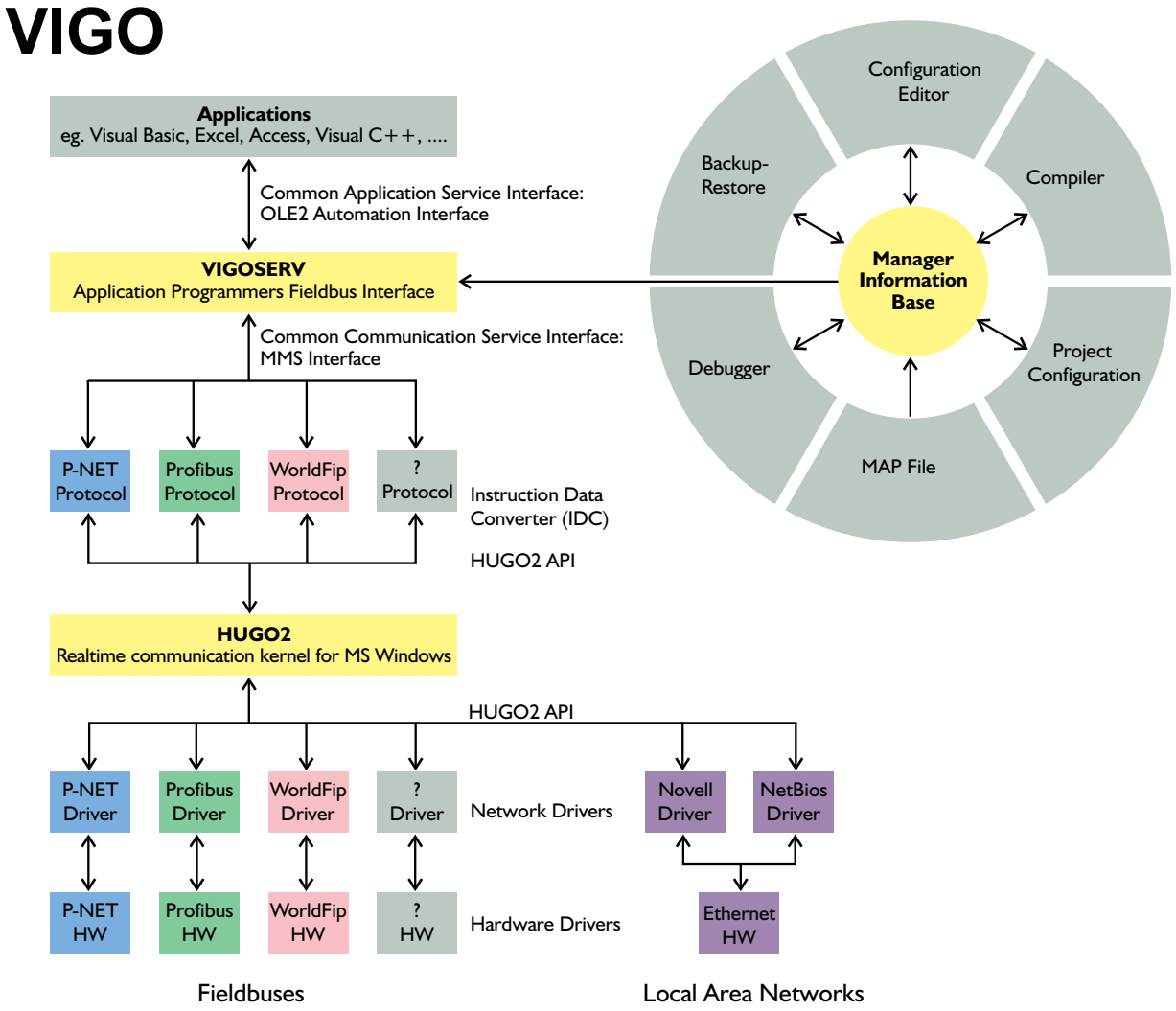


Abb. 8: Die Elemente von VIGO

Software

Neben dem Standardverfahren für den Datenaustausch unter MS-Windows nach OLE2 Automation existieren auch P-NET-Treiber für DDE (Dynamic Data Exchange).

Software Tools für das Monitoring und Debugging, grafische Überwachungssysteme, Tools für das Downloading von Programmen und Konfigurationsdaten, Editoren etc. sind für P-NET verfügbar.

Als Programmiersprache speziell für P-NET-Controller ist Process-Pascal erhältlich, das Standard-ISO-Pascal mit zusätzlichen Elementen für die Deklaration von Netzwerkvariablen und für das

Taskmanagement in Multitasking-Umgebungen darstellt. Programme, die in Process-Pascal geschrieben wurden, benutzen globale P-NET-Variablen genau so, als ob sie lokale Variablen wären. Der einzige Unterschied besteht in der Variablendeklarations-Technik. Die in Process-Pascal enthaltenen Mulittasking-Elemente unterstützen bis zu 64 Tasks pro Master.

Einfache Implementierung von P-NET

Einer der Gründe für die hohe Anzahl von P-NET-Installationen, die zur Zeit in Betrieb sind, ist in den niedrigen Kosten der Implementierung in einen Knoten zu sehen.

Ein Grundprinzip von P-NET besteht darin, denselben Mikroprozessor sowohl für die Implementierung der eigentlichen Anwender-Task als auch der Kommunikationstask zu verwenden. Daten werden immer nur an einem Ort gespeichert. Indem P-NET integrierter Bestandteil des Gerätes ist, kann es zur Konfiguration und zum Auslesen des Gerätestatus benutzt werden.

Dies bedeutet typischerweise, daß DIP-Schalter für die Auswahl einer Baudrate und das Setzen der Knotenadresse nicht benötigt werden (vgl. Abb. 9).

Andere Feldbus-Typen verwenden zusätzliche Schaltkreise in jedem Knoten in Form separater Chips bzw. Mikroprozessoren für die Kommunikation. Daten werden über ein Dualport-RAM ausgetauscht. Dieses Verfahren resultiert immer in signifikant höheren Kosten für das Endprodukt (vgl. Abb. 10).

Für die Implementierung des P-NET-Protokolls besteht **kein Bedarf** an speziellen Chips, da das P-NET-Kommunikations-Protokoll für einen Slave lediglich einige wenige kByte Code beansprucht. Dies ermöglicht die Verwendung weit verbreiteter Single-Chip-Mikroprozessoren, die über eine UART verfügen, wie H8-300, 68HC11, 6805, 80851, 8051 etc.

Man erkennt daran, daß ein P-NET-Feldbusknoten nicht teurer ein muß als ein konventionelles, mit einem Mikroprozessor ausgestattetes Gerät, das keinen Feldbusanschluß hat.

Viele Jahre an Erfahrung existieren mittlerweile in der Implementierung von P-NET-Knoten. Die International P-NET User Organization vermittelt gerne entsprechende Unterstützung.

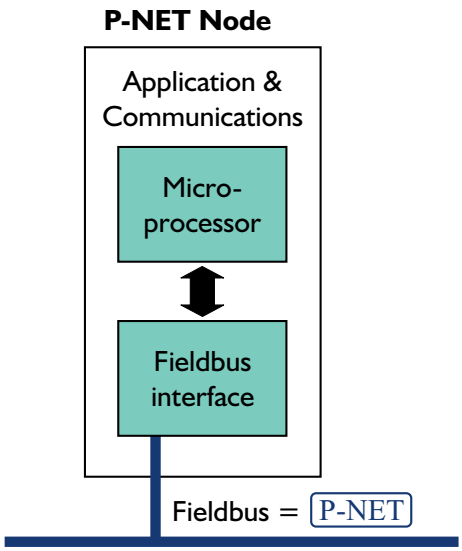


Abb. 9: Implementierung von P-NET

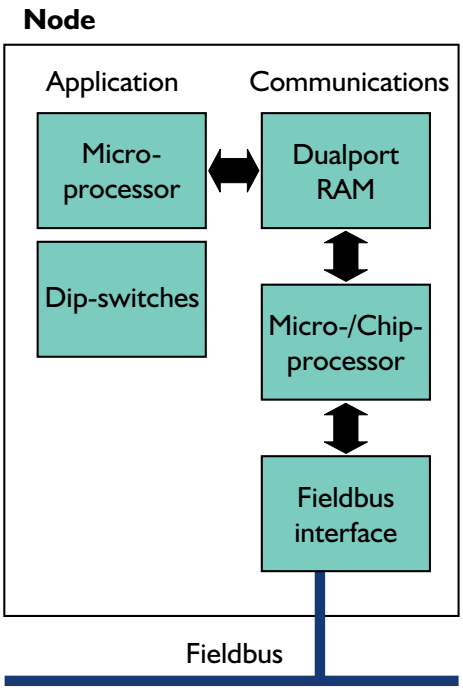


Abb. 10: Typische Chip-Implementierung in anderen Feldbus-Systemen

Die Architektur von P-NET

Die Spezifikation und Implementierung von P-NET richtet sich nach dem Open Systems Interconnection Referenzmodell für die Schichten 1, 2, 3, 4 und 7, wie in Abb. 11 gezeigt.

Üblicherweise implementiert ein Feldbus nur die Schichten 1,2 und 7, da aber P-NET Multinet-Strukturen unterstützt, implementiert das Protokoll auch die Schichten 3 und 4.

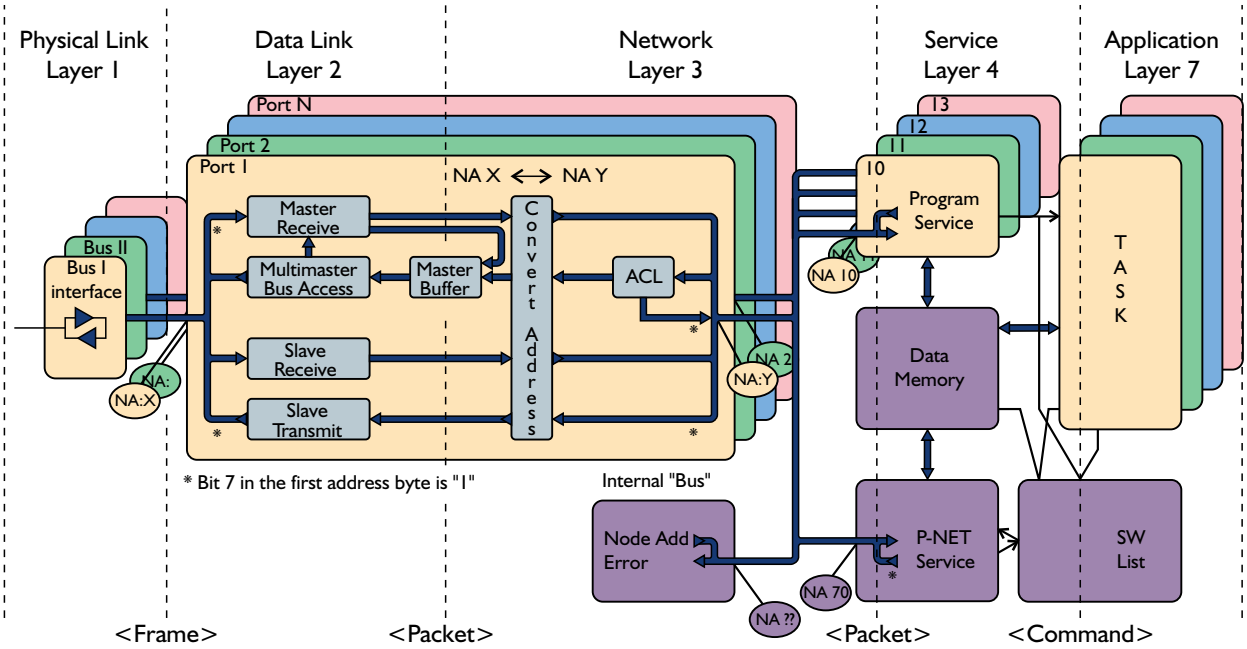


Abb. 11: Die auf dem ISO-Referenzmodell basierte Architektur von P-NET

Schicht 1 behandelt die Übertragung „roher“ Bits über den Bus. Sie spezifiziert das Kabel, wie eine „1“ bzw. „0“ auf dem Bus dargestellt werden, welche Spannungspegel zu beutzen sind etc.

Schicht 2 kümmert sich um den Multimaster-Token, verpackt die zu sendenden Daten in einen Rahmen, fügt Absender- und Zieladressen ein und führt die Fehlererkennung durch.

Die Schícht 3 stellt das P-NET-“Postamt“ dar, indem sie die Rahmen unter Bezugnahme auf die Zieladresse empfängt und aussendet. Eine Nachricht kann je nach Situation aus einem anderen-P-NET-Port heraus oder zu dem sogenannten P-NET-Service oder auch zurück zur anfordernden Applikation bzw. in Form einer Nachricht, die eine unbekannte Adresse anzeigt, zurück gesendet werden. Die Schicht 3 führt weiterhin die Adreßumwandlung durch, die benötigt wird, damit eine Antwort wieder ihren Weg zurück findet.

Schicht 4 umfaßt zwei verschiedene Tasks. Die erste beinhaltet den P-NET-Service, der Daten in den internen Speicher mittels SOFTWIRE-Liste schreibt bzw. herausliest oder eine Anfrage zurückleitet, falls die SOFTWIRE-Liste anzeigt, daß die Variable in einem anderen Modul abgelegt ist. Die zweite Task enthält Details über alle Anfragen, die ausgesandt wurden, jedoch noch auf eine Antwort warten. Wenn die Antwort eintrifft, wird sie zur anfordernden Applikations-Task zurückgesandt.

Schicht 7 wird von den Anwenderprogrammen benutzt, um auf Variable in anderen Knoten zuzugreifen. Dies wird bewerkstelligt, indem ein Kommandoblock, der Referenzen auf die SOFTWIRE-Liste enthält sowie detaillierte Informationen wie Knotenadresse, interne Adresse etc. spezifiziert, gesendet wird. Die SOFTWIRE-Liste wird auch für interne Variablen benutzt.

Virtuelles Token Passing

Jeder P-NET-Master erhält eine Knotenadresse (node address NA), die zwischen 1 und der erwarteten Gesamtanzahl der Master liegt.

Alle Master verfügen über einen „*idle bus bit period counter*“, der mit jeder Bitperiode, zu der der Bus inaktiv (idle) ist, inkrementiert wird. Sobald der Bus aktiv wird, wird er zu Null zurückgesetzt. Jeder Master verfügt weiterhin über einen „*access counter*“, der immer dann inkrementiert wird, wenn der *idle bus bit period counter* 40, 50, 60 ... erreicht.

Wenn der *access counter* in einem Master den Wert der eigenen Knotenadresse enthält, dann hält dieser Master den Token und kann auf den Bus zugreifen. Wenn der *access counter* die Maximalzahl der Master überschreitet, wird er auf 1 zurückgesetzt.

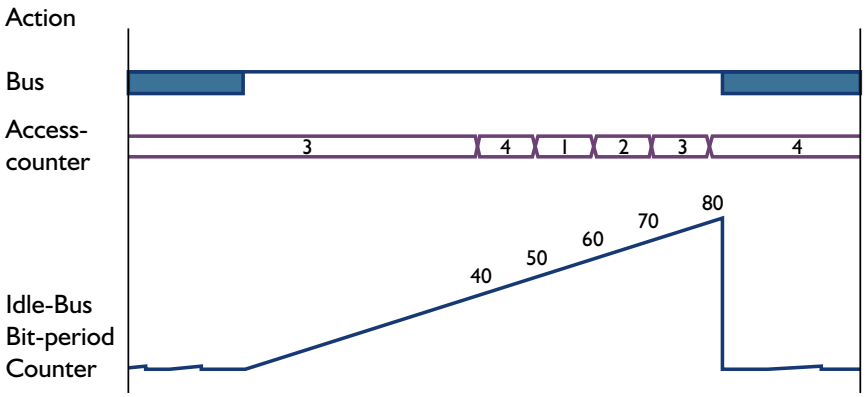


Abb. 12: Virtuelles Token-Passing

Das Diagramm in Abb. 12 zeigt ein Beispiel des Token Passing-Prinzips bei P-NET für ein System mit 4 Mastern.

Zuerst hat Master 3 den Token und empfängt eine Antwort von einem Slave. Daraufhin wird der Bus inaktiv (idle).

Sobald 40 idle bit periods gezählt wurden, werden alle access counters um eins erhöht und Master 4 kann nun auf den Bus zugreifen. Da Master 4 gerade nichts zu senden hat, kommt beim Stand von 50 bit periods wieder Master 1 an die Reihe.

Master 1 benötigt den Bus auch nicht (er könnte sogar ganz vom Bus abgeschaltet sein), woraufhin der virtuelle Token zu Master 2 transferiert wird, sobald der *idle bus bit period counter* 60 erreicht hat.

Da auch Master 2 und 3 keinen Zugriff durchführen wollen, gelangt der Token zu Master 4, wenn der *idle bus bit period counter* 80 enthält. Zu dieser Zeit kann Master 4 auf den Bus zugreifen. Daten erscheinen auf dem Bus, wodurch der *idle bus bit period counter* auf Null zurückgesetzt wird.

Die Weitergabe des virtuellen Tokens findet innerhalb von nur 130 s bzw. 10 Bit-Perioden statt. Es werden dazu keine Daten über den Bus gesendet. Ein einfaches Netzwerk kann bis zu 32 Master mit gleicher Priorität enthalten, wobei keine Hierarchie verwaltet werden muß.

Daraus folgt, daß P-NET keine zentralen Buszuteilungsfunktionen erfordert. Virtuelles Tokenpassing ist wesentlich effizienter als die Weitergabe des Tokens durch eine Nachricht.

P-NET im Vergleich mit dedizierten Feldbus-Chip-Lösungen

Bussysteme, die spezielle Chips nutzen (z.B. Profibus FMS), empfangen typischerweise zuerst den kompletten Datenrahmen. Daraufhin sendet der Chip eine Bestätigung an den Master und löst einen Interrupt in der Host-CPU aus.

Nun startet der Slave seine Bearbeitung und wenn alle Daten verfügbar sind, werden sie zum Chip transferriert. Jetzt muß der Master eine zweite Anfrage durchführen, um das gewünschte Resultat zu erhalten. Dies ist in Abb. 13 dargestellt.

P-NET-Slaves führen die Bearbeitung der Daten und den Empfang sowie die Aussendung von Datenrahmen parallel durch. Die Bearbeitung der Anfrage beginnt im Slave sofort, wenn die ersten Datenbytes ankommen. Auf diese Weise ist die bei P-NET standardmäßig verwendete Datenrate von 76.800 bit/s kein die Leistungsfähigkeit limitierender Faktor.

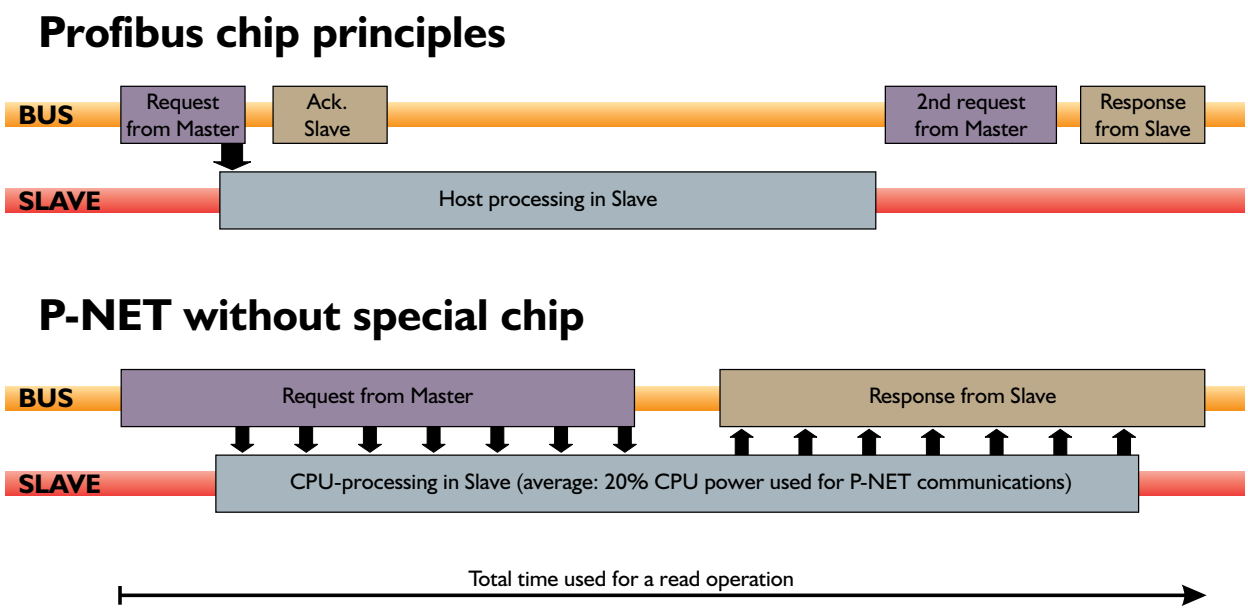


Abb. 13: P-NET-Kommunikation im Vergleich zum Chip-Prinzip bei Profibus

Jedes P-NET-Slave-Modul muß auf eine Anfrage innerhalb von 390 Mikrosekunden antworten („immediate response“). Dadurch besteht keine Notwendigkeit für Mehrfachanfragen auf eine einzelne Variable oder kontinuierliches Polling bis zum Erhalt des Ergebnisses. Die „immediate response“ benötigt keine Puffer im Slave, um eine Schlange von Anfragen oder Polling von verschiedenen Mastern zu verwalten.

Das Prinzip der „immediate response“ ergibt im Verbund mit der parallelen Abarbeitung sowie dem schnellen Tokenpassing eine Leistungsfähigkeit, die der von Bussystemen mit einer wesentlich höheren Datenrate (z.B. 500 kbit/s) entspricht.

Einer der Nachteile der Erhöhung einer Datenrate wäre nämlich, daß dies zu einer bedeutenden Reduzierung der erlaubten Kabellänge führen würde. So zum Beispiel kann die Kabellänge bei 76,8 kbit/s in der Gegend von 1,2 km liegen, während sie bei 500 kbit/s auf 200m reduziert werden müßte.

Ein vergleichbares komplett ausgebautes System würde deshalb 5 zusätzliche Repeater benötigen, wenn es mit der höheren Datenrate ausgeführt werden soll.

International P-NET User Organization

Nahezu 100 Firmen sind nun Mitglieder der International P-NET User Organization. Die Mitgliedsbeiträge (1995) für Firmen betragen 1.500 Dkr (ca. 400 DM bzw. £160).

Spezielle Konditionen gelten für Universitäten und andere Ausbildungseinrichtungen.

Mit der Aufnahme in die International P-NET User Organization erhält ein neues Mitglied eine Ausgabe des P-NET-Standards.

Mitglieder haben das Recht, das P-NET-Protokoll und das P-NET-Logo in Produkten zu benutzen, ohne Lizenzgebühren abführen zu müssen.

Die International P-NET User Organization organisiert internationale P-NET-Konferenzen, beteiligt sich an der internationalen Standardisierung und fördert das Wissen über P-NET.

Literaturlisten können ebenfalls von der International P-NET User Organization bezogen werden.

Wenn Sie zusätzliche Informationen wünschen, kontaktieren Sie bitte das Hauptquartier oder Ihre local society.

Hauptquartier:

- International P-NET User Organization,
  P.O. Box 192,
  DK-8600 Silkeborg
  Denmark
  Tel. +45 87 200 396, Fax +45 87 200 397
  e-mail: P-NET@post4.tele.dk

Local Societies:

- b+ Prof. Dr.-Ing. Jörg Böttcher, Deggendorf, Germany
  Tel. +49 991 340 897, Fax +49 991 340 447
  e-mail: 0991340897-0001@t-online.de

- PROCES-DATA (UK) Ltd., Oxon, England
  Tel. +44 (0) 1491 828 200, Fax +44 (0) 1491 828 201
  e-mail: pnet@easynet.co.uk

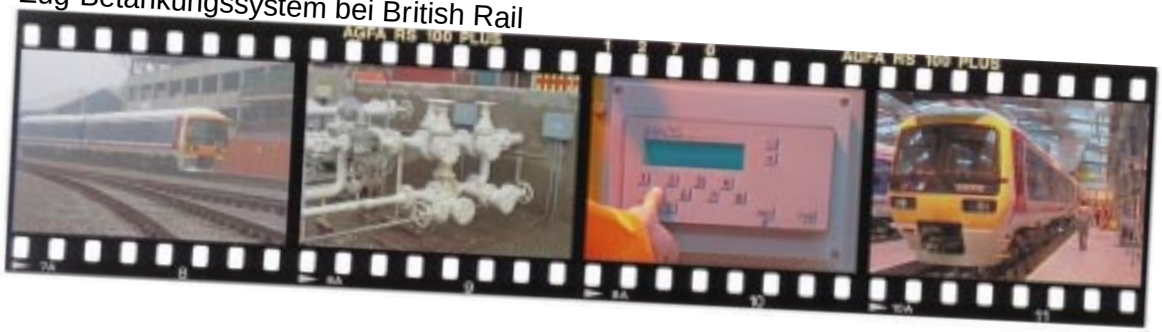
- TECNOCON, Vale de Cambra Cedex, Portugal
  Tel. +351 56 412 789, Fax +351 56 412 792

- CONFLOW Technologies Inc., Ontario, Canada
  Tel. +1 905 840 6800, Fax +1 905 840 6799

P-NET-Applikationen in der Praxis

Gebäudesicherung

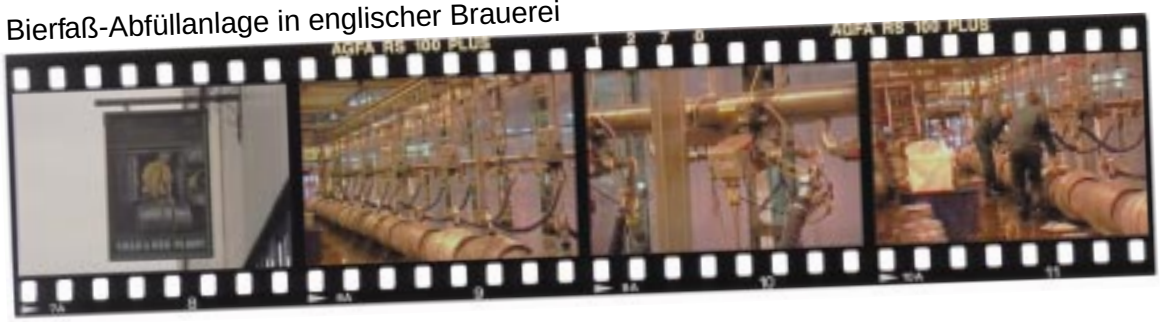
Zug-Betankungssystem bei British Rail



Vollautomatische Auto- & Passagier-Fähre in Holland

P-NET-Konferenzen und Ausstellungs-Teilnahmen

Bierfaß-Abfüllanlage in englischer Brauerei



Mil ch-Sammelwagen-Datenerfassungssystem

Komplette Anlage für die Betonherstellung

Automation einer Käseproduktionsanlage

Mengenmessung im eichpflichtigen Verkehr bei Öltankfahrzeugen