Tilo Klesper

LONWORKS®-Netzwerke

Eine Einführung in die Technologie

Echelon GmbH
Karl-Böhm-Str. 2
D-85598 Baldham b. München
Tel.: +49-8106-32024
Fax: +49-8106-32050
Web: www.echelon.com

Als Manuskript veröffentlicht

Inhaltsverzeichnis

1	Das LonTalk-Protokoll	2
2	Der Neuron Chip	7
3	LonWorks-Bausteine und -Transceiver	15
4	Programmierung des Neuron Chips	20
5	Entwicklungstools	29
6	Planung, Installation und Betrieb von LonWorks-Netzwerken	34
n/a	Glossar, Literaturhinweise	41

Kap.1: Das LonTalk-Protokoll

Grundlage jeder Datenkommunikation ist ein geeignetes Protokoll, damit die richtigen Gesprächspartner adressiert werden und sich diese gegenseitig auch verstehen. Wie am Beispiel der menschlichen Kommunikation ersichtlich, sind hierbei viele Voraussetzungen zu erfüllen, wie gleiche Sprache, Syntax, Semantik, normative und situative Kontextumstände etc. Die Implementierung von Adressierungsverfahren, Medienzugriff, Fehlersicherung usw. haben direkten Einfluss auf Verhalten und Leistungsfähigkeit des Datenaustauschs. Um sicher Daten aus einer Applikation heraus in eine oder mehrere andere Applikationen zu übertragen, ohne dass die Applikation die zugrundeliegenden Mechanismen und Hardwarearchitekturen behandeln muss, wurde von der ISO das Open Systems Interconnection (OSI) Reference Model entworfen. Hier werden Dienste, Mechanismen und Schnittstellen definiert und jeweils einer aus sieben festgelegten Schichten zugeordnet.

Dem Vorbild der Computer-Netzwerke folgend, implementiert das LonTalk-Protokoll entsprechende Dienste auf allen sieben Schichten, es wurde jedoch für die speziellen Anforderungen an Netzwerke in der Steuerungstechnik, den sog. Kontrollnetzwerken, optimiert. Im Vordergrund standen dabei hoher Nachrichtendurchsatz, Medienunabhängigkeit, freie Topologie und kurze Reaktionszeiten. Mit einer Vielzahl von Parametern werden die einzelnen Protokolldienste auf die spezifischen Systemeigenschaften eines Netzwerkes wie Übertragungsmedien, Nachrichtenaufkommen, Sicherheit, etc. angepasst. In der Regel wird dies automatisch von den Tools übernommen, ohne dass sich Programmierer und Betreiber des Netzwerkes darum kümmern müssen. Die Datenlänge eines Paketes ist variabel, der Neuron Chip erlaubt Paketlängen bis 255 Bytes.

Das LonTalk-Protokoll unterstützt auf Schicht 1 *(Physical Layer)* verschiedene Übertragungsmedien wie Twisted Pair, Powerline, Funk, Infrarot, Koaxkabel, Lichtwellenleiter. Die elektrische Datenmodulation kann dabei entweder vom Neuron-Chip oder vom jeweiligen Transceiver selbst übernommen werden (siehe Kap. 1.2.5). Ein LonWorks-Netzwerk kann mit verschiedenen Medien gleichzeitig realisiert werden, man spricht hier von Kanälen, an welche die einzelnen Knoten angeschlossen werden. Nachrichten werden von intelligenten Routern und Bridges von einem Kanal zum anderen weitergeleitet, ihre physikalischen Parameter dabei automatisch umgesetzt. Hierum kümmern sich die Dienste der OSI-Schicht 3. Je nach Konfigurierung eines Routers als lernender oder konfigurierter Router, Bridge oder Repeater kann ein Netzwerk auch logisch segmentiert werden. Gesamtkapazität und Zuverlässigkeit insbesondere großer Systeme lassen sich hierdurch sehr einfach erhöhen. Die Konfiguration der Router kann ebenfalls automatisch bei der Installation des Netzwerkes erfolgen.

Bitrate (kBit/s)	Pakete/s (max.)	Pakete/s (Dauerlast)
4,883	25	20
9,766	45	35
19,531	110	85
39,063	225	180
78,125	400	320
156,25	625	500
312,5	800	650
625,0	950	760
1.250,0	1000	800

Tabelle 1.1 Kanalkapazität (12-Byte-Pakete)

Bitrate (kBit/s)	Pakete/s (max.)	Pakete/s (Dauerlast)
4,883	7	5
9,766	13	10
19,531	25	20
39,063	50	40
78,125	100	80
156,25	200	160
312,5	355	285
625,0	545	435
1.250,0	750	600

Tabelle 1.2 Kanalkapazität (64-Byte-Pakete)

Die Hauptaufgabe der Schicht 2 *(Link Layer)* ist neben der Rahmenbildung und CRC-Fehlerprüfung der Medienzugriff (Media Access Control - MAC). LonTalk verwendet ein prädiktives *p*-persistentes CSMA-Verfahren mit optionaler Kollisionserkennung (/CD) sowie Kollisionsvermeidung (/CA). Dieses mehrfach patentierte Verfahren vermeidet die Nachteile bisher eingesetzter Verfahren wie TDM, Token Ring, Token Bus und CSMA wie z.B. IEEE 802.3 (Ethernet) und ermöglicht so einen kontinuierlichen Datenaustausch auch bei Mehrfach-Medien-Kommunikation, geringen Datenraten (Funk, Powerline), hohem Nachrichtenaufkommen und großen Netzwerken. Grundsätzlich wird der Zugriffzeitpunkt auf das Medium stochastisch über eine bestimmte Anzahl von Zeit-Slots zwischen den Knoten verteilt. Beim vorausschauenden *p*-persistenten Medienzugriff wird die Anzahl der Slots dynamisch von jedem Knoten für jeden einzelnen Zugriff berechnet und so unabhängig von der Anzahl der kommunizierenden Knoten die Kollisionswahrscheinlichkeit konstant niedrig gehalten. Da die Kollisionserkennung bei offenen Medien wie Funk nicht möglich ist, kann sie bei drahtgebundenen Medien optional eingeschaltet werden.

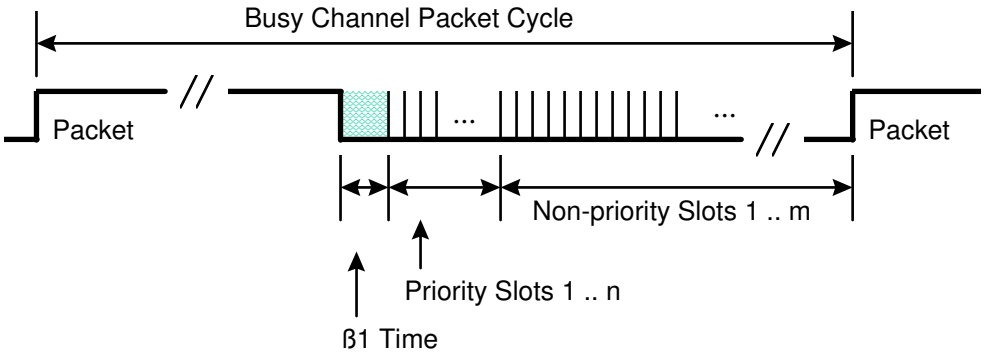

Bild 1.1: LonTalk-Medienzugriff

Zusätzlich kann einem Knoten optional eine bestimmte Priorität zugeordnet werden, so dass wichtige Nachrichten garantiert einen fest definierten, bevorzugten Slot für den Zugriff erhalten. Zusammen mit den optionalen Prioritäten für Nachrichten und den Task-Scheduler entsteht so ein System, das wichtigen Daten eine garantierte Reaktionszeit zuteilt und dennoch keinen Teilnehmer vom Zugriff ausschließt. Damit nimmt der MAC-Algorithmus eine herausragende Sonderstellung sowohl im Bereich der Computernetzwerke als auch der Feld-, Device- und Sensor-/Aktorbusse ein.

Schicht 3 *(Network Layer)* ist neben dem Routing für die Adresserkennung zuständig. Bei LonWorks-Netzwerken werden verschiedene Adressierungsverfahren eingesetzt. So besitzt jeder Neuron Chip zunächst einmal eine eindeutige 48 Bit lange Seriennummer, die sog. Neuron ID. Über diese kann jeder Neuron Chip gezielt adressiert werden, beispielsweise solange er noch keine logische Netzwerkadresse besitzt. Eine solche Adresse wird ihm bei der Konfiguration des Netzwerkes zugeteilt. Ähnlich wie ein Brief von der Post durch die Angabe von Ort, Straße und Hausnummer zugestellt wird, wird auch ein Netzwerk in Domains, Subnets und Nodes eingeteilt. Zusätzlich können innerhalb einer Domain Knoten zu logischen Gruppen eingeteilt werden, die einander überlappen und auch verschiedene Subnets umfassen dürfen. Auf diese Weise kann eine einzelne Nachricht an alle Knoten einer Domain oder eines Subnets geschickt werden (Broadcast), es kann eine einzelne Nachricht an alle Mitglieder einer bestimmten Gruppe gesendet werden (Multicast) oder es kann ein Knoten gezielt durch Angabe seiner Subnet-/Node-Adresse angesprochen werden (Unicast). Gerade die Gruppenadressierung ist sehr effizient: nur eine Teilmenge der Knoten wird angesprochen, mit einer einzigen Nachricht, deren Zieladresse aus nur einem Byte besteht.

Ein Netzwerk kann insgesamt aus bis zu 2**48 Domains bestehen, die jeweils 127 * 255 = 32385 Knoten umfassen können.

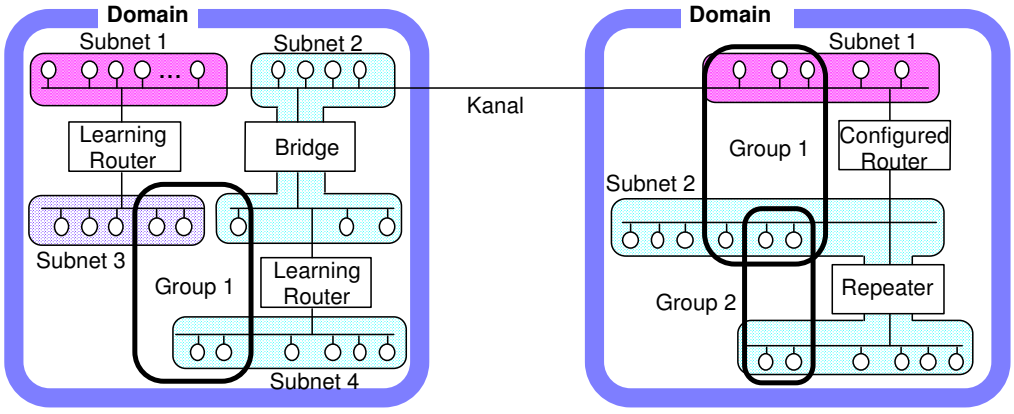

Bild 1.2: Segmentierung eines Netzwerkes

Bestätigungen von Nachrichten werden von der Schicht 4 *(Transport Layer)* abgewickelt. Hier wird Buch geführt über jede einzelne Transaktion, um so deren richtige Reihenfolge sicherzustellen und um Duplikate zu erkennen, was bei freier Topologie mit Ringen und Schleifen unerlässlich ist. Bestätigungen sind somit, im Gegensatz zu vielen anderen Systemen, tatsächlich end-to-end und eindeutig. Um unberechtigte Zugriffe zu unterbinden, ist ein Authentifizierungsmechanismus eingebaut, der gleiche Schlüssel bei Sender und Empfänger voraussetzt. Um schnelle Übertragungen mit dem "Unacknowledged Repeated Service" zu ermöglichen, sind konfigurierbare Timer und Counter für Anzahl der Retries und Reminder, Abstand der Wiederholungen und Time-Outs der Transaktionen eingebaut.

Auf Schicht 5 *(Session Layer)* wird ein Request-Response-Protokoll zur Verfügung gestellt, das für Remote Procedure Calls verwendet wird. Neben Applikationsfunktionen werden auch die Netzwerkmanagement-Funktionen über dieses Protokoll ausgeführt. Hier steht ebenfalls der Authentifizierungsmechanismus zur Verfügung.

Die Schicht 6 *(Presentation Layer)* stellt, in Zusammenarbeit mit dem *Application Layer* auf Schicht 7, die Funktionen zur Behandlung der zu sendenden oder empfangenen Daten zur Verfügung. Daten können aus der Applikation heraus implizit als Netzwerkvariablen oder als explizite Messages (mit allen Adressierungsarten, implizit oder explizit) abgesetzt werden. Beim Empfang werden automatisch die entsprechenden Events generiert und die Daten passend im Speicher abgelegt. Eine Netzwerkvariable kann als eine Art netzwerkweit gültige gloabale Variable betrachtet werden. Mit der passenden logischen Konfiguration, dem sog. "Binding", werden Nachrichten automatisch gesendet bzw. bei Empfang der richtigen Task übergeben. Netzwerk-Management- und -Diagnosenachrichten werden ebenfalls automatisch bearbeitet. Weiterhin ist eine Fremdrahmenübertragung vorgesehen, mit der beispielsweise Nachrichten von fremden Protokollen "getunnelt" werden können. Dieser Mechanismus ist auch hilfreich bei der Erstellung von Gateways.

Auf Schicht 7 wird weiterhin eine Schnittstelle für die interoperable Dateninterpretation mit Hilfe von Standard-Typen für Netzwerkvariablen ("SNVT" - Standard Network Variable Types) und Parametrierungsdaten angeboten. Es entfällt damit die Notwendigkeit, eine korrelierende Schnittstelle in der Applikation zu codieren.

Trotz der Implementation eines vollständigen Protokoll-Stacks bleibt der Overhead pro Nachricht relativ klein. Jeder Layer addiert lediglich ein Byte (z.B. Gruppenadressierung in Schicht 3), beim Layer 6 sind es zwei, hinzu kommen zwei Bytes für die CRC-Prüfsumme. Durch die Verteilung der einzelnen Funktionen auf die 3 Prozessoren des Neuron Chips wird der Protokollstack sehr effizient abgearbeitet, mit der 10-MHz-Version des Neuron

Chips sind so Applikation-zu-Applikation-Reaktionszeiten von 7 ms möglich, bei der 20-MHz-Version werden diese Zeiten annähernd halbiert. Inwieweit eine sequentielle Implementation auf anderen Prozessoren Einfluss auf diese Zeiten haben wird, bleibt abzuwarten. Motorola arbeitet zur Zeit der Drucklegung dieses Buches beispielsweise an einer 32-Bit-Implementation auf dem MC 68360. Generell sind seit der freien Verfügbarkeit des LonTalk-Protokolls alle Plattformen für eine eigene Implementation denkbar. Der Neuron Chip lässt sich jedoch auch als reiner Kommunikations-Coprozessor an beliebigen Plattformen, vom Mikrocontroller bis zum Großrechner, einsetzen. Echelon stellt die notwendige Soft- und Firmware in Form des MIP (Mikroprocessor Interface Program) zur Verfügung. Schicht 6 und 7 des Protokolls werden dann auf dem Hostprozessor abgearbeitet. Damit ist die Erweiterung beliebiger Applikationen zu LonWorks-Netzwerken mit minimalem Aufwand möglich.

OSI Layer		**Purpose**	**Services provided**	**Processor**
7	Application	Application Compatibility	Standard Network Variable Types	Application
6	Presentation	Data Interpretation	Network Variables, Foreign Frame Transmission	Network
5	Session	Remote Actions	Request-Response; Authentication; Network Management; Network Interface	Network
4	Transport	End-to-End Reliability	Acknowledged & Unacknowledged; Unicast & Multicast, Authentication; Common Ordering; Duplicate Detection	Network
3	Network	Destination Adressing	Addressing, Broadcast, Routers	Network
2	Link	Media Access and Framing	Framing; Data Encoding; CRC Error Checking; Predictive CSMA; Collision Avoidance; Optional Priority & Collision Detection	MAC
1	Physical	Electrical Interconnect	Media-Specific Interfaces and Modulation Schemes (twisted pair, power line, radio frequency, coaxial cable, infrared, fiber optic)	MAC, Transceiver

Tab 1.3: Das LonTalk-Protokoll in der ISO/OSI-Darstellung

Kap. 2: Der Neuron Chip

Intelligent und kommunikativ

Der Neuron Chip ist eine einzigartige Kombination von Hardware und Firmware zum einfachen und kostengünstigen Aufbau von Netzwerkapplikationen, basierend auf dem Konzept der verteilten Intelligenz. Als Kern eines intelligenten Netzwerkknotens kann er sowohl die eigentliche Applikation abarbeiten als auch die Schnittstelle zum Netzwerk betreiben. Weiterhin stellt er I/O-Funktionen zum Betreiben der physikalischen Sensoren und Aktoren zur Verfügung,

Die Hardware-Architektur, direkt auf das LonTalk-Protokoll zugeschnitten, besteht grundsätzlich aus einem symmetrischen Multiprozessor-System, das drei 8-Bit-Prozessoren, RAM, EEPROM und ROM (Neuron 3120) umfasst. Die drei Prozessoren sind über einen gemeinsamen 16-Bit-Adreßbus und einen 8-Bit-Datenbus verbunden. So teilen sie sich gemeinsame Ressourcen wie Speicher und ALU. Weitere Funktionseinheiten des Neuron Chip sind vor allem ein dedizierter Kommunikationsport, eine I/O-Logikeinheit, zwei Timer/Counter sowie zusätzliche Steuerlogik. Die Firmware des Neuron Chip enthält den Microcode für die drei Prozessoren, das LonTalk-Protokoll, einen Task-Scheduler für den Applikationsprozessor und eine I/O-Library. Damit stellt die Firmware ein vollständiges, eventgesteuertes Betriebssystem zur Verfügung.

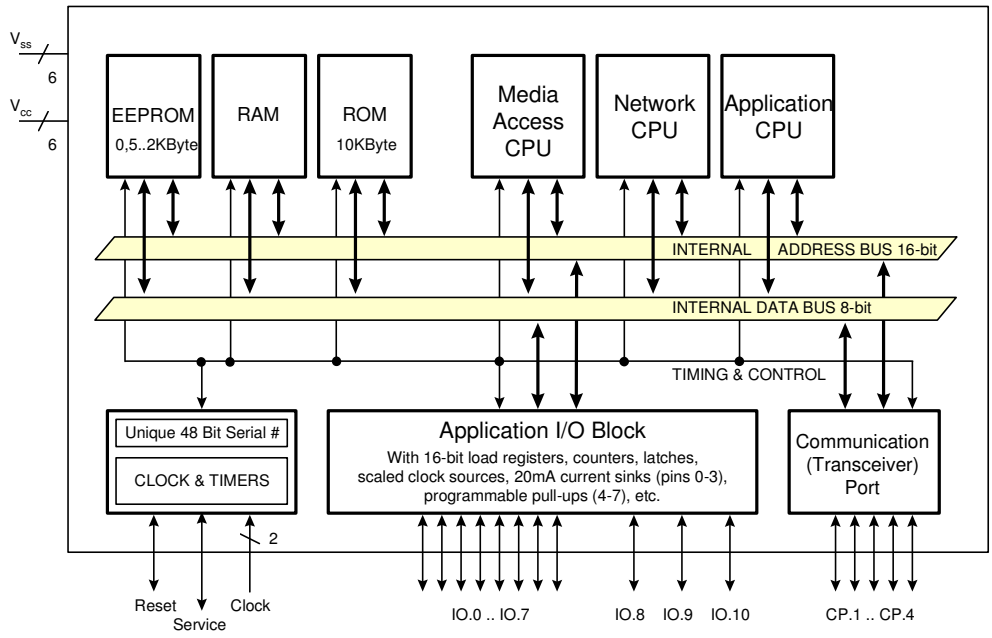

Bild 2.1: Neuron 3120

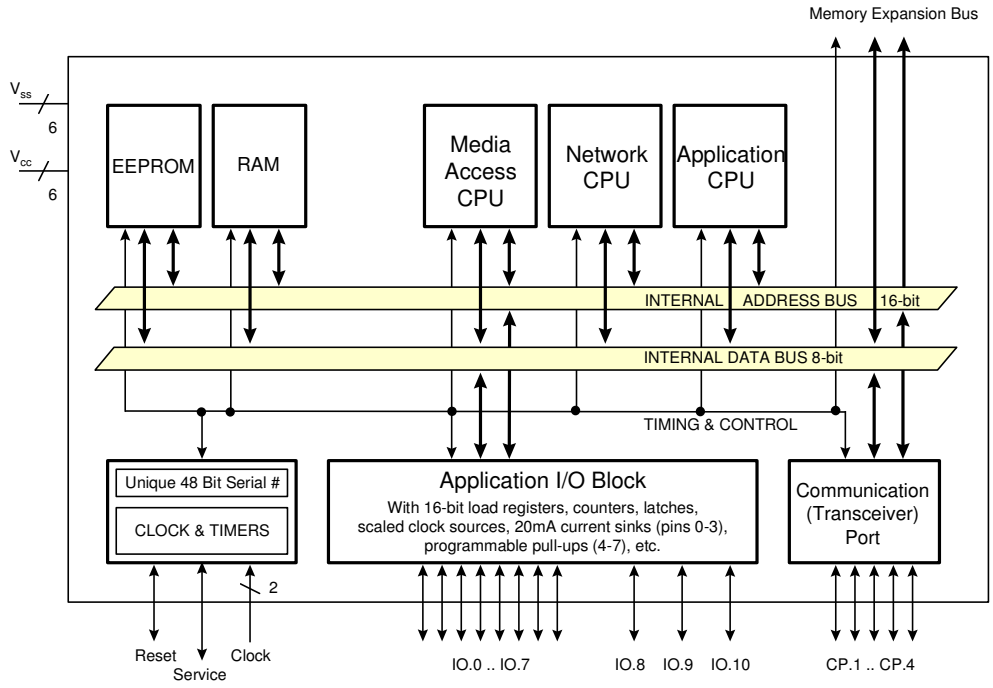

Bild 2.2: Neuron 3150

Hersteller	Bezeichnung	EEPROM	RAM	ROM
Motorola	MC143120B1DW	0.5K	1K	10K
	MC143150B1FU	0.5K	2K	off-chip
	MC143120E2	2K	2K	10K
Toshiba	TMPN3120B1F	0.5K	1K	10K
	TMPN3150B1F	0.5K	2K	off-chip
	TMPN3120E1	1K	1K	10K

Typ. Stromaufnahme	18mA @ 10MHz
Sleep-Mode-Strom	15µA
Temperaturbereich	-40ºC to +85ºC

Bild 2.3: Neuron-Derivate

Dreifache Intelligenz

Die drei stackorientierten Prozessoren enthalten jeweils einen eigenen Registersatz, teilen sich aber Adress- und Daten-ALUs sowie die Speicherzugriffslogik. Auf diese Weise können sie in einer Pipeline quasi-parallel betrieben werden, was eine hohe Performance ohne zusätzlichen Hardwareaufwand ermöglicht. Auf zeitraubende Interrupts und Ladevorgänge kann somit verzichtet werden. Die Aufgaben der drei Prozessoren werden durch die Firmware wie im folgenden beschrieben festgelegt:

CPU 1 ***"MAC CPU"*** ist der Medienzugriffsprozessor, der die Schichten eins und zwei des Protokollstacks abarbeitet. Er bedient den Kommunikationsport und wickelt neben Telegrammrahmung und -sicherung die Kollisionsvermeidung ab. Die Kommunikation mit der CPU 2 erfolgt über Netzwerkbuffer im gemeinsamen Speicher.

CPU 2 ***"Network CPU"*** bearbeitet die Aufgaben der Protokollschichten drei bis sechs, wozu vor allem Netzwerkvariablenbearbeitung, Adressierung, Transaktionsmanagement, Authentifizierung, Netzwerkmanagement und Routing gehören. Darüber hinaus ist sie für Hintergrunddiagnose, Software-Timer und die Steuerung der I/O-Logik zuständig. Zur Kommunikation mit CPU 1 werden Netzwerkbuffer, zur Kommunikation mit CPU 3 Applikationsbuffer benutzt. Der Zugriff auf die frei konfigurierbaren Buffer wird über Hardwaresemaphoren gesteuert.

CPU 3 ***"Application CPU"*** arbeitet das eigentliche Applikationsprogramm und die Aufrufe der Betriebssystem-Funktionen ab. Programmiert wird sie in der Regel in Neuron C, einer Erweiterung von ANSI C, zugeschnitten auf die speziellen Erfordernisse von Eventverarbeitung, Kommunikations-, Timer- und I/O-Objekten. Der Programmierer braucht sich also um die Implementation dieser Funktionen nicht mehr zu kümmern und kann sich ganz auf den Anwendungs-Algorithmus konzentrieren.

Derzeit wird der Neuron Chip von Motorola und Toshiba in zwei Derivaten mit mehreren Versionen angeboten: Neuron 3120 und 3150. Sie unterscheiden sich (außer ihrem Gehäuse) nur durch den Aufbau ihrer Speicherstrukturen. Der 3120 stellt die Firmware in einem on-chip-ROM zur Verfügung und ist somit eine echte single-chip-Lösung, während der 3150 externen Speicher betreibt und damit auch größere Applikationen bearbeiten kann. Beide Versionen benutzen Teile des RAM für Stack, Buffer und Applikationsvariablen. Daten und Funktionen können aus der Applikation heraus gezielt im RAM untergebracht werden. Im internen EEPROM werden sämtliche Konfigurationsparameter (Adresstabellen, Bindings, Konfigurationsvariablen, usw.) abgelegt. So sind sie zum einen nichtflüchtig gespeichert, können aber jederzeit aus der Applikation heraus oder über das Netzwerk geändert werden.

Beim 3120 wird im EEPROM ebenfalls das Anwendungsprogramm gespeichert, was für kleinere Applikationen ausreicht, da aufwendige Funktionen zur Kommunikation und I/O von den Firmware-Routinen im ROM übernommen werden.

Beim 3120 befindet sich der größte Teil der Firmware in einem 10 KByte großen ROM on-chip, beim 3150 wird das Betriebssystem zusammen mit der Applikation im externen Speicher untergebracht. Adress- und Datenbus sowie die Steuerleitungen sind beim 3150 nach außen geführt und können zum Anschluss von Speicherbausteinen ebenso genutzt werden wie für Memory-mapped-I/O, beispielsweise über Dual-Port-RAM oder einen UART. Es können beliebige Speichertypen verwendet werden - PROM, RAM, EEPROM oder Flash-Memory, wobei 16 KByte ROM oder Flash für das Betriebssystem reserviert sind. Die am häufigsten verwendete Konfiguration ist wohl der 3150 mit 32 KByte EPROM, was eine Applikation von 16 KByte zulässt. Insgesamt stehen für den Applikationscode 42 KByte zur Verfügung, nach Abzug der 16 KByte Systemspeicher und 6 KByte internem Speicher. Die adressierbaren 64 KByte können leicht mit Bausteinen wie der PSD39xx-Serie von WSI erweitert werden, ebenso die Anzahl der I/O-Leitungen. Die einzelnen Speicherbereiche werden von der Firmware sowohl beim Bootvorgang als auch während des Betriebes kontinuierlich auf Datenkonsistenz geprüft.

Ein-/Ausgabe

Der Neuron Chip enthält zwei verschiedene Schnittstellen zur Außenwelt - einerseits zum Netzwerk hin, andererseits zum Prozess, d.h. zur I/O-Seite hin. Hier kann an 11 Pins die externe I/O-Hardware angeschlossen werden. Im Gegensatz zu den I/O-Ports herkömmlicher Mikrocontroller stellt der sog. I/O-Block des Neuron Chip zusätzliche Hardwareschaltungen und Register zur Verfügung, um in Kombination mit speziellen Firmware-Funktionen, den sog. I/O-Modellen, den Anschluss vieler Sensor- und Aktortypen besonders einfach zu gestalten. Als wesentliche Hardwareelemente sind hier zwei 16-Bit-Timer/Counter zu nennen. In der Firmwareversion 7 stehen insgesamt 34 I/O-Modelle zur Verfügung, die sich bei der Programmierung als Objekte darstellen. Durch die Deklaration dieser I/O-Objekte werden die entsprechenden I/O-Pins für das jeweilige Modell konfiguriert, so dass sich die Programmierung auf den Aufruf der Funktionen io_in und io_out reduziert. Im Kapitel zur Programmierung wird ein Beispiel für eine Infrarotfernsteuerung vorgestellt. Die Modelle werden in folgende Klassen eingeteilt:

Direkte I/O-Modelle für den Austausch von digitalen Bit-, Nibble- oder Byte-Daten sowie die Nutzung eines Pins als Triggereingang.

Serielle I/O-Modelle zum bitseriellen Datentransfer in verschiedenen Formaten: Bitshift (Schieberegister), I²C, Magnetkartenleser, NeuroWire

(Motorola SPI bzw. National Microwire), Seriell (z.B. EIA-232), Touch I/O und Wiegand Input (Wiegand-Kartenleser).

Parallele I/O-Modelle, entweder als Muxbus zum Aufbau eines parallelen bidirektionalen 8-Bit-Ports mit Multiplex-Adressierung oder zur parallelen Ankopplung eines zweiten Neuron Chips oder beliebiger Mikroprozessoren. Dieses Modell mit Fluss- und Richtungssteuerung wird beispielsweise bei dem Microprocessor Interface Program (MIP) verwendet, mit dem der Neuron Chip als Koprozessor an einem beliebigen Host-Prozessor betrieben wird.

Timer/Counter Input-Modelle zur Erfassung von zeit- und impulsbezogenen Eingangsgrößen. Dies sind im einzelnen: Ontime (Dauer eines Einzelimpulses), Period (Periodendauer einer Impulsfolge), Pulsecount (Anzahl Impulse/Zeiteinheit), Totalcount (Gesamtzahl von Impulsen), Quadratur (Auf-/Abwärtszählen, z. B. Inkrementalgeber), Infrared (RC5-Infrarot-Demodulator), Edgelog (zeitverteilte Impulsfolge, z. B. Barcodeleser) und Dualslope (A/D-Wandlung).

Timer/Counter Output-Modelle zur Ansteuerung verschiedener Aktortypen mit zeit- und impulsbezogenen Ausgabewerten. Dies sind im einzelnen: Oneshot (Einzelimpuls definierter Länge), Pulsecount (eine bestimmte Anzahl von Impulsen), Pulsewidth (Pulsweitenmodulierter Impulsstrom), Frequency (Rechteckfolge mit variabler Frequenz), Edgedivide (programmierbarer Frequenzteiler), Triac (Phasenanschnittsteuerung mit Triac) und Triggeredcount (getriggerte Anzahl von Impulsen, z. B. für Schrittmotor).

Die kleinste Zeiteinheit für die Timer/Counter-Modelle ist durch den Systemtakt des Neuron Chip festgelegt und beträgt 200 ns bei 10MHz. Die Auflösung ist programmierbar und auch während der Programmlaufzeit änderbar, dies ist wichtig bei der Messung unbekannter Eingangsgrößen. Durch die Implementation der Hardware-Busse SPI/Microwire und I²C steht dem Entwickler eine große Zahl von Peripheriebausteinen zum direkten Anschluss an den Neuron Chip zur Verfügung. Da die einzelnen I/O-Modelle gezielt auf den jeweils konfigurierten I/O-Pins arbeiten, können verschiedene I/O-Objekte gleichzeitig am I/O-Block betrieben werden, lediglich das parallele Modell belegt die Schnittstelle exklusiv.

Kommunikation

Die Schnittstelle zum Netzwerk ist der Communication Port, der ebenfalls über eine eigene Logikeinheit verfügt. Um dedizierte Transceiver für die unterschiedlichen Medien zu unterstützen, können die fünf Pins des Kommunikationsports in drei verschiedenen Modi betrieben werden:

Im ***Differential Direct Mode*** können die eingebauten Twisted-Pair-Transceiver der Neuron Chips direkt (oder über passive Komponenten) miteinander gekoppelt werden. Sie führen eigenständig das Manchester-En-/Decoding durch, wodurch einerseits das Hardwaredesign vereinfacht wird, andererseits die Schnittstelle polaritätsinvariant wird. Um einen höheren Common Mode Range zu erreichen und vor allem eine galvanische Entkopplung zu ermöglichen, werden in der Regel transformator-entkoppelte, passive Transceiver angeschlossen. Echelons Twisted-Pair-Transceiver für 78 kBit/s und 1,25 MBit/s ermöglichen hierbei einen Common Mode Range von ±277 V RMS. Vier Pins werden für Transmit/Receive verwendet, der fünfte als Eingang für die Kollisionserkennung geschaltet.

Der ***Single-Ended Mode*** wird für den Anschluss von externen, aktiven Transceivern verwendet. Solche sind notwendig für Medien wie Funk, Infrarot, Lichtwellenleiter und Koaxkabel; aber auch Transceiver für EIA-485 und vor allem Echelons Free-Topology-Transceiver werden in diesem Modus betrieben. Es wird ebenfalls die Manchester-Codierung verwendet, neben den beiden Pins für Transmit/Receive gibt es einen Eingang für die Kollisionserkennung, einen Transmit-Enable-Ausgang und einen Ausgang für das Sleep-Mode-Signal. Spezielle Features wie Signalgenerierung mit abgerundeten Flanken (EMV) und Echoerkennung (freie Topologie) können so direkt vom DSP des FTT-10A und des LPT-10 vorgenommen werden.

Der ***Special Purpose Mode*** sendet und empfängt Daten in einem nicht codierten Format und überlässt die Modulation und die Verarbeitung der Präambel einem intelligenten Transceiver. Die Daten werden getaktet simultan gesendet und empfangen, ein Datenwort jeweils aus einem Status- und Datenbyte bestehend. Der Pin CP3 wird bidirektional für den Sleep Mode betrieben. Ein Beispiel für intelligente Transceiver sind Echelons Powerline-Transceiver, die ihre eigene Breitband- oder Schmalband-Modulation vornehmen. Auch Fehlerkorrektur und Kollisionsbehandlung werden dabei eigenständig durchgeführt.

In allen drei Modi sind die Übertragungsrate und die Parameter für die verschiedenen Transceiver unabhängig vom Applikationsprogramm frei einstellbar. Diese Parameter werden vom Entwicklungs- oder Installationstool automatisch konfiguriert, können bei Bedarf jedoch (auch während des Netzwerkbetriebs) jederzeit geändert und bestimmten Betriebsbedingungen angepasst werden.

Neben diesen Kernelementen des Neuron Chips sind weitere Funktionseinheiten integriert, die zum Aufbau eines Netzwerkes wesentlich sind und das Knotendesign vereinfachen. Zum einen ist jedem Neuron Chip eine 48 Bit lange Seriennummer, die sog. Neuron ID, fest einprogrammiert. Jeder Knoten im Netzwerk kann damit unabhängig von seiner logischen Konfiguration erkannt und angesprochen werden. Zum anderen dient der

bidirektionale Service Pin dazu, das Senden dieser Neuron ID sowie einer Programmkennung als Broadcast-Nachricht auszulösen (was ebenfalls durch Aufruf einer Systemfunktion möglich ist). Netzwerkmanagement-Tools zur Installation und Diagnose können dieses Telegramm auswerten, wichtig bei Plug-and-Play-Systemen und zum transparenten Ersetzen eines Knotens im laufenden Betrieb. Der Service Pin treibt auch direkt eine LED, um über den aktuellen Knotenstatus mit verschiedenen Kennungen zu informieren. Ein Watchdog-Timer überwacht die korrekte Funktion des Chips und löst im Fehlerfall einen Reset aus. Ursache eines Resets (Watchdog, Power-up, Reset-Pin, Software etc.), ein Log für fatale Fehlersituationen und Zähler für Kommunikationsprobleme (Timeouts, Überläufe, Checksummenfehler etc.) werden in eigenen Registern festgehalten. So können Fehlerursachen sowohl über das Netzwerk als auch aus der Applikation heraus ausgewertet werden.

Der Neuron Chip führt während des Betriebes Checksummen über die verschiedenen Speicherbereiche, wobei er Konfiguration, Applikation und Betriebssystem getrennt erfasst. Für den Fehlerfall können verschiedene Aktionen vorgesehen werden, wie in Tabelle 2.1 zusammengefasst.

Recovery Word	Description
0x0001	Reboot application if fatal error
0x0002	Always reboot application on reset (see Note 1)
0x0004	Reboot configuration if configuration checksum fails
0x0008	REboot configuration on an application fatal error
0x0010	Always reboot configuration on reset
0x0020	Reboot communication parameters if configuration checksum fails
0x0040	Reboot communication parameters if type or rate mismatch
0x0080	Always reboot communication parameters on reset
0x0100	Reboot EEPROM variables when rebooting application
0x0200	Go applicationless if an application fatal error occurs (prohibits network loading, used in conjunction with 0x0001). Application fatal errors are defined below. (See Note 1)
0x0400	Checksum all code, including system image.

Note 1: Application exported with these options cannot be loaded over the network.

Tab. 2.1: Recovery Action Bit Masks

Mit diesen Reboot-Möglichkeiten, zusammen mit den Events für Kommunikations- und Applikationsfehler, können fehlerbehandelnde Maßnahmen auch bei sicherheitskritischen Anwendungen sehr einfach und vielfältig programmiert werden.

Für den Aufbau stromsparender Geräte (z. B. für Batteriebetrieb) steht der Sleep Mode zur Verfügung. Er kann aus der Applikation heraus initiiert werden und entweder durch Betätigung eines dedizierten Wake-Up-Pins oder durch Empfangen einer Nachricht beendet werden. Die typische Stromaufnahme

reduziert sich hierbei auf 15 µA (3120B1) bis 4 µA (MC143120LE2DW). Bei den Neuron Chips der dritten Generation sind on-chip LVD- und LVI-Schaltkreise enthalten (Low Voltage Detection / Low Voltage Inhibit). Damit beschränkt sich der Hardwareaufwand für Intelligenz und Kommunikationsfähigkeit eines Knotens im wesentlichen auf Neuron Chip, Speicher (nur 3150), Transceiver, Quarz, 2 Taster, 2 LEDs und die Spannungsversorgung. Selbst auf diese kann verzichtet werden, wenn der Knoten über den LPT-10 fremdgespeist wird.

Der Neuron Chip kann auch als Kommunikations-Koprozessor an einem beliebigen Host-Prozessor betrieben werden. Dies kann notwendig sein, wenn beispielsweise höchste Rechenleistung oder ein bestimmtes Betriebssystem gefordert wird. Aber auch, um eine bereits bestehende Applikation netzwerkfähig zu gestalten, ist dies eine einfache und gleichzeitig effiziente Lösung. Für diesen Zweck steht mit dem Microprocessor Interface Program (MIP) eine spezielle Firmware zur Verfügung, zusammen mit Treibern und Libraries für die Entwicklung der Host-Software. Echelons Rechner-Interfaces beispielsweise bauen auf dem Neuron 3150 und dieser Software auf.

Da die beschriebenen Funktionen entweder bereits als Hardware-Logik oder als Firmware-Funktion vorliegen, wird es sehr einfach, diese zu programmieren und zu benutzen. Die Programmiersprache Neuron C enthält entsprechende Erweiterungen der Sprachdefinition und wird direkt von den Entwicklungssystemen unterstützt. Zusammengenommen wird hierdurch der Aufwand für Aufbau und Programmierung eines Knotens für viele Aufgaben gegenüber herkömmlichen Ansätzen drastisch reduziert. Dies ist der Grund für manchen Entwicklungsingenieur, den Neuron Chip auch dann als Mikrocontroller einzusetzen, wenn an die Vernetzung eines Produktes vorerst gar nicht gedacht wird. Die Erweiterung zu einem LON-fähigen Produkt beschränkt sich dann auf die Nachrüstung der Transceiver-Baugruppe. Ebenso ist die Möglichkeit, ein Software-Update oder neue Programmparameter über das Netzwerk zu laden, ein Grund für den Einsatz des Neuron Chip auch für Stand-Alone-Applikationen.

Kap. 3: LONWORKS-Bausteine und -Transceiver

Die Hardware-Bausteine

Wichtigstes Element eines Netzwerkknotens ist neben dem Neuron-Chip (oder Prozessor) der Transceiver als Bindeglied zwischen Kommunikationsport und Medium. Er ist für die physikalische Einkopplung des Signals auf das jeweilige Medium und beim Empfang für die korrekte Demodulation zuständig. Da das LonTalk-Protokoll medienunabhängig implementiert wurde, kann ein LON-Knoten auf beliebigen Medien kommunizieren, indem einfach der entsprechende Transceiver verwendet wird. Die physikalisch unterschiedlichen Parameter werden vom Entwicklungs- oder Installationstool automatisch korrekt eingestellt. Als Medien werden Twisted Pair, Powerline, Funk, Koaxkabel, Infrarot, Lichtwellenleiter und Ultraschall verwendet. Allen Medien gemeinsam ist die Forderung nach universeller Verwendbarkeit und sicherer Funktion auch bei gestörter Umgebung. Bei offenen Medien wie Funk, Powerline usw. kann eine exklusive und ungestörte Übertragung in der Regel nicht garantiert werden, um so wichtiger sind hier eingebaute Funktionen zur Filterung und Fehlerkorrektur. Aber auch bei exklusiv genutzten Medien wie Twisted Pair sind Störungen möglich, zum Beispiel durch induktive Einkopplung oder durch Fehler bei der Installation. Wichtige Forderung an Twisted-Pair-Transceiver sind daher galvanische Trennung, Verpolungssicherheit, großer Common Mode Range und Unempfindlichkeit gegen Störimpulse. Da die früher eingesetzte EIA-485-Technik diese Anforderungen nicht erfüllt, hat Echelon verschiedene Transceiver mit Transformatorkopplung entwickelt.

Mit dem TPT/XF-78 und TPT/XF-1250 stehen zwei passive, transformatorgekoppelte Transceiver für differentielle Übertragung zur Verfügung mit Datenübertragungsraten von 78 kBit/s bzw. 1,25 MBit/s. Diese Transceiver werden in Bustopologie verdrahtet, erlauben Stichleitungen von 0,3 m und 3 m, getrennte Masse, Segmente bis zu 64 Knoten und bieten einen Common Mode Range von ±277 V RMS.

Die Bustopologie setzt neben korrekter Verdrahtung eine saubere Terminierung der beiden Leitungsenden voraus. Dies nicht immer gegeben, beispielsweise wird von manchen Installateuren das Kabelende hinter dem letzten Knoten einfach aufgerollt und im Kabelschacht verstaut. Auch muss im ungünstigsten Fall eine Hin- und Rückleitung zu jedem Knoten gezogen werden, was die reale Ausdehnung der Anlage einschränkt oder unnötig viel Kabel verbraucht. Diese Nachteile werden beim Einsatz eines Transceivers für freie Topologie umgangen. Neben der Flexibilität bei Verdrahtung und Terminierung bietet freie Topologie aber noch weitere Vorteile. So ist bei einer Ringarchitektur auch dann noch eine Kommunikation zwischen allen Knoten möglich, wenn die Leitung unterbrochen oder kurzgeschlossen ist. Verschiedene Anlagenteile

können einfach zusammengeschaltet werden, neue Knoten ohne Unterbrechung des Netzwerkbetriebes an das Kabel angeschlossen werden, ohne es aufzutrennen.

Mit dem FTT-10A bietet Echelon einen Transceiver für freie Topologie an, der einen Ringkerntransformator und einen integrierten Chipsatz in einem Gehäuse mit nur 7,2 mm Bauhöhe enthält. Ein spezieller DSP nimmt eine Aufbereitung des vom Neuron Chip gelieferten Signals vor, filtert die harmonischen Oberwellen heraus und überträgt ein Signal mit abgerundeten Flanken. Das Abstrahlverhalten wird damit insbesondere bei ungeschirmten Kabeln wesentlich verbessert. Umgekehrt werden beim Empfang Funktionen zur Echounterdrückung ausgeführt und das Signal unverfälscht an den Neuron Chip weitergeleitet. Nur mit Hilfe dieses Chips ist es möglich, ein Netzwerk in freier Topologie mit nur einer einzelnen Terminierung aufzubauen. Weiterhin ist eine physikalische Repeaterfunktion integriert, so dass Einfach- oder Mehrfach-Repeater simpel und kostengünstig aufgebaut werden können, ohne dass weitere Prozessoren notwendig sind. Der FTT-10A erfüllt die Anforderungen für den Betrieb in elektrostatisch gestörter Umgebung nach IEEE 801.

Ein ähnlicher Chipsatz wird in dem Link Power Transceiver LPT-10 verwendet. Statt des Transformators enthält der LPT-10 einen DC-DC-Wandler und ermöglicht so die Fremdspeisung der Netzwerkknoten direkt über das Twisted Pair selber. Der Verzicht auf eine lokale Spannungsversorgung und die Verwendung eines normalen Zweidrahtkabels ist nicht nur für die Installationstechnik interessant. Genauso wie ein Bewegungsmelder klein und kostengünstig realisiert werden kann, profitieren Geräte in allen Umgebungen hiervon. Der LPT-10 ist kompatibel mit FTT-10/FTT-10A und kann in einer Mischbestückung am selben Kabel betrieben werden. Ein spezielles Interface für die Spannungsversorgung, das LPI-10, ist als fertige Baugruppe oder als Schaltungsanleitung erhältlich.

Produkt	Ü-Rate	Topologie	Knoten pro Segment	Distanz	Typ
TPT/XF-78	78 kbps	Bus	64	1400m	Trafo-isoliert
TPT/XF-1250	1,25 Mbps	Bus	64	130m	Trafo-isoliert
FTT-10	78 kbps	Bus	64	2700m	Trafo-isoliert
FTT-10	78 kbps	Free	64	500m	Trafo-isoliert
LPT-10	78 kbps	Bus	128	2200m	Link Power
LPT-10	78 kbps	Free	128	500m	Link Power

Besonderes Know-How besitzt Echelon ebenfalls auf dem Gebiet der Powerline-Übertragung. Mit PLT-10A, PLT-21 und PLT-30 stehen drei Transceiver zur Verfügung, die weltweit alle Einsatzfälle abdecken. Alle drei sind intelligente Transceiver, das heißt, dass ein spezieller Chipsatz nicht nur die jeweilige Modulation vornimmt, sondern auch Filterung von Störungen, Fehlerkorrektur und Kollisionsbehandlung. Für den Einsatz im öffentlichen

Stromnetz sind in Europa der PLT-30 für das A-Band und der PLT-21 für das C-Band zugelassen. PLT-10A und PLT-30 arbeiten nach dem Spread-Spectrum-Verfahren, verteilen die übertragene Information also auf einen weiten Frequenzbereich. Dies ist im für die private Nutzung vorgesehenen C-Band nicht möglich. Echelon hat daher ein spezielles Schmalbandverfahren nach dem BPSK-Prinzip entwickelt und neue Methoden der Fehlerkorrektur eingebaut, so dass der PLT-21 (und sein Vorgänger, der PLT-20), obwohl ursprünglich nur für Europa entwickelt, inzwischen weltweit eingesetzt wird.

Die Powerline-Transceiver sind jedoch nicht nur für den Betrieb am 230V- oder 380V-Stromnetz vorgesehen. Vielmehr können diese Transceiver an beliebige Spannungen geklemmt werden, Gleichstrom oder Wechselstrom, oder an spannungsfreie Leitungen. Auch die Art des verwendeten Kabels ist beliebig, die Kommunikation kann über jedes elektrisch leitende Medium erfolgen. In der Fördertechnik beispielsweise wird der PLT-21 eingesetzt, um sowohl die Versorgungsspannung als auch die Daten per Schleifring direkt von Schienen abzunehmen. Für den PLT-21 ist ein spezieller Booster erhältlich, um die Übertragungsleistung (bei nichtöffentlichen Medien) noch einmal zu erhöhen. Damit können Dämpfungen von mehr als 75 dB überwunden werden, Entfernungen von 10 km und mehr sind möglich. Damit sind viele Anwendungen, z. B. im Bahnbereich, überhaupt erstmalig zu realisieren. Auch die Integration in Haushaltsgeräte nimmt durch die Möglichkeit der Plug-and-Play-Installation zu.

Da die Übertragungsqualität beim öffentlichen Stromnetz stark von den Umgebungsbedingungen abhängt (Dämpfung, Störungen, etc), können diese mit speziellen Messgeräten, den PLCA-10, -21, -30 (Powerline Communication Analyzer) getestet werden. Auch das korrekte Design des Kommunikationsteils eines Powerlineknotens kann mit diesen Geräten leicht überprüft werden. Sie sollten daher zum Inventar jedes Entwicklungslabors gehören.

Transceiver für weitere Medien werden von anderen Herstellern angeboten. Wenn sie fest definierte Parameter einhalten, werden sie von Echelons Entwicklungs- und Installationstools direkt unterstützt. Häufig eingesetzt werden Transceiver für Funk, Koaxkabel, Infrarot und Lichtwellenleiter. Echelon veröffentlicht über die Homepage im Internet eine Liste aller freigegebenen Transceiver.

Um ein Netzwerksegment zu erweitern, können Repeater eingesetzt werden. Verschiedene Medien werden durch Bridges und Router verbunden und der Datenverkehr intelligent gefiltert. In FTT-10A und LPT-10 ist eine Repeaterfunktion integriert, für den Aufbau von beliebigen Routern bietet Echelon mit dem RTR-10 ein Modul mit integrierter Router-Firmware an. Fertiggeräte mit diesem Modul (z. B. Echelons LonWorks Router) können als

konfigurierter oder lernender Router, als Bridge oder als Repeater konfiguriert werden.

Für die schnelle Entwicklung von Prototypen und Geräten werden sogenannte Kontrollmodule eingesetzt, die auf einer kleinen Platine einen vollständigen Knoten mit Transceiver unterbringen. Sie sind pinkompatibel zu den Testadaptern der Entwicklungstools, die direkt in der Zielhardware eingesteckt werden. Nach der Programmerstellung werden die Testadapter dann gegen die Kontrollmodule ausgetauscht. Die Module mit Flashspeicher können weiterhin direkt in den Produkten umprogrammiert werden.

Interfaces zur Ankopplung beliebiger Geräte an ein LON werden als Fertiggeräte und als Bausteine angeboten. Als Fertiggeräte sind verschiedene PC-Steckkarten, PC-Cards (PCMCIA II) und serielle Adapter erhältlich. Die Integration eines LON-Interfaces in eigene Geräte kann durch den Einbau eines Interface-Moduls (LTS-10, NSS-10, PSG-10) erfolgen. Alternativ ist eine spezielle Interface-Firmware als Microprocessor Interface Program (MIP) zur Ankopplung des Neuron Chip an ein beliebiges Host-System vorgesehen. Das MIP findet Verwendung in diversen Interfaces von Drittherstellern, u.a. für PCI, PC/104, AT-96, VME-Bus, S-Bus, usw. Für die Erstellung von Gateways gibt es programmierbare Module und Fertiggeräte zur seriellen Ankopplung (PSG/2 und PSG-10), für viele Feldbusse werden passende Gateways auf dem Markt angeboten. Auch diese Produkte listet Echelon im Internet auf.

Die Erstellung von Hostsoftware wird durch eine Reihe von Software-Produkten unterstützt. Für DOS, Windows, Windows 95 und NT stehen Interface-Treiber zur Verfügung, für andere Betriebssysteme oder Prozessoren ist Sourcecode erhältlich. Viele Hersteller bieten Treiber mit ihren Produkten an, zum Beispiel für UNIX und QNX, Motorola liefert Treiber für die Prozessorreihen HC11 und 683xx. Für die eigene Entwicklung von Interfaces gehört entsprechende Software zum Lieferumfang des MIP Developer's Kit. In der Firmware von SLTA/2, SLTA-10 und LTS-10 ist die Unterstützung von handelsüblichen Modems bereits integriert. Fernwartung und -diagnose sind damit ohne zusätzlichen Aufwand realisierbar. Zur einfachen Anbindung von Windows-Applikationen steht ein DDE-Server zur Verfügung, der bei Produkten wie Wonderware's InTouch auch den Fast-DDE-Mechanismus benutzt. Unter Windows 95 und NT wird mit LNS for Windows der OLE-Mechanismus benutzt. LNS for Windows (LonWorks Network Services) ist eine Entwicklungssoftware, die einen OLE-Server unter der LonWorks Component Architecture (LCA) bereitstellt. Sehr einfach kann somit auch Komponentensoftware entwickelt oder zur Entwicklung eingesetzt werden. Ebenso wie LNS for Microcontrollers stellt auch LNS for Windows ein High-Level-API für komplexe Netzwerkmanagement-Funktionen zur Verfügung. Aber auch die Programmierung von Visualisierungs-, Monitor-, Diagnose- und Steuerungsprogrammen wird mit LNS/LCA und Programmierumgebungen wie Delphi, Visual C++ oder Visual Basic wesentlich vereinfacht. Sollen keine

Netzwerkmanagement-Funktionen implementiert werden, so kann die Hostapplikation auch direkt auf den Treibern aufgesetzt werden, beispielsweise bei einem Embedded Prozessor. In allen Fällen ist die Programmierschnittstelle zum Treiber hin gleich, so dass dieselbe Software auf beliebigen Interfaces eingesetzt werden kann.

Kap.4: Programmierung des Neuron Chips

Die Programmiersprache Neuron C

Die Programmierung des Neuron Chips erfolgt in der Regel in Neuron C, wodurch die Realisierung einer Applikation einfach, schnell, sicher und kostengünstig wird[1]. Neuron C basiert auf ANSI C und stellt die Ressourcen des Neuron Chips und seines Betriebssystems durch spezielle Erweiterungen des Sprachumfangs bereit. Diese Erweiterungen betreffen vor allem die Datenkommunikation, die Eventsteuerung, Timer- und I/O-Objekte. Dafür fehlen gegenüber ANSI C alle Konstrukte zur Unterstützung von Standard-I/O und Filesystem. Fließkomma-Arithmetik und spezielle Routinen zur Multiplikation/Division werden durch Libraries zur Verfügung gestellt.

Das eventgesteuerte Neuron-Betriebssystem stellt sich dem Programmierer in Form eines Task-Schedulers für den Applikationsprozessor dar. Ein Neuron C-Programm besteht demnach aus einer Ansammlung von Tasks, deren Ausführung durch das Auftreten bestimmter Events veranlasst wird. Diese Art der Programmierung ist beispielsweise von MS-Windows bekannt. Eine Task wird eingeleitet durch das Schlüsselwort "when", gefolgt von dem oder den Events als Parameter:

```
[priority] when (Event)
{
   // When Body
}
```

Durch die Angabe des zusätzlichen Schlüsselwortes "priority" kann eine Task eine bestimmte Priorität erhalten, deren Wertigkeit unmittelbar durch die Reihenfolge im Sourcecode festgelegt wird. Der Scheduler vergleicht nach dem in Bild 4.1 festgelegten Zyklus die eingetroffenen Events und die definierten Tasks. Trifft er auf das entsprechende Event, so wird die zugehörige Task ausgeführt. Die priorisierten Tasks sind dabei in einer Kette angeordnet, die nicht priorisierten Tasks werden in einem Round-Robin-Zyklus überprüft. Hierbei wird nach Überprüfung jeder einzelnen Task und deren eventueller Ausführung die Kette der priorisierten Events erneut durchlaufen. Bei jedem Durchlauf werden im Startpunkt ("Embark Point") bestimmte Systemfunktionen angestoßen. Hierzu gehören der Austausch von Daten und Events mit dem Netzwerkprozessor, das Rücksetzen des Watchdog Timers und die Bufferverwaltung.

[1] Neben Neuron C Compilern gibt es Front-Ends beispielsweise für IEC-1131.

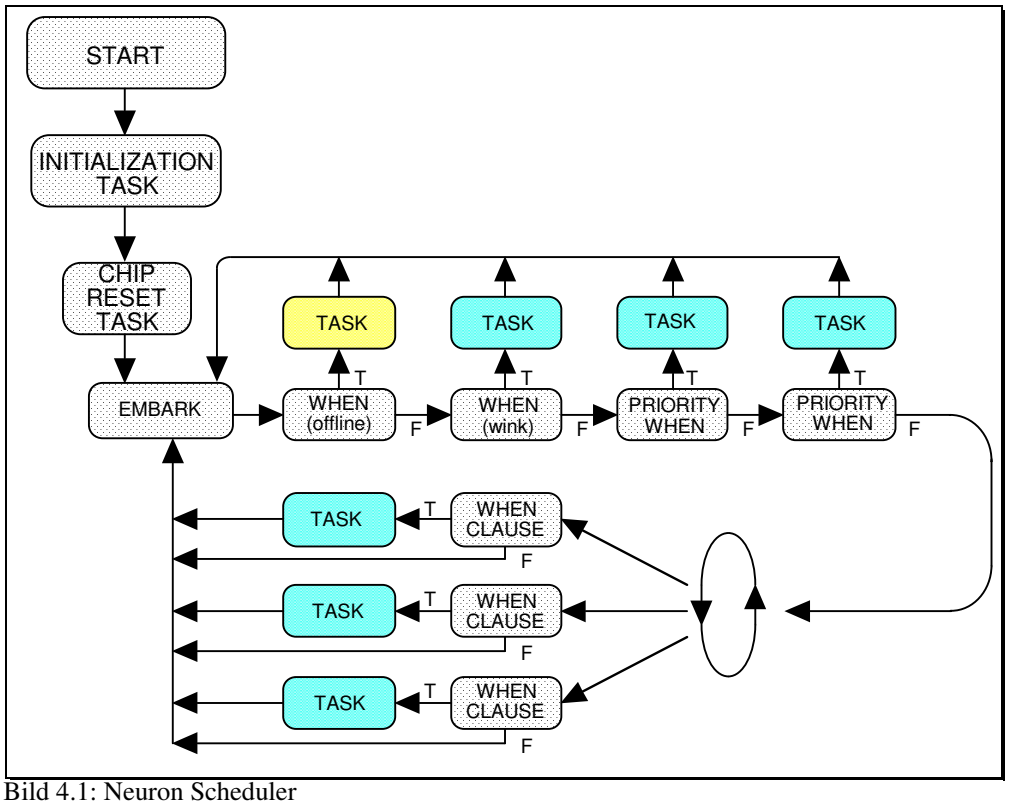

Bild 4.1: Neuron Scheduler

Events

Events werden in verschiedene Kategorien eingeteilt. Zu den vordefinierten Events gehören I/O-Events (Änderungen im I/O-Block), Kommunikations-Events (Eintreffen einer neuen Netzwerkvariablen oder Nachricht, Reaktion auf Fehler etc.), Timer-Events (Timer abgelaufen) sowie verschiedene System-Events (Reset, Netzwerkmanagement-Kommandos). Weiterhin kann der Anwender eigene Ereignisse definieren. Jeder in Neuron C gültige Ausdruck wird hierbei vom Scheduler als Event erkannt, wenn er von Null verschieden ist, wobei die Verwaltung dieser Events vom Anwender vorgenommen wird.

Beispiele für vordefinierte Ereignisse sind:

```
io_changes (I/O-Objekt) [by...|to...]
io_update_occurs (I/O-Objekt)
nv_update_occurs (Netzwerkvariable)
nv_update_fails (Netzwerkvariable)
timer_expires [(Timername)]
```

Bei der Formulierung der when-Klauseln können mehrere Events mit einer UND- bzw. ODER-Verknüpfung kombiniert werden. Dies ermöglicht die elegante und flexible Steuerung des Programmablaufes ohne die sonst notwendigen if-/else-Abfragen.

Neuron C-Objekte

In Neuron C werden die Ressourcen des Neuron Chip durch Objekte angesprochen. Zu diesen Objekten gehören Netzwerkvariablen, Timer und I/O-Objekte.

Netzwerkvariablen werden wie normale Variablen in C deklariert und verwendet. Bei der Deklaration wird das Schlüsselwort network input oder network output vorangestellt:

```
network input type identifier;
network output type identifier;
```

Beispiele:

```
network output int nvoZustand;
network input SNVT_temp nviTemperature;
```

Als Typen sind die in C üblichen Typen zulässig, ebenso Strukturen, Felder und Pointer. Die maximale Datenlänge beträgt dabei 31 Bytes. Darüber hinaus bietet Neuron C eine Sammlung von Datentypen mit definierter inhaltlicher Bedeutung wie physikalische Größen oder Datentypen für bestimmte Branchen und Anwendungen. Diese Standard-Netzwerkvariablen-Typen (SNVTs, gesprochen "snivits") werden von der LonMark Interoperability Association definiert und garantieren eine interoperable Programmierung. Der Typ SNVT_temp ist ein Beispiel aus dieser Liste, weitere Beispiele zeigt der Ausschnitt in Tabelle 4.1.

Variable	Name	Auflösung	Bereich	Bits
Temperatur	SNVT_temp	0,1º C	-274,0 .. 6.279,5ºC	16
Temperatur	SNVT_temp_f	ANSI/IEEE 754	-273,17 .. 1E38ºC	32
Strom	SNVT_amp	0,1 A	-3.276,8 .. 3.267,7A	16
String	SNVT_str_asc	char	30 Zeichen	248
Wert, diskret	SNVT_lev_disc	enum{}	OFF, LOW, MED, HIGH, NUL	8
Zeit	SNVT_time_stamp	YYYY:MM:DD:HH:MM:SS	0:0:0:0:0:0 .. 3000:12:31:23:59:59	56

Tabelle 4.1: Auszug aus der SNVT Master List

Eine einfache Zuweisung eines Wertes zu einer Netzwerk-Ausgangsvariablen bewirkt das automatische Versenden dieser Variablen. Es werden im Programm weder Empfänger noch die Protokollparameter (Acknowledged, Anzahl Wiederholungen etc.) festgelegt, die Kommunikationsbeziehungen werden vielmehr bei der Installation des Knotens definiert. Nur so ist es möglich, eine wirklich universelle Applikation zu erstellen. Gleichwohl kann aber auch explizite Adressierung und explizites Messaging (mit impliziter oder expliziter Adressierung) verwendet werden, allerdings ist eine solche Programmierung immer anwendungsspezifisch und nicht interoperabel. Mit expliziten Nachrichten lassen sich Datenblöcke von bis zu 228 Bytes in einem Telegramm versenden. Auch zum Empfangen einer Netzwerkvariablen sind außer der Formulierung einer when-Klausel keine weiteren Vorkehrungen nötig. Das Event wird automatisch ausgelöst und die zugehörige Task ausgeführt:

```
when (nv_update_occurs (nviTemperature))
  {
  if (nviTemperature <= 17)
    heizen();                    // dort steckt Ihr eigenes Know-How
}
```

I/O-Objekte werden wie Variablen deklariert und fortan im Programm unter ihrem Namen angesprochen. Zur Ein- und Ausgabe stehen Funktionen bereit, die die Umsetzung auf die physikalischen Chip-Ressourcen vornehmen (I/O-Pins, Auflösung, etc.). Die Syntax lautet:

```
pin input device_type [options] device_name;
pin output device_type [options] device_name [= initial_level];

return_value = io_in (device [, args]);        /* Einlesen vom -
                                                  I/O-Objekt device */
io_out (device, output_value [, args]);        // Schreiben auf
                                                  I/O-Objekt device */
```

Je nach I/O-Modell werden zusätzliche Schlüsselwörter und Parameter ergänzt. Bild 4.2 listet die in der Firmware-Version 7 enthaltenen I/O-Modelle auf.

Die I/O-Logik des Neuron Chips umfasst Flags, Latches und Register, die ebenfalls als Neuron C-Objekte angesprochen werden. Die Zustandsänderung eines I/O-Objektes wird beispielsweise automatisch in der eingebauten Variablen input_value erfasst. Mit diesen Sprachelementen können nun bereits vollständige Programme geschrieben werden. Das folgende Programm eines Sensors zeigt eine Task, die den Zustand eines Tasters an einem der I/O-Pins erfasst und bei Änderung den jeweiligen Wert als Netzwerkvariable versendet:

```
network output boolean nvoSwitchValue;
IO_4 input bit ioSwitch;

when (io_changes (ioSwitch))
{
  nvoSwitchValue = (boolean)input_value;  /* Netzwerkvariable wird
                                             automatisch versendet */
}
```

Der Empfänger, im LonMark-Sprachgebrauch Aktor genannt, führt beim Empfang dieser Netzwerkvariablen folgende Task aus:

```
network input boolean nviLedValue;
IO_1 output bit LED_green;

when (nv_update_occurs (nviLedValue))
{
  io_out (LED_green, nviLedValue); // Umschalten der grünen LED
                                   // an Pin IO_1
}
```

Der Neuron Chip stellt dem Programmierer bis zu fünfzehn **Timer** zur Verfügung, deren Auflösung Sekunden oder Millisekunden betragen kann. Auch diese Timer werden als Neuron C-Objekte deklariert und im Programm nur noch unter ihren Namen angesprochen. Sie werden durch Zuweisen eines Wertes gestartet (>0) bzw. angehalten (0) und können wie Variablen ausgelesen werden. In der Regel werden sie benutzt, um durch ihr Ablaufen ein Event auszulösen und so in festen Zeitabständen eine Task auszuführen:

```
stimer repeating blinkfrequenz = 60;    // Sekunden-Timer, fortlaufend
mtimer blinkdauer;                      // Millisekunden-Timer

when (timer_expires (blinkfrequenz))
{
   io_out (LED_green, ON);        // alle sechzig Sekunden geht grüne
                                  // LED an
   blinkdauer = 500;              // Zweiten Timer starten
}

when (timer_expires (blinkdauer))
{
   io_out (LED_green, OFF)        // nach ½ Sekunde wieder ausschalten)
}
```

Wie einfach die Entwicklung einer netzwerkfähigen Applikation sein kann, wird am Beispiel eines Empfängers für eine handelsübliche Infrarot-Fernbedienung deutlich. Das Programm benutzt das I/O-Modell "infrared input", um den von der Fernbedienung gesendeten Befehl zu decodieren und auf das Netzwerk weiterzugeben. Beim Programmstart (Reset oder power-on) wird das Reset-Event ausgelöst, durch die Reset-Task werden alle korrespondierenden Eingangsvariablen im Netzwerk automatisch initialisiert. Das Wink-Event wird durch das Netzwerkmanagement-Kommando "wink" ausgelöst, der physikalische Ort eines Knotens kann auf diese Weise ermittelt werden. Dieser Mechanismus ist bei der Installation hilfreich, wenn nicht auf den Service Pin zurückgegriffen werden kann. Die Funktion `io_in()` legt

(wie bei allen Timer/Counter-Inputmodellen) die dekodierten Daten gleich in dem Array `data_in` ab.

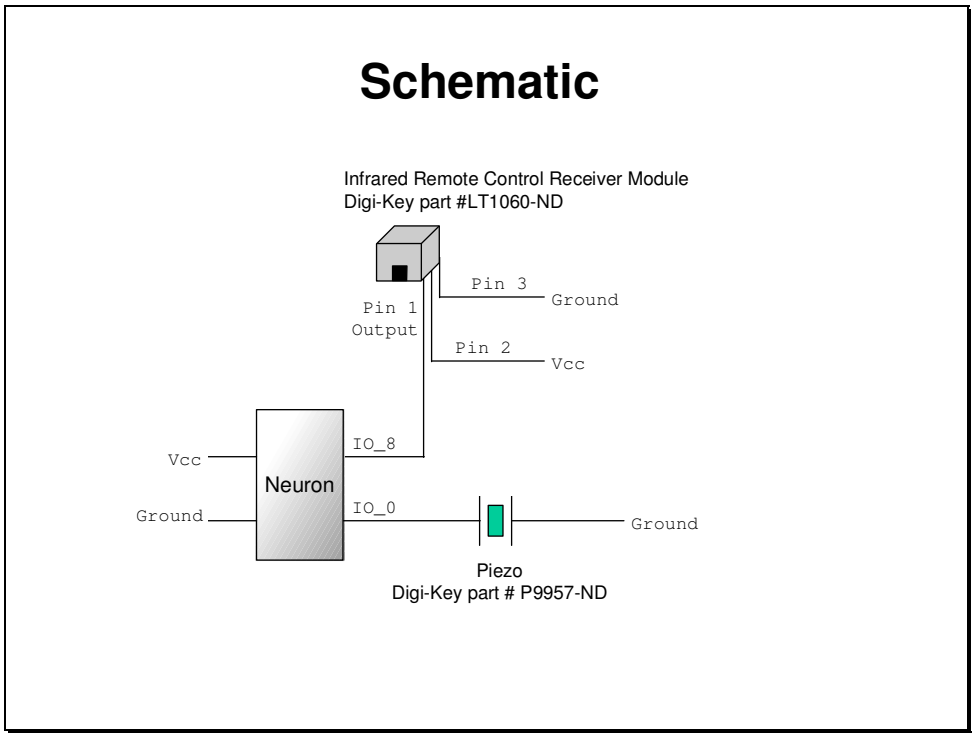

Bild 4.3: Infrarot-Empfänger

```
//--------------- SONY.NC ------------------------------------------
#include <stdlib.h>
#include <control.h>
#include <sony.h>

//--------------- Declarations -------------------------------------
#define     MAX_PERIOD      65407UL
#define     THRESHOLD       65465UL
#define     NUM_BITS        12

IO_4    input   infrared ded invert clock(7) ir_decoder;  // Reader
IO_4    leveldetect                          ir_start;
IO_0 output frequency clock (1) ioPiezo;                 // Beeper
mtimer beepTimer;
network output unsigned long remote_command;

//--------------- Reset Task ---------------------------------------
when (reset)
{
   remote_command = 0;                 // Start in a known state
}

//--------------- Wink Task ----------------------------------------
when (wink)
{
   io_out (ioPiezo, 4000);             // Start beep.
   beepTimer = 250;                    // Beep will shutdown in about
}                                      // ¼ second.

//--------------- get command and propagate it----------------------
when (io_changes (ir_start) to 1)      // Catch pulse edge....
{
   union {                             // Handy for manipulating
       unsigned long irbl;             // 8 or 16 bit quantities.
       unsigned int  irbi[2];          // Neuron uses 8 bit integers,
    } irb;                             // 16 bit longs.

    unsigned data_in[2];               // Buffer to store IR bit
stream.

    unsigned int temp;                 // Scratch variables
             int bits_read;

    bits_read = io_in (ir_decoder, data_in,
                       NUM_BITS, MAX_PERIOD, MAX_PERIOD + 58UL);

    if (NUM_BITS == bits_read)         // Did we get a complete
message?
    {
        temp = data_in [1] >> 4;       // Zero high order bits.

        irb.irbi [1] = data_in [0];    // Get things into the correct
        irb.irbi [0] = temp;           // byte order.

        io_out (ioPiezo, 3000);        // Turn on beeper
        beepTimer = 100;               // 100 ms
    }

    remote_command = irb.irbl;         // Report code to the world.
}

//--------------- Beeper -------------------------------------------
when (timer_expires (beepTimer))       // Turn off beeper
{
      io_out (ioPiezo,0);
}
```

Interoperable Programmierung

Um die Entwicklung interoperabler Produkte zu ermöglichen, gibt die LonMark Interoperability Association neben der SNVT Master List unter anderem die "Application Layer Interoperability Guidelines" heraus. Hier sind neben der Einteilung von Softwareobjekten nach ihrer logischen Funktion vor allem die Verfahren zur Implementation der externen Schnittstelle eines Objektes beschrieben. Mit einer standardisierten Schnittstellenbeschreibung wird nicht nur die Zusammenarbeit von Produkten verschiedener Hersteller möglich, auch werten die auf LNS basierenden Netzwerkmanagement-Tools die Informationen über LonMark-Objekte, Schnittstellen und Funktionsprofile aus. Dies alleine ist schon Grund genug, nach den LonMark-Richtlinien zu programmieren.

Nach LonMark werden die folgenden grundlegenden Funktionsobjekte definiert: Open Loop Actuator, Closed Loop Actuator, Open Loop Sensor, Closed Loop Sensor, Controller und Node Object. Darüber hinaus werden fortlaufend Funktionsprofile für bestimmte Produktgruppen erstellt. Für jedes einzelne dieser Objekte werden bestimmte vorgeschriebene und optionale Netzwerkvariablen, Konfigurationseigenschaften und Selbstdokumentations-mechanismen definiert. Mit Hilfe der Funktionsprofile können Produkte für unterschiedliche Umgebungen konfiguriert und ihre Funktion und Eigenschaften ausgelesen werden.

Die objektbasierte Programmierung nach diesen Richtlinien erleichtert die Entwicklung eines Produktes, zumal auch die Entwicklungstools diesen Standard unterstützen. Dem Codegenerator des NodeBuilders beispielsweise genügt die Auswahl von LonMark-Objekt und der SNVTs aus seinen Listen, um ein Codegerüst für die jeweilige Applikation zu generieren.

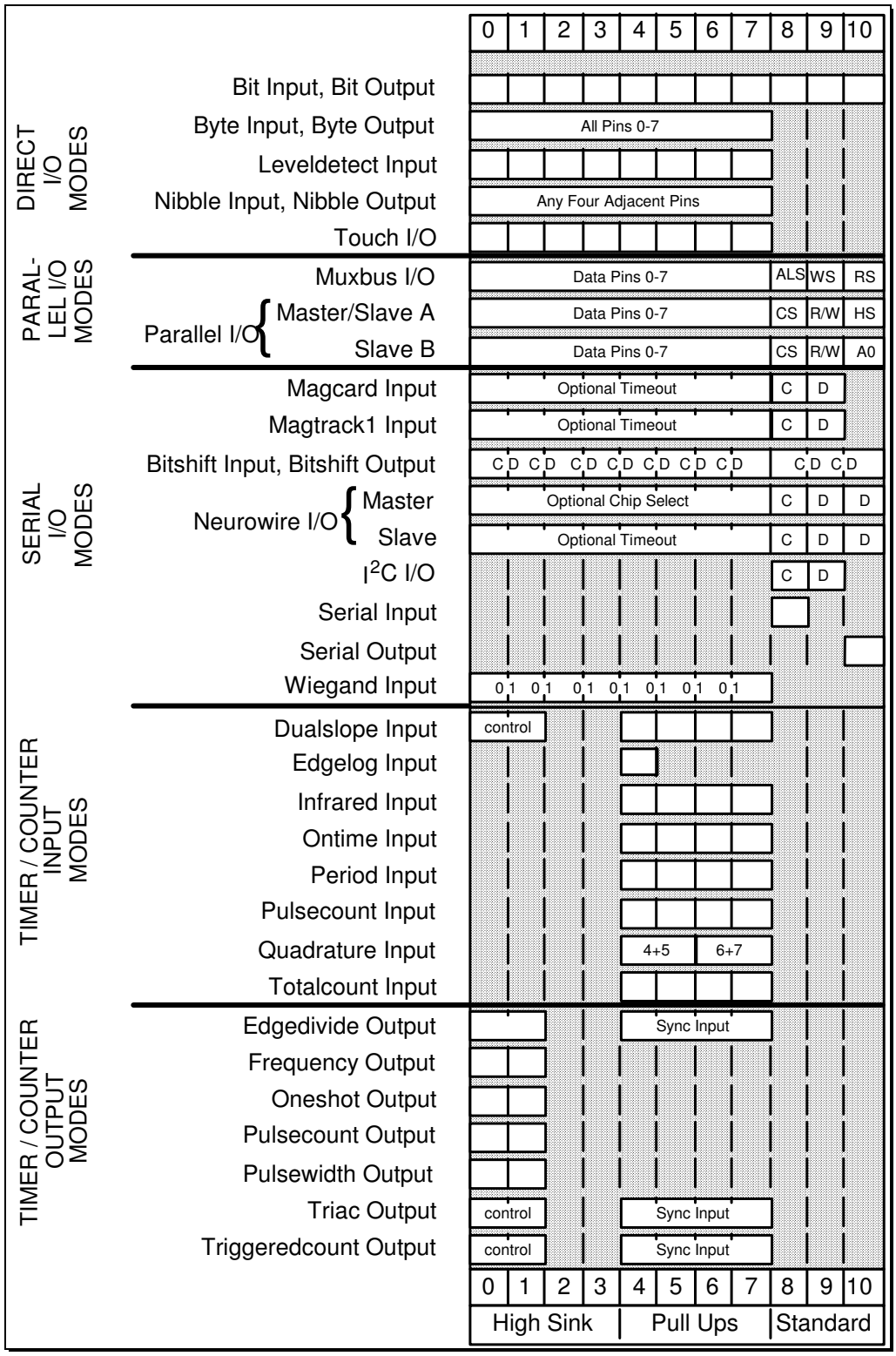

Bild 1.2.7.2: Neuron Chip I/O-Modelle

Kap. 5: Entwicklungstools

NodeBuilder, LonBuilder, Field Compiler & Co.

Zur Applikationsentwicklung stehen von Echelon zwei leistungsfähige Entwicklungstools zur Verfügung: LonBuilder und NodeBuilder. Wie der Name bereits andeutet, ist der NodeBuilder für die Entwicklung einer einzelnen Knotenapplikation konzipiert. Der LonBuilder ist zur Entwicklung vollständiger Systeme ausgelegt, er enthält daher Tools und entsprechende Hardware zum parallelen Entwickeln und Testen mehrerer Knoten in einem echten Netzwerk.

Der NodeBuilder ist das kostengünstige "Einsteigertool" und ist vergleichbar mit den Entwicklungskits für andere Mikroprozessoren/-controller. Ähnlich wie diese besteht er aus einer Testhardware, auf der die Applikation läuft und ausgetestet wird, einer PC-Interfacekarte und der eigentlichen Entwicklungssoftware für den PC. Hierzu gehören ein Editor, ein Neuron C Crosscompiler, ein komfortabler Debugger, Linker und Loader, ein Netzwerkvariablen-Browser und ein DDE-Server zur Anbindung an externe Windows-Software. Die Interfacekarte "PCNSS" enthält das NSS-10, ein spezielles Modul zum Netzwerkmanagement, das eigenständig die Kommunikation mit dem Testknoten abwickelt. Sie kann aber auch als normales LON-Interface eingesetzt werden. So können Netzwerke mit dem LonMaker-Tool installiert werden oder Daten zwischen dem Netzwerk und einer Hostapplikation über den DDE-Server ausgetauscht werden. Auf die Interfacekarte werden die sog. SMX-Transceiverkarten aufgesteckt, so dass alle Standard-Übertragungsmedien eingesetzt werden können. Die gleichen SMX-Transceiverkarten werden in dem Testknoten installiert, die Übertragung der Applikation vom PC aus erfolgt per Download über das echte Netzwerk.

Der Testknoten enthält in einem kleinen Gehäuse das LTM-10 Modul, das im wesentlichen aus einem Neuron 3150, 32 KB SRAM und 32 KB Flash Memory besteht. Die Applikation wird zum Testen in das RAM geladen, gelinkt mit einem Debug-Kernel. Hieraus wird zweierlei deutlich: es handelt sich um einen Software-Debugger (im Gegensatz zum LonBuilder-Emulator) und die Applikation ist auf 32 KB Codegröße beschränkt. Ist der Code ausgetestet, kann die Applikation dauerhaft ins Flash geladen werden. Außerdem kann die NodeBuilder-Software das LTM-10 in einen speziellen Modus schalten: dann steht der gesamte Flashspeicher für den Code und zusätzlich ein Teil des RAM für Daten und Buffer zur Verfügung. Damit ist der Testknoten ein exzellenter Prototyping-Knoten auch für daten- und kommunikationsintensive Applikationen. Er ist deshalb auch einzeln als "LonTalk Node" erhältlich, auch das LTM-10 kann einzeln bezogen werden und empfiehlt sich beispielsweise für Kleinserienprodukte.

Der eigentlichen Applikationsentwicklung geht die Definition der Zielhardware voraus. In einem "Device Template" wird der Neuron-Typ angegeben (z.B. 3120E2), die Memory Map (nur für 3150) und der verwendete Transceiver (um die passenden Kommunikationsparameter zu laden). Wenn die endgültige Applikation erst nach Installation des Gerätes über das Netzwerk geladen werden soll, dann müssen mindestens die Kommunikationsparameter und beim 3150 die Firmware zuvor geladen sein. Beim 3120 ist die Firmware ja im On-Chip-ROM enthalten, auch das LTM-10 enthält seine eigene Firmware. Diese umfasst im übrigen auch das MIP, so dass das Modul auch zum Aufbau eines Host-Interfaces benutzt werden kann.

Da sowohl die Zielhardware wie auch die Programmierverfahren unterschiedlich sein können, exportiert die NodeBuilder-Software verschiedene Code-Dateien: *.NRI, *.NXE und *.NEI. Mit der NRI-Datei (Neuron ROM Image) wird das PROM (meist ein EPROM) für den 3150 "gebrannt", hierzu wird ein handelsübliches Programmiergerät verwendet. Analog wird die NEI-Datei (Neuron EEPROM Image) für den 3120 verwendet, hierfür stehen von Echelon und System General spezielle Programmiergeräte zur Verfügung. Die NXE-Datei schließlich enthält nur den ausführbaren Code (eXecutable) und wird von Tools wie dem LonMaker über das Netzwerk geladen.

Die Applikationsentwicklung geht im üblichen Zyklus vonstatten: Editieren - Compilieren - Debuggen - Testen. Zum Testen des richtigen Verhaltens des Programms wird der Netzwerkvariablenbrowser verwendet. Er zeigt die Werte der Ausgangsnetzwerkvariablen so an, wie sie vom Knoten über das Netzwerk gesendet werden würden. Umgekehrt kann man die Eingangsvariablen auf bestimmte Werte setzen, was ein Event auslöst und die Ausführung der zugehörigen Task zur Folge hat. Ein direktes Zusammenarbeiten mehrerer Knoten oder das Debuggen im Netz ist mit dem NodeBuilder nicht zu realisieren, dies bleibt dem LonBuilder vorbehalten. Ein, wenn auch sehr umständlicher, Umweg besteht aus dem Einsatz eines zusätzlichen Installations- und Bindertools, wie LonMaker oder, etwas einfacher, der Easylon Toolbox Lite. Nach jedem Ladevorgang muss aber, wenn keine zweite Interfacekarte eingesetzt wird, das NodeBuilder-Tool beendet werden. Dann muss das Installationstool aufgerufen werden, um Installation und Binding vorzunehmen. Danach ist der ganze Vorgang umgekehrt vorzunehmen. Dabei geht natürlich wertvolle Entwicklungszeit verloren. Da auch der Ladevorgang der Applikation bei größeren Programmen mehrere Minuten betragen kann, ist von diesem Verfahren abzuraten.

Der Testknoten besitzt eine I/O-Schnittstelle, an der die I/O-Signale sowie die Steuerleitungen des Neuron Chips (Reset, Service, etc.) aus dem Gehäuse herausgeführt werden. Hier kann man entweder den mitgelieferten Gizmo-3 anschließen oder über einen Platinenadapter die eigene Peripherie-Hardware des zu entwickelnden Produktes anschließen, wodurch ein In-Circuit-Test möglich wird. Dieser Adapter hat dasselbe Pin-Out wie die Kontrollmodule

von Echelon, so dass statt des Testknotens später einfach ein solches Kontrollmodul aufgesteckt werden braucht, um einen vollständigen Knoten zu erhalten. Der Gizmo-3 ist eine kleine Platine mit allerlei Peripherie-Elektronik, wie Taster, LEDs, 7-Segment-Display, Temperatursensor, A/D-Wandler, Drehgeber etc. Gerade in der Anfangszeit ist er ein wertvolles Hilfsmittel zur Erlernung der I/O-Programmierung, wird aber auch gerne für Demonstrationsaufbauten etc. verwendet.

Der LonBuilder ist ein integriertes Tool zum Entwickeln ganzer Systeme. Mit ihm können mehrere Knoten gleichzeitig programmiert und getestet werden, diese können sogar in einem laufenden Netzwerk integriert sein. Er enthält dazu alle notwendigen Netzwerkmanagement-Tools zur Installation von bis zu 256 Knoten und zum Binding der Netzwerkvariablen (und Message Tags). Über die auch beim NodeBuilder enthaltenen Tools hinaus besitzt er Tools zur Analyse des Netzwerkes und dessen Performance selber, etwa einen vollständigen Protocol Analyzer. Alle diese Tools sind unter einer Oberfläche zusammengefasst, dem Navigator. Ein Projektmanager sorgt dafür, dass alle Tools immer synchronisiert den aktuellen Status jeden Knotens kennen und man direkt von einem Tool zum nächsten umschalten kann, etwa vom Protocol Analyzer zum Editor. Um nach einer Codeänderung im Editor die Applikation erneut zu testen, genügt ein einziger Mausklick auf "Automatic Build". Sofort werden alle geänderten Programme kompiliert, auf die richtigen Emulatoren geladen, eventuell hinzugekommene Bindings erstellt, die Applikationen gestartet, die Debugger neu gestartet, etc. Die notwendigen Informationen hierzu werden in einer Datenbank gehalten, die auch später von anderen Tools wie dem LonMaker benutzt werden kann. Jedes Entwicklungsprojekt wird in einem eigenen Verzeichnis auf der Festplatte gespeichert, so dass man bei Zugriff auf ein bereits älteres Projekt immer die passende Umgebung vorfindet.

Die Hardware der Entwicklungsstation besteht aus einem 19"-Gehäuse mit sieben Einschüben. Neben einem Kontrollprozessorboard, das auch die Protocol Analyzer Hardware enthält, können bis zu sechs Emulatoren oder Router eingesetzt werden. Die Grundausstattung besteht aus dem Kontrollprozessor und zwei Emulatoren (bei der Powerline-Version kommt noch ein Router hinzu). Die Emulatoren enthalten neben dem 3150 und 64 KB RAM unter anderem auch Debugger-Logik (Hardware-Breakpoints etc.), so dass ein Debuggen in Echtzeit möglich ist. Insgesamt lassen sich vier dieser Stationen an die PC-Interfacekarte anschließen, so dass man auf bis zu 24 Emulatoren gleichzeitig debuggen kann. Im Gegensatz zum NodeBuilder kann der gesamte externe Codebereich des Neuron Chips von 42 kB genutzt werden, auch sind beliebige Speicherkonfigurationen und Kommunikationsparameter emulierbar.

Die Emulatoren können entweder über aufgesteckte Transceiver (alle Standardtransceiver sind nutzbar) oder über das eingebaute Backplane-Netzwerk kommunizieren. Ein separater Backplane-Bus (10 Mbit/s) wird vom

Kontrollprozessor zur Steuerung des gesamten Systems genutzt, alle Kommunikation mit dem PC (Debugger, Download) beeinflusst also nicht die Abarbeitung der Programme auf den Emulatoren oder die Kommunikation. Auch die Downloadzeiten reduzieren sich gegenüber dem NodeBuilder erheblich.

Die Emulatoren weisen jeweils zwei Steckplätze auf. Einer von ihnen nimmt wie beim NodeBuilder einen SMX-Transceiver auf, so dass die Kommunikation über ein echtes Netzwerkmedium erfolgt und auch externe Knoten eingebunden werden können. Der andere Steckplatz nimmt eine I/O-Interfacekarte auf. Hier können ein Gizmo oder ein I/O-Hardwareadapter angeschlossen werden, so dass der Emulator direkt in die bestehende Hardwareperipherie integriert werden kann.

Weiterhin ist für den LonBuilder ein Routereinschub erhältlich. So kann von der Backplane oder von einem bestimmten Medium auf ein anderes umgesetzt werden, beispielsweise um Powerline-Knoten an einen mit FTT ausgestatteten LonBuilder anzuschließen. Auch können so verschiedene Emulatoren mit verschiedenen Transceivern ausgerüstet werden. Der Router ist dabei für die Entwicklungstools transparent, da die interne Kommunikation der Tools über die Backplane erfolgt. Der Router kann auch stand-alone außerhalb der Entwicklungsstation betrieben werden, es ist dann lediglich ein Netzteil an der hierzu vorgesehenen Buchse anzuschließen.

	LonBuilder	**NodeBuilder**
Neuron Emulator Support	Bis zu 24	NEIN
Laden der Applikation	Über das Netzwerk oder die Backplane	Nur über das Netzwerk
Anzahl Knoten	bis zu 256	1
Router Support	bis zu 8	NEIN
Debugging	Hardware, über Backplane	Software, über Netzwerk
Debugging Sessions	bis zu 24	1
Taktrate des Testknotens	625kHz - 10MHz	10MHz
Compiler/Editor integriert	JA	NEIN
Dateiexport "pre-configured"	JA	NEIN
kundenspezifisches System Image	JA	NEIN
Protocol Analyzer	Integriert	Separat
Netzwerkvariablen-Binder	Integriert	Separat

Tabelle 5.1: Vergleich NodeBuilder/LonBuilder

Sollen fertige Knoten im Feld programmiert werden, dann können außer diesen beiden Entwicklungssystemen auch andere Tools eingesetzt werden. Meist wird ein solches Programmiertool Teil eines umfassenden Netzwerk-Tools auf der Basis von LNS sein. Mit der LonWorks Component Architecture (LCA) steht eine Plattform bereit, mit der sich solche Tools sehr leicht erstellen lassen und vor allem mit anderen Tools gleichzeitig im selben Netzwerk betreiben lassen. Echelon bietet hierzu das LCA Field Compiler API an, das als fertige Komponente in ein LNS-Tool integriert werden kann. Ein solcher Field Compiler übersetzt nicht nur Neuron C Source Code, sondern kann auch als

Compiler für beliebige Front-Ends verwendet werden. Auf dem Markt werden heute verschiedene Tools angeboten, die andere Programmiersprachen verwenden, stellvertretend seien hier das IEC-1131-Tool von Gesytec und der STEP5-Interpreter von Weidmüller genannt.

Beim Kauf eines Entwicklungssystems - sei es Node- oder LonBuilder - sollte für die Programmierung von Knoten mit dem Neuron 3120 das zugehörige Programmiergerät mitbestellt werden. Der 3120 muss mindestens mit den Kommunikationsparametern (wenn nicht Direct Connect mit 1,25 MBit/s verwendet wird) programmiert werden, und zwar vor der Platinenbestückung.

Als Ergänzung zu den Programmiertools für den Neuron Chip ist das Microprocessor Interface Program (MIP) Developer's Kit erhältlich. Das MIP ist eine spezielle Firmware für den Neuron Chip zum Betrieb als Kommunikations-Coprozessor für einen Host-Prozessor oder -Rechner. Das Entwicklungskit enthält dazu Libraries und Treiber zur Entwicklung der notwendigen Hostsoftware. Diese liegen im Source Code vor, so dass sie für jedes beliebige System angepasst werden können. Voraussetzung ist ein C-Compiler für das jeweilige System. Motorola liefert Treiber für die HC-11- und die MC683xx-Familien, für andere Prozessoren kann auf ein spezielles Treiber-Entwicklungskit von Gesytec zurückgegriffen werden. Alle bekannten PC-Interfacekarten auf dem Markt basieren auf dem MIP.

Kap.6 Planung, Installation und Betrieb von LonWorks-Netzwerken

Netzwerkdesign und Netzwerkmanagement

Die Entwicklung von LonWorks-Produkten ist nur der erste Schritt auf dem Weg zur eigentlichen Anwendung und zum Betrieb des vollständigen Systems. Netzwerke werden beispielsweise von Systemintegratoren in Betrieb genommen, von Installateuren aufgebaut, von Planern konzipiert, von Behörden ausgeschrieben. Alle in dieser Kette beteiligten Personen benötigen keine Kenntnisse über Knotenentwicklung, Entwicklungssysteme, ja teilweise nicht einmal über die Technologie an sich. Die Installation einer intelligenten Leuchte oder eines intelligenten Ventils ist grundsätzlich nicht von der herkömmlicher Produkte verschieden. Bei der Konzipierung eines intelligenten Netzwerkes sind jedoch einige Überlegungen hinsichtlich Topologie und Strukturierung, aber auch zur Auswahl der richtigen Produkte aus dem immer größer werdenden Angebot notwendig. LonWorks-Netzwerke erleichtern diese Planungsaufgaben aber erheblich gegenüber herkömmlichen Bussystemen, beispielsweise durch ein einheitliches Protokoll, den LonMark-Standard und die Verwendung von Free-Topology-Transceivern,

Von der Idee bis zum Netzwerk

Die einzelnen Schritte von der Idee bis zum Betrieb des Netzes lassen sich grob in fünf Phasen einteilen:

1. Entwicklung oder Einkauf der einzelnen Geräte und Produkte
2. Planung des Netzwerkes
3. Installation
4. Kommissionierung
5. Betrieb

Die Entwicklung der Produkte wurde in den vorangegangenen Kapiteln bereits besprochen. Wichtig für den Einsatz in offenen, interoperablen Netzwerken ist die Einhaltung der LonMark-Richtlinien. So wird von Anfang an sichergestellt, dass sich ein Knoten problemlos und mit seiner vollen Funktionalität in das Netzwerk einbinden lässt. Auch dem Planer gibt dies eine ausreichende Sicherheit bei der Auswahl der einzusetzenden Produkte.

Netzwerkarchitektur und -design

Da ein LonWorks-Netzwerk auf dem Prinzip der dezentralen Intelligenz aufbaut, bei dem Informationen beliebig im Netzwerk verteilt werden können, ist die erreichbare Funktionalität höher als bei herkömmlichen Technologien.

Es kommt bei der Planung im Prinzip darauf an, zu wissen, welche Einzelfunktionen ein bestimmtes Produkt ausführen kann und welche Eingangs- und Ausgangsvariablen zur Verfügung stehen. Zur Erreichung der gewünschten Funktionalität müssen dann diese Variablen in der geeigneten Weise verbunden werden. Dies geschieht entweder noch auf Papier oder man setzt entsprechende Planungstools ein. Wichtig ist, dass Informationen vielfach genutzt und von vielen Knoten gemeinsam verarbeitet werden können. Man muss sich dabei von der Vorstellung der alten Verdrahtungs- und Funktionspläne vollkommen lösen. Der intelligente Kartenleser an der Eingangstüre öffnet eben nicht nur diese selber, sondern gibt seine Information gleichzeitig auch an die Lichtsteuerung, den Aufzug, das Zeiterfassungssystem, die Einzelraumregelung und alle anderen Systeme des richtigen Büros weiter.

Die Gesamtfunktionalität des Netzwerkes ergibt sich also aus den Funktionen der Einzelknoten und damit steht zunächst einmal fest, welche Knoten eingesetzt werden. Die wesentliche Aufgabe besteht nun darin, die richtigen Verbindungen (Bindings) der Netzwerkvariablen zwischen den einzelnen Knoten festzulegen. Bei wenigen Knoten ist dies eine triviale Aufgabe, bei immerwiederkehrenden Verbindungen ebenfalls. Doch bei Tausenden von Knoten ist die Unterstützung eines Tools hilfreich, das verschiedene Ansichten - grafische, Baum- und Listendarstellung - und Plausibilitätsprüfungen bietet. Auch das Kopieren von Knoten, Funktionsgruppen und ihren Verbindungen erspart Zeit und Fehler. Generell sind LonMark-kompatible Produkte aufgrund der genau aufeinander abgestimmten Schnittstellen zu bevorzugen.

Die nächste Aufgabe lautet nun, Architektur und Design des Netzwerkes festzulegen. Die beiden wesentlichen Begriffe hierbei sind Topologie und Segmentierung. Unter der Topologie verstehen wir den Aufbau der physikalischen Verdrahtung. Aus der Historie von Bussystemen und Computernetzwerken sind im wesentlichen zwei Topologien bekannt: Bus-Topologie (auch als Linie bezeichnet) und Ring-Topologie. Bei der Bustopologie wird das Kabel (Twisted Pair, Koaxkabel oder Lichtwellenleiter) ohne Verzweigung an allen Knoten vorbeigeführt und es muss an beiden Enden mit einer bestimmten Impedanz abgeschlossen (terminiert) werden. Es sind entweder gar keine oder nur sehr kurze Abzweigungen (Stichleitungen, Stubs) zu den Geräten zulässig. Die Ringtopologie sieht einen geschlossenen Ring vor, in der Regel ebenfalls mit den genannten Einschränkungen für die Stichleitungen. Eine Ringleitung ist natürlich ausfallsicherer als eine Busleitung, wenn das Medium unterbrochen wird. Weiterhin ist die Stern-Topologie (Star) bekannt. Hierbei wird von einem Zentralknoten ausgehend sternförmig verdrahtet. Sie bietet ebenfalls eine hohe Ausfallsicherheit bei Beschädigung der Verdrahtung, unterliegt aber auch den genannten Einschränkungen hinsichtlich Stichleitungen und Abschluss.

Die Free-Topology-Transceiver von Echelon heben alle diese Einschränkungen auf. Die Geräte werden "frei" verdrahtet, in Bus-, Ring-, Sterntopologie oder

einer beliebigen Kombination hieraus. Anders ausgedrückt, Stichleitungen können beliebig lang sein und es ist nur noch eine einzige Terminierung notwendig. Da weder getrennte Hin- und Rückleitungen zum einzelnen Knoten noch die Einhaltung der Bus- oder Ringtopologie notwendig sind, kann vor allem sehr viel Kabel eingespart werden. Natürlich kann auch mit diesen Transceivern die Physik nicht überlistet werden, es gibt Spezifikationen für maximale Kabellänge und Anzahl der Knoten eines Segmentes. Doch können diese Segmente sehr einfach durch Repeater erweitert werden.

Eine Aufteilung in Segmente ist vor allem bei großen Netzwerken mit einer hohen Zahl von Knoten notwendig. Bei Fabriken und Gebäuden, die bereits heute mehrere Tausend Knoten vernetzen, kann man sich leicht vorstellen, dass die Anzahl der abgesetzten Nachrichten die Kapazität des Mediums (Kanal) schnell übersteigen kann. Aus diesem Grund können Netzwerke nicht nur elektrisch, sondern auch logisch segmentiert werden. Das Nachrichten-aufkommen einer bestimmten Gruppe von Knoten kann in dieser Gruppe lokal gehalten werden. Beispielsweise braucht das Bremssystem eines Zuges nicht mit den Meldungen der Temperatursensoren im Kabinenbereich überschwemmt zu werden. Diese Aufgabe der logischen Filterung übernehmen die Router, zusätzlich zu der Übersetzung von einem Medium zum anderen. Die Entscheidung, ob ein Datenpaket den Router passieren darf oder nicht, trifft dieser aufgrund der Zieladresse des Paketes. Man koppelt also die Gruppen von Knoten, die vorrangig nur untereinander kommunizieren, mit einem Router vom Rest des Netzwerkes ab. Nur Nachrichten, die an einen Knoten außerhalb dieses Segmentes gerichtet sind (und umgekehrt), können ihn dann noch passieren.

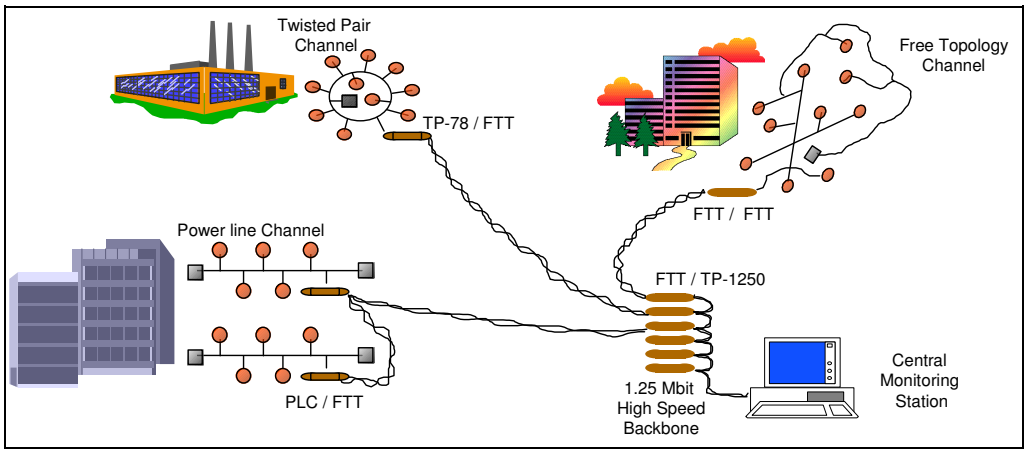

Bild 6.1: Segmentierung eines Netzwerkes

Bei der Auslegung des Netzwerkes sind verschiedene Kennwerte entscheidend: Kanalkapazität, max. Datenaufkommen der einzelnen Knoten, jeweils in Paketen/Sek., Anzahl der Knoten, geforderte Übertragungszeiten, etc. Je nach den Anforderungen des Systems ergibt sich dann das passende Netzwerkdesign. Der Planer kann - muss aber nicht - diese Berechnung selber durchführen. Benutzt er ein Tool hierzu, so kann die gesamte logische

Konfiguration des Netzwerkes bereits erstellt und in einer Datenbank abgespeichert werden. Nach der Installation wird dann einfach die Information aus der Datenbank auf die einzelnen Knoten geladen. Dies kann automatisch und sogar per Telefon oder Internet erfolgen.

In der Praxis haben sich verschiedene Standardarchitekturen bewährt. In der Gebäudeautomation wird zum Beispiel gerne ein Backbone-Segment (Bus oder Ring) verlegt. An diesem werden die Rechnersysteme und Bedieneinheiten angeschlossen, auf denen die wichtigen Meldungen auflaufen. An diesen Backbone werden über Router die Verzweigungen in die einzelnen Stockwerke angeschlossen. Werden auch Powerline-Knoten eingesetzt, so wird von den Rechnern ein FT-Bus an die Unterverteilung (meist im Keller) gezogen und dort über einen Router und einen passiven Phasenkoppler alle drei Phasen gleichzeitig angeschlossen.

Installation und Kommissionierung

Die physikalische Installation des Netzwerkes erfordert keine speziellen Kenntnisse. Die Verdrahtung erfolgt nach den üblichen Plänen, der Netzwerkanschluss ist polaritätsinvariant. Es muss lediglich auf korrekte Terminierung geachtet werden. Ist ein PC in einem Free-Topology-Netzwerk vorhanden, steckt man einfach dort den Jumper auf der Transceiverkarte in die "Free"-Position. Ein solcher Jumper ist auch in vielen Routern und Repeatern bereits eingebaut. Hinweise und Spezifikationen für die zu verwendenden Kabel sind in den Transceiver-Manuals enthalten, diese können über die Echelon-Homepage im Internet bezogen werden.

Nun erfolgt die logische Konfiguration des Netzwerkes, die Kommissionierung. Hierzu gehören die Zuordnung von logischen Adressen für die einzelnen Knoten, das Einrichten der Bindings und die Konfiguration der Knotenapplikationen. Es werden hier grundsätzlich vier verschiedene Verfahren unterschieden (auch als Installationsszenarien bezeichnet):

1. vorkonfigurierte Systeme (preconfigured)
2. Selbstinstallation
3. manuelle (Feld-) Installation
4. automatische Installation

Hinzu kommt die Kombination aus diesen Verfahren.

Vorkonfigurierte Systeme bestehen aus Knoten, denen die logischen Adressen und die Bindings zusammen mit der Applikation bereits bei der Produktion einprogrammiert wurden. Hierzu kann beispielsweise der LonBuilder verwendet werden. Dieses Verfahren kommt zum Einsatz, wenn ein System immer aus denselben Knoten besteht, immer die gleiche Konfiguration

verwendet werden kann und das System nicht erweitert oder angepasst werden muss. Beispiele hierfür sind Maschinensteuerungen, Telefonanlagen, Flugzeuge, kurz gesagt, abgeschlossene Systeme. Nach dem Zusammenbau des Systems kann das Netzwerk sofort seinen Betrieb aufnehmen.

Selbstinstallierende Systeme bestehen aus Knoten, die sich nach dem Aufbau des Netzwerkes eigenständig kommissionieren können. Dies setzt einen entsprechenden Algorithmus in den Knotenapplikationen voraus, der auf der Kenntnis der im System vorkommenden Knotentypen basiert. Es können zwei verschiedene Techniken zur Adressvergabe angewendet werden. Zum einen kann der Anwender eine Knotenadresse vorgeben, beispielsweise über Codierschalter oder ein menügeführtes Display mit Tastenfeld. Ein Knoten kann aber auch das Netzwerk abscannen und sich dann selber eine noch freie Adresse zuordnen. Die Mechanismen hierzu sind bereits im Protokoll, d.h., in der Firmware enthalten. Dieses Verfahren kommt allerdings ebenfalls nur für kleine und abgeschlossene Systeme in Frage. Die Programmierung entsprechender Algorithmen ist zudem nicht trivial und setzt besondere Kenntnisse der Netzwerkmanagementfunktionen voraus. Vorteil der beiden genannten Verfahren ist, dass der Anwender nicht erst mit einem Tool das Netzwerk einrichten muss.

Die manuelle Kommissionierung mit Hilfe eines Tools ist das übliche Verfahren, wenn ein spezifisch geplantes Netzwerk, das meist aus Standardkomponenten verschiedener Hersteller besteht, eingerichtet werden soll. Aber auch bei Erweiterungen und Änderungen eines Netzwerkes wird ein Tool eingesetzt. Unter Tool verstehen wir dabei sowohl Applikationen, die auf einem PC laufen als auch Hand-Terminals, die natürlich weit weniger komfortabel sind. Üblicherweise nehmen diese Tools die Adressvergabe eigenständig vor, nur wenige erlauben zusätzlich eine Anwender-Eingabe. Der Leistungsumfang ist nicht ganz so unterschiedlich wie das jeweilige Mensch-Maschine-Interface. Manche Tools arbeiten rein textbasiert mit Listen, in denen Knotennamen, Netzwerkvariablen usw. angezeigt werden. Andere haben grafische Oberflächen, auf denen die Knoten in Baumstrukturen oder durch kleine Symbole angezeigt werden. Es gibt sogar Tools, die verschiedene Ansichten erlauben oder deren Oberfläche kundenspezifisch konfiguriert werden kann. Generell gilt diese Vielfalt als besonderer Vorteil, da die Anwender aus völlig verschiedenen Industrien und Berufsumfeldern kommen und jeder seine eigene Fachsprache in den Tools wiederfinden möchte. Wichtig bei der Auswahl eines Tools ist, dass es nicht nur den eigenen Vorstellungen entspricht, sondern auch interoperabel ist. Dies bedeutet, dass ein bereits bestehendes Netzwerk erkannt wird und die Datenbank kompatibel zu der von anderen Tools ist. Alle Tools, die auf der Basis der LonWorks Network Services (LNS) programmiert wurden, erfüllen die letzte, wichtigste Voraussetzung.

Die Grenzen zum vierten Szenario, der automatischen Kommissionierung, sind schwimmend. Automatische Kommissionierung bedeutet, dass zum Einrichten des Netzwerkes keine Aktion seitens des Anwenders oder Installateurs mehr notwendig ist. Ein spezieller Netzwerkmanagementknoten übernimmt diese Aufgabe. Dies kann einer der ohnehin im System vorhandenen Knoten sein (Echelon bietet hierzu ein spezielles Modul, das NSS-10 an), dies kann aber auch eines der vorgenannten Tools sein. Wenn beispielsweise der Planer oder Systemintegrator die Kommissionierung "off-net" mit diesem Tool (oder einem geeigneten anderen LNS-Tool) vorgenommen hat, wird diese Konfiguration ohne weitere Interaktion auf die Knoten geladen.

Auch die Kombination der genannten Szenarien ist möglich. Es bietet sich manchmal an, vorkonfigurierte oder selbstinstallierende Knoten zu produzieren, die in vielen Fällen gleich eingesetzt werden können (Plug and Play). Sollen diese Knoten nun in ein größeres Netzwerk integriert werden, so kann diese Konfiguration von einem Tool nachträglich geändert werden. Der LonMark-Standard sieht hierfür eine spezielle Konfigurationsvariable vor.

So vielfältig wie die Installationsszenarien sind auch die Hilfsmittel, mit denen die Technologie den Installateur oder Anwender bei der Kommissionierung unterstützt. Jeder Neuron Chip hat zunächst einmal eine eindeutige 48-Bit Hardware-Adresse, Neuron ID genannt. Die meisten Installationsalgorithmen fragen zunächst einmal diese ID ab, danach werden alle weiteren Netzwerk-management-Kommandos an diese ID adressiert und dem Neuron Chip auf diesem Wege seine logische Domain-, Subnet- und Nodeadresse mitgeteilt. Dieses Verfahren wird auch von den Ethernet-Karten bei Computern verwendet. Der Neuron Chip kann diese ID als Broadcast versenden, wenn man einen bestimmten Pin auf Nullpotential legt. Daher sind die meisten Produkte auch mit einem Taster für diesen Service Pin ausgestattet. Die Neuron ID kann aber auch über das Netzwerk abgefragt werden, neue Knoten somit automatisch erkannt werden. Außerdem gibt es einen Systembefehl, mit dem man das Senden der ID aus der Applikation heraus vornehmen kann, in der Reset Task beispielsweise. Weiterhin steht Speicherplatz zur Verfügung, in dem ASCII-Text zur Selbstidentifikation und -dokumentation abgelegt werden kann. Hierzu gehören die Program ID, die Auskunft über das geladene Programm gibt, sowie Text-Strings zur Beschreibung von Knoten und Netzwerkvariablen. Zusätzlich steht noch der sog. Location String zur Verfügung, der nicht Bestandteil der Applikation ist, sondern von einem Netzwerkmanagement-Tool verwaltet werden kann. Alle diese Texte können über das Netzwerk ausgelesen werden. Ein typisches Installationsszenario besteht in der Vorab-Definition des Netzes mit einem Tool, nach der Installation der Knoten braucht man nur noch die Service Pins zu betätigen und die Knoten werden automatisch vom Tool konfiguriert.

Besteht keine Möglichkeit, den Service Pin zu drücken, etwa, weil ein Gerät nicht zugänglich ist, so kann von dem "wink"-Kommando Gebrauch gemacht

werden. Ein Knoten wird mit der Neuron-ID adressiert, das wink-Kommando an ihn gesendet und eine bestimmte Aktion wird in dessen wink-Task ausgelöst. Meist wird eine LED oder ein Summer angesteuert, um visuell die richtige Zuordnung zwischen Datenbankeintrag und tatsächlichem Knoten herauszufinden. Sinnvollerweise wird das wink-Kommando zusammen mit dem query-Befehl verwendet, der automatisch alle neuen Knoten im Netzwerk findet.

Betrieb des Netzwerkes

Es gibt Netzwerke, die ihren Dienst innerhalb eines Systems ohne jede weitere Beachtung oder Interaktion verrichten. Neuron Chips als Diagnoseprozessoren in Supercomputern, als Kommunikationsprozessoren in Telefonvermittlungsanlagen oder Steuerungsnetzwerke in der Industrieautomation gehören hierzu ebenso wie Lichtrufsysteme in Krankenhäusern. Andere Netzwerke sind interaktiv bedienbar oder geben ihre Daten auf einer Visualisierungsstation aus. Dies sind zunehmend PC als universelle Plattform, auf denen eine entsprechende Applikation läuft. Eine Applikation, die Teil eines LonWorks-Netzwerkes ist, kann man grundsätzlich vollständig selber programmieren. Eine anspruchsvolle Applikation direkt auf den Treibern der Interfacekarten aufsetzend zu programmieren, ist allerdings eine mühsame, zeitraubende, fehlerträchtige und teure Beschäftigung. Einfacher ist der Einsatz eines kommerziellen Programms, wie beispielsweise Wonderware's InTouch. Hierzu wird der DDE-Server von Echelon eingesetzt, so dass keinerlei eigene Programmierung notwendig ist. Ebenfalls relativ einfach ist die Programmierung eines DDE-Clients, z. B. mit Visual Basic. Universeller ist jedoch die Programmierung mit Hilfe eines der beiden LNS Developer's Kits. Vor allem LNS for Windows zeichnet sich aus durch einfache Programmierung, höhere Performance und Flexibilität durch Verwendung von ActiveX (früher OLE). Größter Vorteil ist jedoch, dass alle Funktionen zum Netzwerkmanagement bereits in LNS enthalten sind. So sind eigene LNS-Applikationen nicht nur zu den Installationstools kompatibel, sondern können diese Funktion gleich mit übernehmen. So kann dasselbe Tool zur Planung, Simulation, Kommissionierung des Netzes, Konfiguration der Knoten, Visualisierung der Daten, Wartung und Diagnose, Reparatur und Änderung sowie zur interaktiven Kommunikation mit den Knoten verwendet werden.