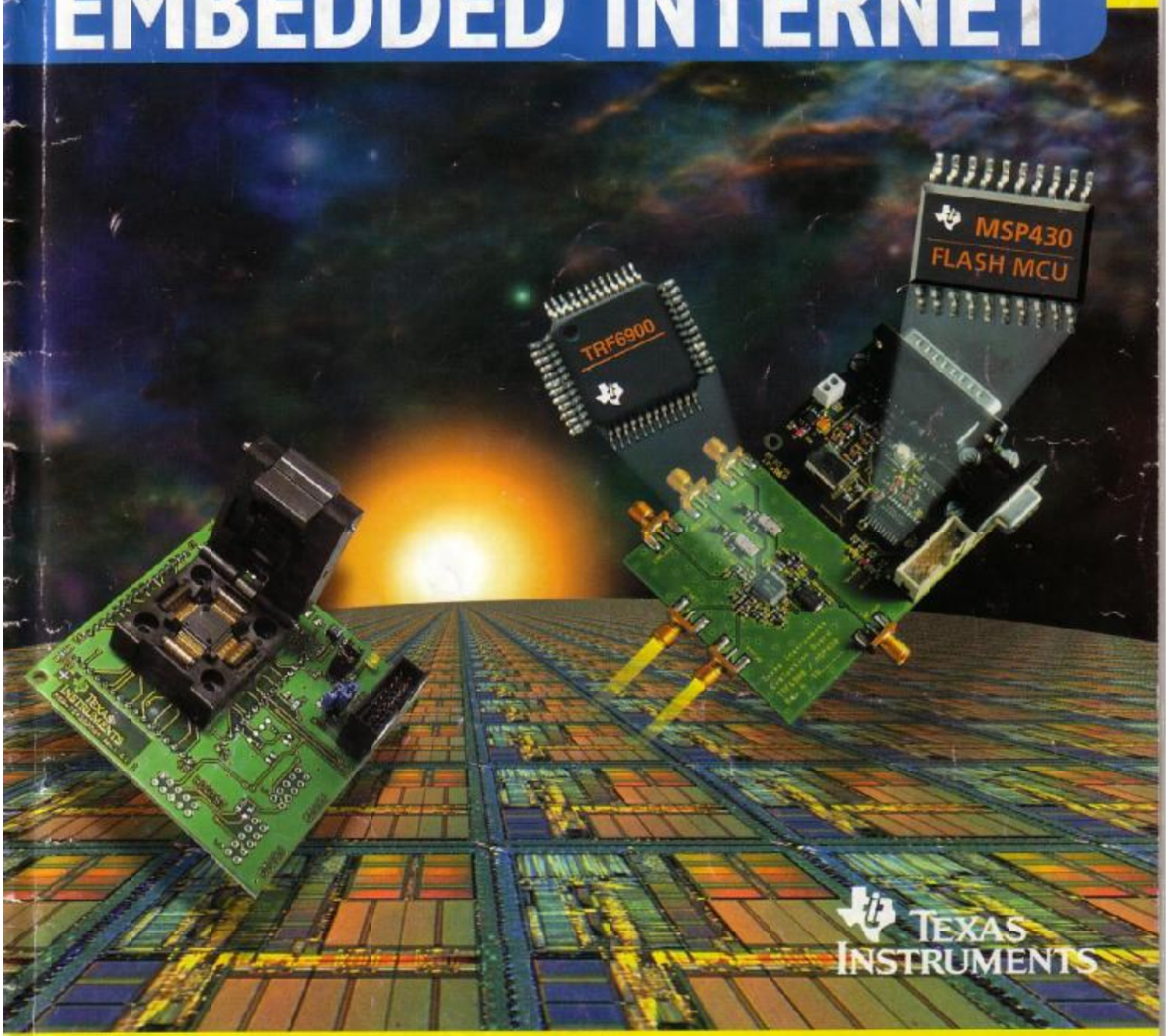
EXTRA AUSGABE · SEPTEMBER [illegible] · DM 29,80 · € 15,24

# DESIGN&ELEKTRONIK EXTRA

PRODUKTE UND KNOW-HOW F[illegible] DEN ELEKTRONIK-ENTWICKLER

www.elektroniknet.de

# EMBEDDED INTERNET

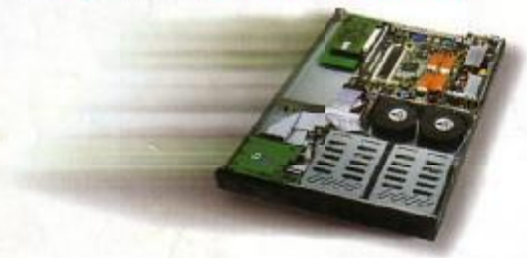



## Eine µC-Webserver-Lösung im Detail:
## Schaltpläne, Lay-outs, Software, TCP/IP-Stack

# DSM Infinity® 3100

## Der **coole** High Speed Flatrack Industrie **Server**

freecall **0800-INFINITY**



Die DSM Infinity Serie bietet multiple Einsatzmöglichkeiten durch eine kompakte Bauform, bei der besonders auf eine effiziente Kühlung geachtet wurde. Die Lösung für ISPs, ASPs, Renderfarmen, High End Workstations u.v.m.

*Cool Performance by* **INFINITY®**

- CoolWorks ® Kühltechnologie
- Dual Pentium ® III, 1GHz +
- Dual Fast Ethernet LAN
- Dual Hot-Swap SCSI Harddisk Drives (SCA-2)
- Skalierbar bis 84 Prozessoren/Rack

**Hotline: 089/15798-250**

**www.Cool-Infinity.de**

intel inside

pentium® III

**DSM Computer AG**
Am Loferfelc 54
81249 München
Tel.: 089/15 798-250
Fax:089/15 798-196
www.dsm-computer.de

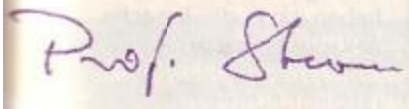
DSM® Computer

# EXTRA Heft

Liebe Leser, dieses EXTRA-Heft »Embedded Internet« der DESIGN&ELEKTRONIK ist in mehrfacher Hinsicht etwas besonderes. Doch der Reihe nach:
Das Internet und seine Möglichkeiten faszinieren nicht nur die Anwender, sondern auch die Ingenieure und Entwickler. Zunehmend sind kleine Netzknoten von großem Interesse. Sie erlauben den extrem kostengünstigen und aufwandsminimierten Anschluss ans Internet – ein Wunsch zahlreicher Anwender.
Im Frühjahr diesen Jahres starteten wir an der Hochschule für Technik, Wirtschaft und Kultur Leipzig (HTWK) das Projekt »easyWeb«, dessen Ergebnisse Sie in diesem Heft finden. In Herrn Dannenberg fand ich einen außerordentlich engagierten Studenten unserer Hochschule, der sich nach intensiven Vorgesprächen bereit fand, sich dieser Problematik zu stellen. Abweichend von Entwicklungen in der Industrie verfolgten wir von Anfang an das Ziel, die gesamten Ergebnisse des Projektes offen zu legen. Ein modernes Schaltungskonzept, Verfügbarkeit der Hardware, aber auch eine einfache Portierbarkeit der Software waren dabei wichtig. Vor allem aber einer umfassenden und erschöpfenden Dokumentation schenkten wir große Beachtung.
**Die Ergebnisse des Projektes liegen Ihnen, verehrte Leser, in diesem EXTRA-Heft komplett vor:**

- **Eine Beschreibung der Realisierung eines Mikrocontroller-basierten Internetknotens auf der Basis des MSP430F149 von Texas Instruments.**
- **Im Mikrocontroller ist ein TCP-Protokoll implementiert.**
- **Als physikalisches Übertragungsmedium wurde das Ethernet gewählt.**
- **Vorstellung einer leicht zu verstehenden, einfach zu nutzenden und effizienten Plattform für viele praktische Einsatzfälle, deren Funktionalität durch ein »Application Program Interface« (API) gekapselt ist.**
- **Als Applikation wird ein Webserver beschrieben, der eine dynamische Internet-Seite bereitstellt.**
- **Die Programmierung der Software in C und die ausführliche Dokumentation unterstützen eine einfache Portierung auf andere Controllersysteme.**

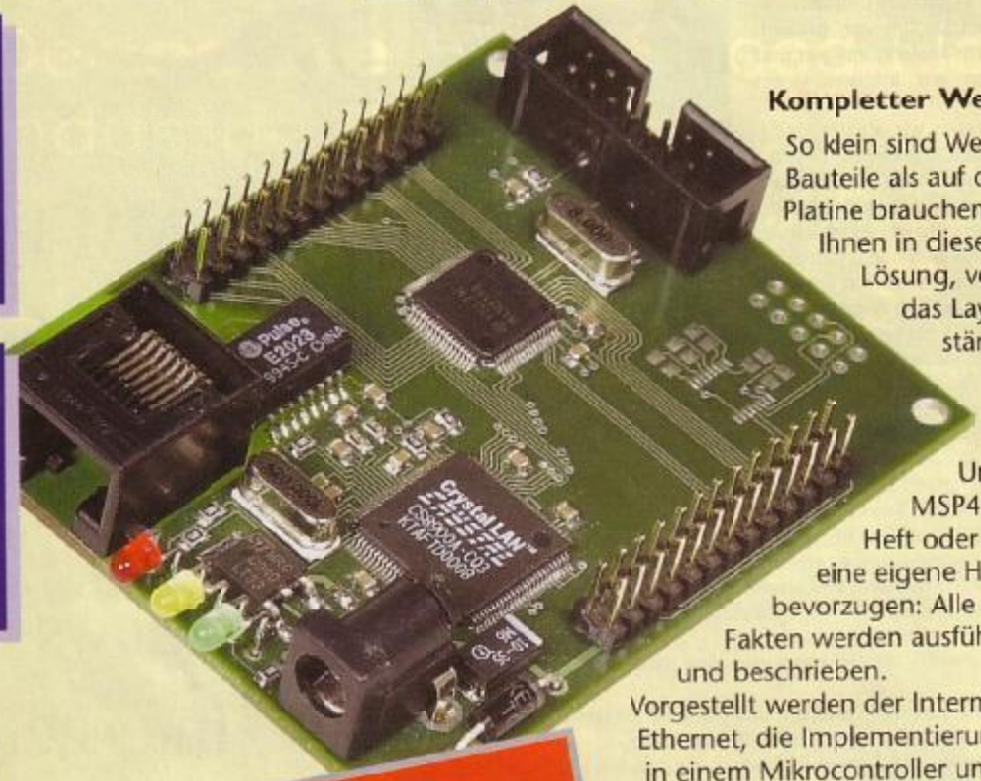
An dieser Stelle möchte ich mich bei den Firmen bedanken, die uns bei der Projektrealisierung tatkräftig unterstützt und dieses Extra-Heft möglich gemacht haben. Besonders zu nennen sind Texas Instruments und Cirrus Logic, die uns Bausteine überlassen haben, sowie IAR Systems, bei denen wir uns ein C-Entwicklungssystem für den MSP430 ausleihen durften.
Anwendungen wie Webserver zur Fernüberwachung und Fernsteuerung von Sensoren bzw. Aktoren über eine universelle Schnittstelle (Webbrowser), automatisches Versenden von E-Mails beim Eintreten von Triggerereignissen (Überwachung, Alarm) oder das Übertragen von Prozessgrößen innerhalb von Automatisierungsanlagen sollen nur einige Beispiele sein, die sich aus dem hier vorgestellten Netzknoten entwickeln lassen.
Schließlich möchte ich noch auf eine Frage eingehen, die mir in Vorbereitung auf dieses Heft des öfteren gestellt wurde: »Wieso ‚verratet' ihr im EXTRA-Heft der DESIGN&ELEKTRONIK bis ins Detail, wie alles geht ?«
Die DESIGN&ELEKTRONIK ist seit jeher bemüht, Entwickler zu unterstützen und Hilfestellung bei der Bewältigung der täglichen Aufgaben zu geben. Die monatlichen Themenhefte und die erfolgreichen Entwicklertoren bis hin zum Kongress »Embedded Intelligence« verdeutlichen dies eindrucksvoll. Mit diesem Heft möchte die DESIGN&ELEKTRONIK einer neuen Technologie breite Anwendung verschaffen.
**Mit dem EXTRA-Heft können Sie sich in die Problematik ‚Internetknoten' einarbeiten, das Projekt nachvollziehen oder auch nachbauen bzw. die Idee auf Ihre Plattform portieren. Sie finden alles im Heft und die Quellen im Internet unter den angegebenen Adressen.**
**Wer noch weitere Anregungen, Lösungen und Tipps sucht, den möchte ich zum Entwicklerforum »Embedded Internet« am 11.10.2001 nach München einladen. Diese Veranstaltung, deren Programm Sie auf Seite 22 finden, besticht durch ihre Kompetenz und wird umfassende Antworten bereit halten.**
Und ganz unter uns, wir vom Team der DESIGN&ELEKTRONIK sind stolz auf die Ergebnisse unserer Arbeit – und ganz speziell auf dieses EXTRA-Heft.
Nachzutragen bleibt, dass Herr Dannenberg seine Diplomarbeit mit der Note »sehr gut« verteidigt hat und inzwischen bei Texas Instruments in Freising tätig ist. Ihm gilt mein besonderer Dank für die Arbeit an diesem Heft.



Prof. Dr.-Ing. Matthias Sturm, Herausgeber
**ProfSturm@design-elektronik.de**

INHALT

# Embedded Internet



**Kompletter Webserver auf 45 cm²**

So klein sind Webserver heute: Mehr Bauteile als auf der nebenstehenden Platine brauchen Sie nicht. Wir zeigen Ihnen in diesem Heft die komplette Lösung, von der Schaltung über das Layout bis hin zum vollständigen Quellcode in C – uneingeschränkt nutzbar für Ihre Applikation. Unabhängig, ob Sie die MSP430-Lösung in diesem Heft oder eine Portierung auf eine eigene Hardware-Plattform bevorzugen: Alle Hintergründe und Fakten werden ausführlich dargelegt und beschrieben.

Vorgestellt werden der Internetzugriff über Ethernet, die Implementierung eines TCP/IP-Stacks in einem Mikrocontroller und darauf aufsetzend eine Webserver-Anwendung mit dynamischer Internetseite.

**Entwicklerforum »Embedded Internet«**

Über die Heftinhalte hinausgehend bietet das Entwicklerforum «Embedded Internet« am 11.10.2001 im Münchner M,O,C, weitere Lösungen und zusätzliche Informationen. Das vollständige Programm finden Sie auf Seite 22

## Die Autoren

**Andreas Dannenberg**

Andreas Dannenberg ist Jahrgang 1978. Nach dem Abitur entschied sich Herr Dannenberg für ein Studium der Nachrichtentechnik an der Hochschule für Technik, Wirtschaft und Kultur (HTWK) in Leipzig. Im Rahmen seiner Diplomarbeit entwickelte er den in diesem Heft vorgestellten Embedded-Webserver auf Basis eines MSP430-Mikrocontrollers. Seit kurzem ist er als Field Application Engineer bei Texas Instruments tätig.

**Prof. Dr.-Ing. Matthias Sturm**

Prof. Sturm lehrt an der Hochschule für Technik, Wirtschaft und Kultur in Leipzig und dürfte den meisten Lesern als Herausgeber der DESIGN&ELEKTRONIK bekannt sein. Seit 1995 wirkt er aktiv an verschiedenen Foren der DESIGN& ELEKTRONIK mit. Seine Kurse und zahlreiche Beiträge in Fachzeitschriften haben ihn in der Branche bekannt gemacht.

## INTERNET-PROTOKOLLE



## EMBEDDED WEBSERVER



## APPLIKATIONS-SCHNITTSTELLE



## HTTP-SERVER



## ANHANG



## LESERSERVICE



**Schaltung, Layout & Software im Web**
Alle Quelltexte, Schaltungs- und Layout-Unterlagen stehen zum kostenfreien Download für Sie bereit unter:
**www.design-elektronik.de/extraheft**

*Aufgelöst in Bits und Bytes*

# Internet-Protokolle – Die Grundlagen

**Um einen Internet-Knoten tatsächlich realisieren zu können, bedarf es zunächst einigen Grundwissens über die Datenübertragungsprotokolle. So sollte der Entwickler nicht nur den Aufbau und die Elemente des ISO/OSI-Schichtenmodells kennen, oder Begriffe wie Ethernet, ARP, IP, ICMP, TCP oder HTTP auseinanderhalten können. Er muss vielmehr bis ins Detail wissen, wie z.B. eine TCP-Verbindung aufgebaut wird, welche Flags dabei wie gesetzt werden müssen, oder aus welchen unterschiedlichen Elementen ein HTTP-Request bzw. ein HTTP-Response besteht. Der folgende Beitrag zeigt im Detail, wie's geht.**

Es ist üblich, Kommunikationsprotokolle schichtweise aufzubauen und zu beschreiben. Dadurch ergibt sich eine Zergliederung unseres Hauptanliegens, nämlich »Wie erreiche ich eine Kommunikation zwischen zwei oder mehreren Geräten mit bestimmten Ansprüchen an Zuverlässigkeit und Geschwindigkeit?«, in mehrere Teilaufgaben. Diese Schichtung von Protokollen mit ihren zugehörigen Diensten nennt man »Protokoll-Stapel« (Stacks), ihr Aufbau wird mit Hilfe von Referenzmodellen beschrieben. Jede dieser Schichten enthält ihre eigenen Funktionen, verwendet aber zur Bereitstellung ihres Dienstes auch die Funktionen der darunter liegenden Schicht.

Ein geschichteter Protokollstapel bringt Vorteile bezüglich Änderungs- und Erweiterungsmöglichkeiten. Das für das Internet geltende, aus vier Schichten bestehende Modell ist eine Teilmenge des ISO/OSI-7-Schichtenmodells (International Organisation for Standardization, Open System Interconnection). Dies resultiert aus der Tatsache, dass das Internet vor dem ISO/OSI-Referenzmodell entstanden ist. In Bild 1 sind die beiden Referenzmodelle gegenübergestellt. Die Funktion der einzelnen Schichten lässt sich Tabelle 1 entnehmen [CNetw].
Von der Anwendung ausgehend, welche Daten übertragen möchte (z.B. ein Webserver), wird durch jede Schicht des Protokollstapels ein entsprechender Protokollkopf (Header) mit Kontrollinformationen vorangestellt (Bild 2 auf Seite 8). Dieser Vorgang wird Datenkapselung genannt (encapsulation). Beim Empfanger eines solchen Frames, z.B. aus dem Ethernet, sind vom TCP/IP-Protocol-Stack dann von den einzelnen Softwareschichten die entsprechenden Header auszuwerten und zu entfernen, um letztlich die eigentlichen Nutzdaten zu gewinnen und an die entsprechende Applikation (z.B. Internet-Browser) weiterzugeben [CNetw].
Im Folgenden werden die einzelnen für dieses Projekt wichtigen Protokolle vorgestellt. Da die Original-Spezifikationen teilweise mehrere hundert Seiten lang sind, sollte klar sein, dass diese Informationen nur einen Überblick darstellen können. Weiterhin werden Grundkenntnisse aus dem Bereich der Netzwerktechnik vorausgesetzt. Für eine detailliertere Beschreibung wird auf die entsprechenden Quellen verwiesen.

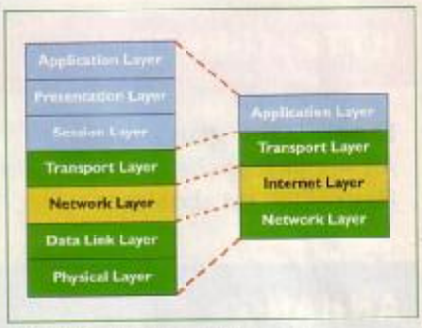



**Bild 1 ISO/OSI und Internet-Referenzmodell**

| Schichtname | Funktion | Beispiele |
|---|---|---|
| Application Layer | Enthält eine Vielzahl von Protokollen, die Anwendungen zur Erbringung ihrer Dienste definiert haben | HTTP, Telnet, FTP, eMail (SMTP, POP) |
| Transport Layer | Ermöglicht Kommunikation zwischen Quell- und Zielhost (Kommunikation zweier Endpunkte) | Transmission Control Protocol (TCP), User Datagram Protocol (UDP) |
| Internet Layer | Zustellung und Routing von Datagrammen zwischen den Internet-Netzknoten | Internet Protocol (IP), Internet Control Message Protocol (ICMP), Address Resolution Protocol (ARP) |
| Network Layer | Hostspezifische Implementierung der Übertragung von Datagrammen | Ethernet (IEEE 802.3), Point-To-Point Protocol (PPP), AX.25 |

**Tabelle 1: Schichten des Internet**

## Ethernet

Ethernet ist ein herstellerneutrales und sehr weit verbreitetes Medium, um im lokalen Netzwerk (Local Area Network, LAN) Datenpakete zu übertragen. Es stellt im Internet-Referenzmodell ein Protokoll der Netzwerkschicht dar (network layer). Der Standard IEEE 802.3 (Institute for Electric and Electronic Engineers) definiert mögliche Übertragungsmedien, die physikalische Umsetzung der Informationen und die eigentliche Datenübertragung mit Geschwindigkeiten von 10 MBit/s bzw. 100 MBit/s (Fast Ethernet). Ethernet basiert auf der gemeinsamen Nutzung eines Kanals, auf den alle angeschlossenen Netzwerkteilnehmer Zugriff haben. Dazu wird ein statistisches Kanalzugriffsverfahren benutzt (CSMA/CD – Carrier Sense Multiple Access with Collision Detection). Die Übertragung der Bits erfolgt Manchester-codiert entweder über differentielle Zweidrahtleitungen (10Base-T, Twisted Pair, RJ 45) oder über eine Koaxialleitung (10Base-2, BNC).

Jedem Netzwerkknoten ist eine eindeutige, 48 Bit lange physikalische Hardwareadresse, die »MAC-Adresse« (Media Access Control), zugeordnet. Die Länge eines mittels Ethernet übertragbaren Frames liegt zwischen 64 Byte und 1518 Byte. Diese Größenangabe bezieht sich auf den gesamten Rahmeninhalt, mit Ausnahme der Präambel. Zur Verdeutlichung ist in Bild 3 ein Rahmen beschrieben [IEEE802.3] [Crystal].

Die Präambel besteht aus identischen Bytes (Binär: 10101010) und dient der Synchronisation des eingehenden Rahmens mit dem Zeitgeber des Empfängers. Direkt im Anschluss daran folgt der eigentliche Frame. Die ersten sechs Byte (48 Bit) bilden die Zieladresse im LAN (auch als DA = Destination Address bezeichnet). Ein wichtiger Sonderfall ist die Belegung der Zieladresse mit »FF-FF-FF-FF-FF-FF«. Frames mit diesem Ziel werden als »Broadcast-Frames« (Rundsende-Frames) bezeichnet und von allen angeschlossenen Netzwerkknoten angenommen. Broadcast-Frames wer-

### Eine kurze Geschichte des Internet

Die Wurzeln des Internet gehen in die 60er-Jahre der USA zurück. Zu dieser Zeit forderte das US-Verteidigungsministerium (Department of Defense, DoD) eine in hohem Maße ausfallsichere Netzwerktechnologie, um beispielsweise nach einem möglichen Nuklearschlag noch einsatzfähig zu sein. Dazu wurde die »Advanced Research Projects Agency« (ARPA) ins Leben gerufen und mit der Entwicklung einer zuverlässigen, paketvermittelnden Netzwerktechnologie beauftragt.

Im Jahre 1969 entstand das erste experimentelle Netz, das »ARPANET«, bestehend aus vier Knoten zwischen vier Universitäten. Als Übertragungsmedium dienten dem Urahn aller Netzwerke Telefonleitungen mit einer Geschwindigkeit von 50 KBit/s. In der folgenden Jahren wurden immer mehr Knoten angeschlossen. Die ersten bedeutenden Anwendungen des ARPANET, welche auch zu dessen rasantem Wachstum in den weiteren Jahren beitragen sollten, waren die e-Mail-Programme »SNDMSG« und »READMAIL« sowie das von Jon Postel entwickelte Datei-Übertragungsprotokoll »FTP« (File Transfer Protocol). Das Bedürfnis nach schnellerer Kommunikation führte im Jahr 1973 zur Entwicklung von Ethernet. Dessen Name ist eine ironische Anspielung auf den früheren Glauben an ein allgegenwärtiges Transportmedium für elektromagnetische Wellen, den »Äther«. Hier kamen für die Datenübertragung zunächst Koaxialleitungen zum Einsatz.

Der Wunsch nach einer verlässlichen Kommunikation in sehr inhomogenen und fehlerbehafteten Netzstrukturen führte 1974 zur Entwicklung des »TCP/IP«-Protokolls (Transmission Control Protocol/Internet Protocol) in seinen Grundzügen. Dieses hat sich seitdem zu dem Standardprotokoll für Netzwerke schlechthin entwickelt. Seit 1983 erfolgte im ARPANET die Nutzung des TCP/IP-Protokolls, welches sich auch vorzüglich über das paketorientierte Ethernet transportieren ließ. Weiterhin erfolgte die Abspaltung eines militärischen Teilnetzes, des »MILNET«, und die Entwicklung des DNS-Systems (Domain Name Service, löst Hostnamen zu IP-Adressen auf) zur Vereinfachung der Kommunikation. Bis zu diesem Zeitpunkt umfasste das Internet etwa 1000 Knoten. Der rasante Wachstumsschub der Neuzeit ist hauptsächlich auf die Entwicklung des »World Wide Web« (Darstellung multimedialer Inhalte mittels Seitenbeschreibungssprache, 1990) und der ersten Version einer Internet-Browser-Software (»Mosaic«-Browser, 1993) zurückzuführen. Die Sammlung von Netzen, welche das ARPANET ursprünglich darstellte, wurde im Laufe der Zeit zu einem weltumspannenden Netzverbund, welcher heute allgemein unter dem Namen Internet bekannt ist [NERDS].

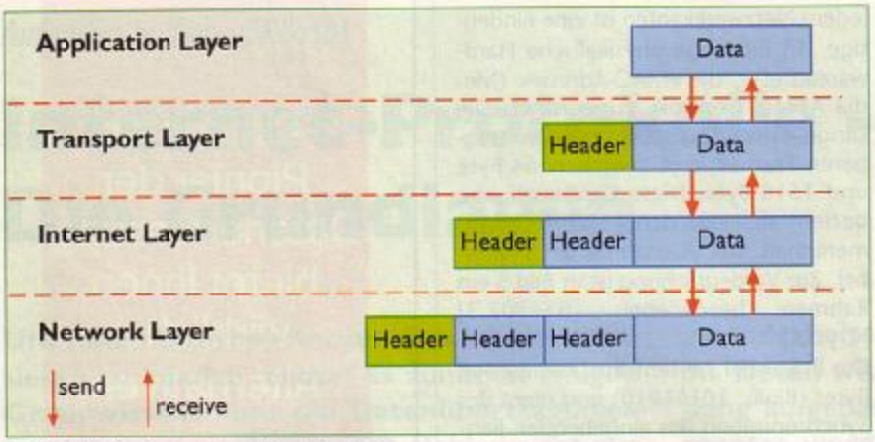



**Bild 2: Datenkapselung im Internet**

den verwendet, wenn der Absender die Empfänger-MAC-Adresse eines Knotens nicht kennt. Die nächsten sechs Byte enthalten die Quelladresse des Frame-sendenden LAN-Teilnehmers (auch als SA = Source Address bezeichnet). Die folgenden zwei Byte geben den Typ der Frames an. Diese Ethernet-Typnummern sind eindeutig und in der IEEE-Norm festgelegt. Sie dienen zur Unterscheidung, an welche höheren Protokolle der Rahmen weiterzuleiten ist. Die zwei für einen TCP/IP-Stack wichtigsten Typnummern sind in Tabelle 2 aufgeführt [TCP/IP].

| Typ | Protokoll |
|---|---|
| 0x0800 | IP |
| 0x0806 | ARP |

**Tabelle 2: Ethernet-Typ-Feld**

Der Abschnitt »LLC data« enthält im Anschluss daran die zu übertragenden Nutzdaten (pay-load) der höheren Protokollschichten. Am Ende eines Ethernet-Rahmens wird eine vier Byte lange CRC-Prüfsumme (Cyclic Redundancy Check übertragen, um den Rahmen auf Fehler oder Beschädigungen prüfen zu können.

Die maximale Anzahl der in einem Datenpaket übertragbarer Nutzdaten-Bytes ergibt demnach sich aus 1518 - 6 - 6 - 2 - 4 = 1500. Da ein Ethernet-Frame nicht kürzer als 64 Byte sein darf, werden eventuell fehlende Bytes durch »Padding« mit beliebigen Bytes aufgefüllt. In Beispiel 1 sind die ersten 16 Byte eines Ethernet-Rahmens dargestellt und beschrieben. Dieser und alle folgenden Frames wurden mit Hilfe des Microsoft Netzwerkmonitors »Netmon« gesammelt. Das Programm ist Bestandteil diverser Server-Versionen von Windows 2000.

## Address Resolution Protocol (ARP)

Wie im vorangegangenen Abschnitt gezeigt, enthält ein Ethernet-Frame sowohl die Daten höherer Protokollebenen wie auch die des Internet-Protokolls. Das Internet-Protokoll verwendet aber im Gegensatz zum Ethernet vier Byte lange, logische Anschlussnummern zur Identifikation von Kommunikations-Endpunkten (IP-Adressen). Diese müssen vor einer Datenübertragung via Ethernet in physikalische Adressen übersetzt werden. Um nun ein IP-Datagramm im LAN übertragen zu können, muss der Absender des Rahmens zunächst die Hardware-MAC-Adresse des Empfängers kennen. Diese Adresse herauszufinden, ist Aufgabe des »Address Resolution Protocols« (ARP). Es ist damit im Referenzmodell der Internet-Schicht zuzuordnen (internet layer).

Die Funktionsweise von ARP lässt sich am Besten an einem Beispiel erklären: Der Internet-Knoten mit der IP-Adresse »192.168.0.200« möchte die MAC-Adresse des Knotens mit der IP »192.168.0.30« herausfinden. Dazu sendet er ein Broadcast-Frame (Zieladresse FF-FF-FF-FF-FF-FF) in das LAN, welches einen »ARP-Request« enthält. Das ist eine Aufforderung an den LAN-Teilnehmer mit der IP »192.168.0.30«, seine MAC-Adresse dem Absender des Requests mitzuteilen. Dieses Frame wird von allen Netzwerkknoten aufgenommen und ausgewertet. In Beispiel 2 ist ein solcher ARP-Request dargestellt und beschrieben. Nach dem Empfangen des

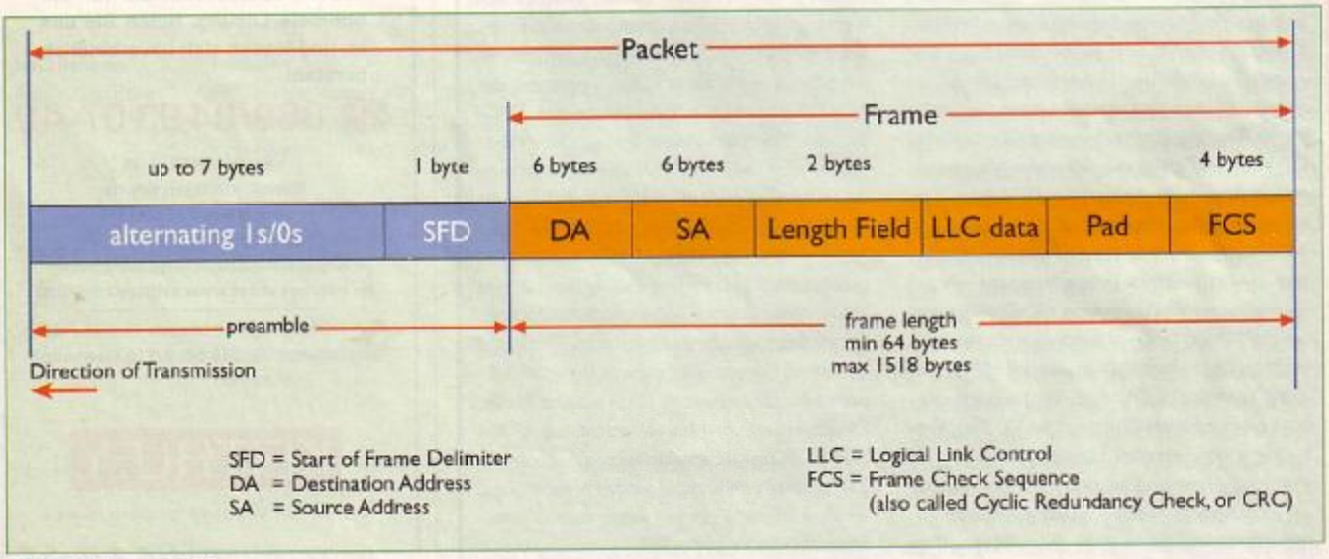



**Bild 3: Ethernet-Rahmen**

```
00000:  FF FF FF FF FF FF C0 01 02 B7 88 44 08 06 00 01   ÿÿÿÿÿÿ....ˆD....

ETHERNET: Destination address : FFFFFFFFFFFF
ETHERNET: Source address : 000102B78844
ETHERNET: Ethernet Type : Cx0806 (ARP:  Address Resolution Protocol)
```

**Beispiel 1: Ethernet Protkollkopf**

```
00000:  FF FF FF FF FF FF 00 01 02 B7 88 44 08 06 00 01   ÿÿÿÿÿÿ....ˆD....
00010:  08 00 06 04 00 01 00 01 02 B7 88 44 C0 A8 00 C8   ..........ˆDÀ¨.È
00020:  00 00 00 00 00 00 C0 A8 00 1E                     ......À¨..

ARP_RARP: Hardware Type = Ethernet (10Mb)
ARP_RARP: Protocol Type = 2048 (0x800)
ARP_RARP: Hardware Address Length = 6 (0x6)
ARP_RARP: Protocol Address Length = 4 (0x4)
ARP_RARP: Opcode = Request
ARP_RARP: Sender's Hardware Address = 000102B78844
ARP_RARP: Sender's Protocol Address = 192.168.0.200
ARP_RARP: Target's Hardware Address = 000000000000
ARP_RARP: Target's Protocol Address = 192.168.0.30
```

**Beispiel 2: ARP-Request**

```
00000:  00 01 02 B7 88 44 01 02 03 04 05 06 08 06 00 01   ....ˆD..........
00010:  08 00 06 04 00 02 01 02 03 04 05 06 C0 A8 00 1E   ............À¨..
00020:  00 01 02 B7 88 44 C0 A8 00 C8

ARP_RARP: Hardware Type = Ethernet (10Mb)
ARP_RARP: Protocol Type = 2048 (0x800)
ARP_RARP: Hardware Address Length = 6 (0x6)
ARP_RARP: Protocol Address Length = 4 (0x4)
ARP_RARP: Opcode = Reply
ARP_RARP: Sender's Hardware Address = 010203040506
ARP_RARP: Sender's Protocol Address = 192.168.0.30
ARP_RARP: Target's Hardware Address = 000102B78844
ARP_RARP: Target's Protocol Address = 192.168.0.200
```

**Beispiel 3: ARP-Reply**

```
00000:  01 02 03 04 05 06 00 01 02 B7 88 44 08 00 45 00   ..........ˆD..E.
00010:  00 30 04 63 40 00 80 06 74 2E C0 A8 00 C8 C0 A8   .0.c@._.t.À¨.ÈÀ¨
00020:  00 1E 0B F2 00 50 F7 86 37 52 00 00 00 00 70 02   ...ò.P÷†7R....p.

IP: Version = 4 (0x4)
IP: Header Length = 20 (0x14)
IP: Type of Service = Normal Service, Precedence Routine
IP: Total Length = 48 (0x30)
IP: Identification = 1123 (0x463)
IP: Flags Summary = 2 (0x2)
    .......0 = Last fragment in datagram
    ......1. = Cannot fragment datagram
IP: Fragment Offset = 0 (0x0) bytes
IP: Time to Live = 128 (0x80)
IP: Protocol = TCP - Transmission Control
IP: Checksum = 0x742E
IP: Source Address = 192.168.0.200
IP: Destination Address = 192.168.0.30
```

**Beispiel 4: IP-Protokollkopf**

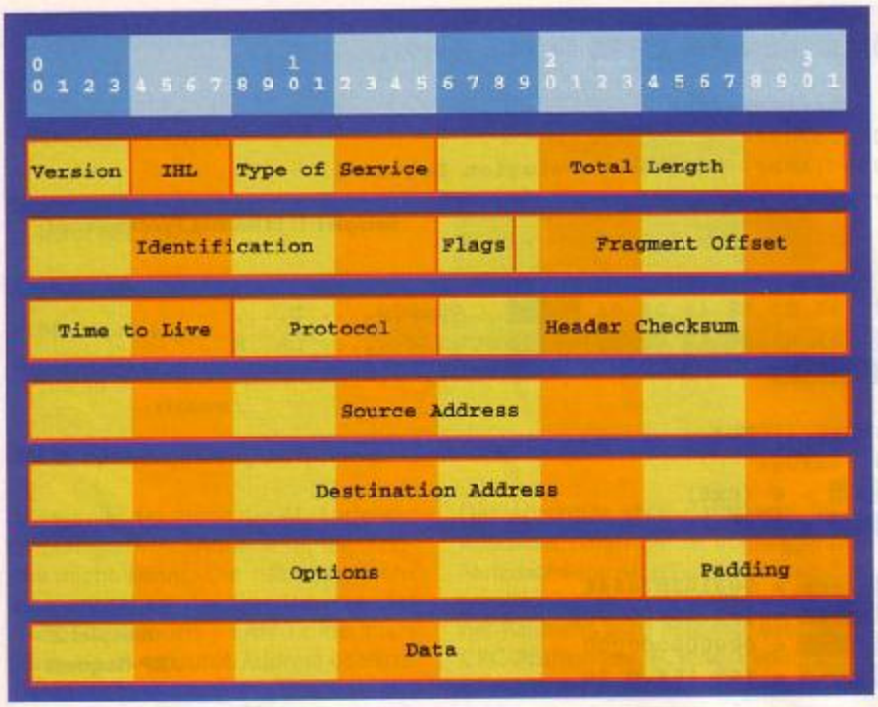



**Bild 4: Der IP Protokollkopf**

Broadcast-Frames durch einen Netzknoten erfolgt das Prüfen der Ziel-IP-Adresse. Daraufhin sollte genau ein Netzwerkteilnehmer (nämlich der mit der IP »192.168.0.30«) seine MAC-Adresse dem fragenden Knoten bekannt geben (Beispiel 3 auf Seite 9). Der Frame ähnelt dem vorangegangenen, nur trägt er einen anderen »Opcode« und die Absender- (Sender's Address) bzw. Zieladressen (Target's Address) sind entsprechend vertauscht. Der Internet-Knoten »192.168.0.200« kennt nun die Hardware-Adresse des Knotens »192.168.0.30« und kann im Folgenden direkt mit diesem kommunizieren.

## Internet Protocol (IP)

Das »Internet-Protokoll« (IP) dient dem Austausch von Datagrammen (IP-Datagrammen) zwischen Internet-Hosts. Es stellt den Hauptbestandteil der Inter-

| Feld | Beschreibung |
|---|---|
| Version | 4 Bit, Version des Internet-Datagramms (aktuell: 4) |
| IHL | 4 Bit, Länge des IP-Headers in 32-Bit-Worten (wenn keine Optionen angegeben sind: 5) |
| Type of Service (TOS) | 8 Bit, Diensttyp, Anweisungen für Router, bestimmte Pfade zu wählen (z.B. hoher Durchsatz, hohe Zuverlässigkeit, geringe Verzögerungszeit) |
| Total Length | 16 Bit, gibt Gesamtlänge des Datagramms in Byte an, berechnet sich aus der Länge der Kopf- und Datenfelder |
| Identification | 16 Bit, dient dem Kennzeichnen zusammengehöriger Datagramme, falls eine Fragmentierung notwendig ist |
| Flags | 3 Bit, dienen der Steuerung der Fragmentierung |
| Fragment Offset | 13 Bit, wird in fragmentierten Datagrammen verwendet, um anzugeben, an welcher Position das Fragment im ursprünglichen Datagramm stand, angegeben in Vielfachen von 8 Byte |
| Time to Live (TTL) | 8 Bit, Restlebensdauer des Datagramms, wird beim Senden gesetzt und bei jedem Routerdurchlauf dekrementiert. Bei TTL = 0 wird das Datagramm verworfen und nicht weiter übertragen. |
| Protocol | 8 Bit, gibt an, welches Protokoll der Transportschicht mitgeführt wird. Gängige Protokollnummern sind 1 (ICMP), 6 (TCP) und 17 (UDP). |
| Header Checksum | 16 Bit, Prüfsumme des IP-Headers, enthält NICHT die Nutzdaten, Berechnung nach [RFC-1071] |
| Source Address | 32 Bit, Absenderadresse des IP-Datagramms in der Form IP1.IP2.IP3.IP4 |
| Destination Address | 32 Bit, Zieladresse des IP-Datagramms in der Form IP1.P2.IP3.IP4 |
| Options | variable Länge, Unterstützung von Debugging-, Mess- und Sicherheitsfunktionen |
| Padding | variable Länge, Füllbytes zum Ergänzen der Länge des Headers auf ein Vielfaches von 32 Bit |
| Data | variable Länge, trägt Nutzdaten (Protokolle höherer Ordnung, z.B. TCP) des Datagramms |

**Tabelle 3: Die Elemente des IP-Protokollkopfs**

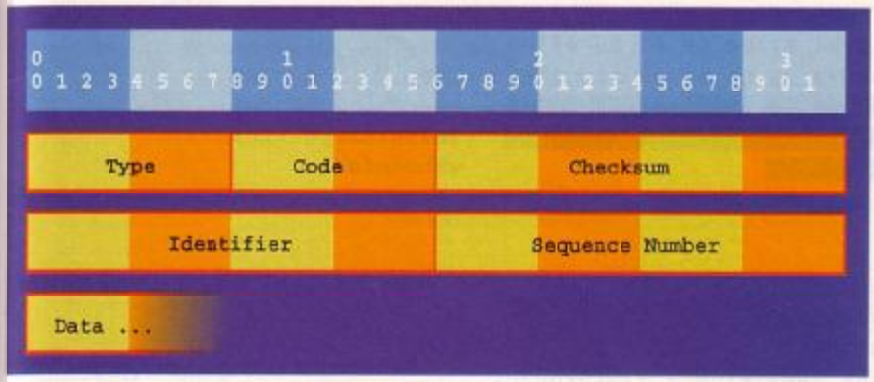



**Bild 5: Der ICMP-Echo-Protokollkopf**

net-Schicht dar (internet layer). Die folgenden Ausführungen beziehen sich auf die heute übliche Version 4 (IPv4), welche in [RFC-791] beschrieben ist.

Das IP-Protokoll baut auf einer unzuverlässigen und verbindungslosen Netzwerkschicht auf, wie beispielsweise Ethernet. Genauso gut würden drahtlose Verbindungen o.ä. diesem Protokoll genügen. Es werden hauptsächlich Mechanismen für die Adressierung der Hosts, das Routing (Wegwahl der Datenübertragung in einem Netzwerk) und die Fragmentierung bzw. Wiederzusammenfügung von Datagrammen bereitgestellt. Das IP-Protokoll garantiert aber weder eine verlässliche Datenübertragung von Host zu Host noch eine Übermittlung von Datagrammen in der richtigen Reihenfolge. Auch werden keine Bestätigungen für korrekt übertragene Datagramme übermittelt oder Mechanismen zur Flusskontrolle bereitgestellt. Die Zuverlässigkeit dieses Protokolls ist weitestgehend von der verwendeten Netzwerkschicht (network layer) abhängig.

IP-Datagramme werden als abgeschlossene Einheiten betrachtet und können aus mehren Gründen ihr Ziel nicht erreichen:

- Der Ziel-Host ist nicht mit dem Netzwerk verbunden
- Das Datagramm wurde beschädigt
- Das Datagramm wurde von einem Router fehlgeleitet

Diese Nachteile sind jedoch von Protokollen der höheren Schichten (Transportschicht, z.B. TCP-Protokoll) kompensierbar.

Werfen wir nun einen Blick auf den Protokollkopf eines IP-Datagramms. Die Anordnung der einzelnen Felder ist in Bild 4 dargestellt. Dabei repräsentiert jedes Farbfeld vier Bit-Positionen. In Tabelle 3 werden die jeweiligen Elemente beschrieben.

Das wohl bekannteste Merkmal dieses Protokolls ist die Einführung von Internet-Adressen (auch als IP-Adressen

| Feld | Beschreibung |
|---|---|
| Type | 8 Bit, enthält 8 für eine »Echo-Message« oder 0 für eine »Echo-Reply-Message« |
| Code | 8 Bit, enthält 0 |
| Checksum | 16-Bit, Prüfsumme der gesamten ICMP-Nachricht, beginnend mit dem Type-Feld. Berechnung nach [RFC-1071] |
| Identifier | 16 Bit Wert, um Bezug zwischen »Echos« und »Echo-Replies« herzustellen, vergleichbar mit einer Session-Nummer, kann 0 sein. |
| Sequence Number | 16-Bit-Wert, um Bezug zwischen »Echos« und »Echo-Replies« herzustellen, wird beim Senden jedes ICMP-Echos inkrementiert. Kann 0 sein. |
| Data | variable Länge, enthält mitgeführte Datenbytes |

**Tabelle 4: Die Elemente des ICMP-Echo-Protokollkopfs**

```
00000:  00 01 02 B7 88 44 00 10 A4 E7 B0 0B C8 00 45 00   ....·D.. ç°...E.
00010:  00 3C 99 00 00 00 20 01 7F 91 C0 A8 C0 17 C0 A8   .<™... . 'À¨..À¨
00020:  00 C8 08 00 49 5C 03 00 01 00 61 62 63 64 65 66   .È..I\....abcdef
00030:  67 68 69 6A 6B 6C 6D 6E 6F 70 71 72 73 74 75 76   ghijklmnopqrstuv
00040:  77 61 62 63 64 65 66 67 68 69                     wabcdefghi

ICMP: Packet Type = Echo
ICMP: Echo Code = 0 (0x0)
ICMP: Checksum = 0x495C
ICMP: Identifier = 768 (0x300)
ICMP: Sequence Number = 256 (0x100)
ICMP: Data: Number of data bytes remaining = 32 (0x0020)
```

**Beispiel 5: ICMP-Echo**

bzw. Hostnummern bezeichnet). Diese 32 Bit langen Nummern dienen der eindeutigen Bezeichnung von Quelle und Ziel eines Datagramms. Sie werden üblicherweise als vier einzelne, durch einen Punkt getrennte Dezimal-Bytes angegeben (z.B. »192.168.0.30«).
Die Größe von IP-Datagrammen beträgt maximal 64 KByte, sie wird jedoch häufig an die maximale Übertragungsgröße (MTU, Maximum Transfer Unit) des zu verwendenden Übertragungsmedium angepasst, um eine Fragmentierung zu vermeiden. Die übliche Datagrammgröße im Ethernet beträgt 1500 Byte. Auf den Vorgang der Fragmentierung und des Wiederzusammensetzens von Datagrammen wird an dieser Stelle nicht weiter eingegangen. Zum Weiterleiten (Demultiplexing) von Datagrammen an die richtige höhere Protokollschicht (z.B. TCP, UDP) dient das Feld »Protocol«, welches Auskunft über die Art des übertragenen Datenfeldes gibt.
Das Beispiel 4 auf Seite 9 stellt den Hex-Dump der ersten Bytes eines IP-Datagramms dar. Die Übertragung erfolgte über das Ethernet (physical layer) vom Netzknoten mit der IP-Adresse »192.168.0.200« (MAC: 00-01-02-B7-88-44) an den Netzknoten mit der IP-Adresse »192.168.0.30« (MAC: 01-02-03-04-05-06). Das abgebildete Datagramm trägt ein Segment der höheren Protokollschicht TCP (transport layer).

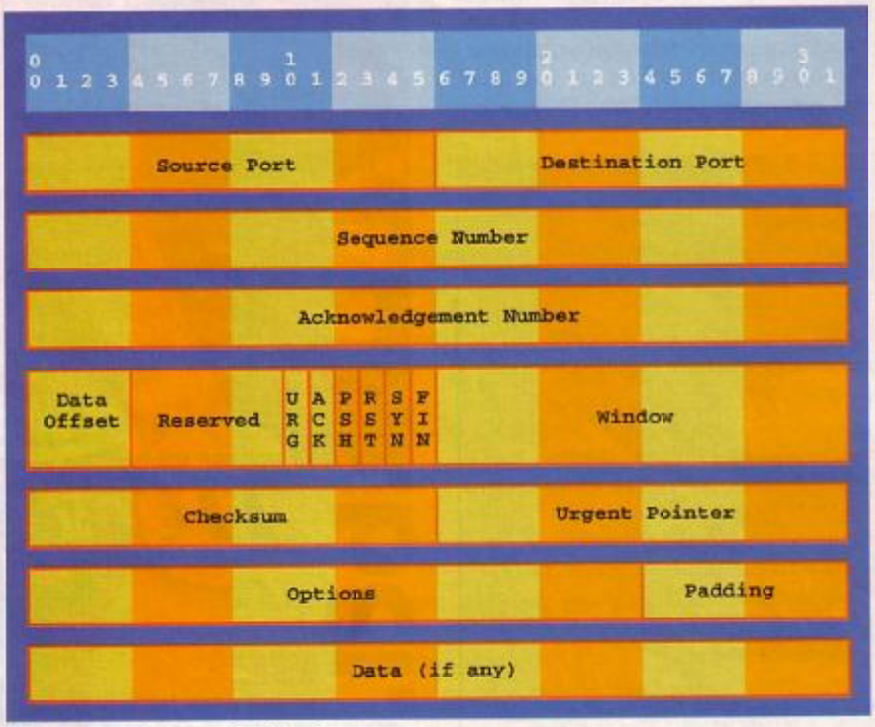



**Bild 6: Der TCP-Protokollkopf**

## Internet Control Message Protocol (ICMP)

Aufgabe des »Internet Control Message Protokolls« (ICMP) ist es in erster Linie, einen Mechanismus zur Rückmeldung beim Auftreten von Kommunikationsproblemen innerhalb der IP-Schicht bereitzustellen. Dadurch kann die IP-Schicht die Zuverlässigkeit des Datagrammdienstes erhöhen. Beispiele für Rückmeldungen sind die Nichtzustellbarkeit von Datagrammen, die Überlastung eines Gateways oder die Empfehlung einer kürzeren Route. Diese Beschreibung ist ein Auszug der Quelle [RFC-792]. Das ICMP-Protokoll benutzt dafür die Dienste der Internet-Schicht (internet layer) genauso wie ein anderes höheres Protokoll, stellt jedoch einen integralen Bestandteil dieser Schicht dar. ICMP-Nachrichten (ICMP-Messages) werden im Datenteil eines IP-Datagramms übertragen.
Für uns sind im Moment nur zwei ICMP-Meldungen von Interesse, die Messages »Echo« und »Echo-Reply«. Ein Host, welcher eine »ICMP-Echo«-Nachricht empfängt, sendet an den Absender dieser Nachricht eine »ICMP-Echo-Reply«-Nachricht zurück. Diesen Mechanismus nutzt hauptsächlich das Dienstprogramm »Ping«, welches Bestandteil eines jeden TCP/IP-fähigen Rechnersystems ist (z.B. Windows, Linux). Mit Hilfe dieses Programms lässt sich auf einfache Art und Weise die Funktionsfähigkeit eines Protokollstapels bzw. eines Rechnernetzes überprüfen. Ping sendet dazu ein IP-Datagramm an eine bestimmte Adresse, welches ein »ICMP-Echo« trägt. Der Protokoll-Stack des Empfängers sendet daraufhin eine »ICMP-Echo-Reply« zurück, anhand derer das Dienstpro-

| Portnummer | Kurzbezeichnung | Beschreibung |
|---|---|---|
| 7 | echo | sendet empfangene Daten zurück |
| 9 | discard | verwirft (aber bestätigt) empfangene Daten |
| 13 | daytime | sendet nach einem Verbindungsaufbau aktuelle Uhrzeit und Datum des Servers zurück (als ASCII-String) |
| 17 | quote of the day | sendet nach einem Verbindungsaufbau das Zitat des Tages zurück (als ASCII-String) |
| 19 | chargen | sendet kontinuierlich gesamten ASCII-Zeichensatz |
| 80 | www-http | Standard-Port eines HTTP-Webservers |

**Tabelle 5: Auswahl reservierter TCP-Ports**

gramm Ping Aussagen über z.B. die Erreichbarkeit, die Verzögerungszeit und die Zuverlässigkeit eines Hosts ermittelt. Bild 5 und Tabelle 4 (Seite 11) zeigen den Aufbau des ICMP-Protokollkopfs. In Beispiel 5 ist eine ICMP-Echo-Nachricht, welche 32 Datenbytes trägt, als Hexdump dargestellt und interpretiert.

Um nun auf dieses ICMP-Echo zu antworten, ist ein IP-Datagramm mit dem empfangenen Inhalt dem Absender zurück zu senden. Dazu müssen lediglich die Quell- und Ziel-IP-Adresse vertauscht, das »ICMP-Type-Feld« auf 0 gesetzt (= Echo Reply) und die Prüfsummen neu berechnet werden.

## Transmission Control Protocol (TCP)

Das »Transmission Control Protocol« (TCP) ist zur Zeit das wichtigste Protokoll für den kanalorientieren Datenaustausch zwischen genau zwei Endpunkten in einem paketvermittelnden Netzwerk und gehört damit der Transportschicht an (transport layer). Es stellt einen Dienst bereit, um einen Datenstrom von Bytes verlässlich und in der richtigen Reihenfolge »voll-duplex« zu übertragen. TCP wird von Applikationen genutzt, welche einen verbindungsorientierten Übertragungsdienst benötigen, z.B. E-Mail (SMTP, Simple Mail Transfer Protocol), Dateitransfer (FTP) oder der Terminal-Service (Telnet). Der folgende Überblick basiert auf [RFC-793].

Ein einfacher, unzuverlässiger Datagrammübertragungsdienst ist als Grundlage für das TCP schon ausreichend, da es robust gegenüber fehler-

| Feld | Beschreibung |
|---|---|
| Source Port | 16 Bit, Quellport |
| Destination Port | 16 Bit, Zielport |
| Sequ. Number | 32 Bit, Sequenznummer des ersten Datenbytes dieses Segmentes. Wenn *SYN*-Flag gesetzt ist, enthält dieses Feld die »Initial Sequence Number« (ISN) |
| Ack. Number | 32 Bit, Wenn das *ACK*-Flag gesetzt ist, enthält dieses Feld die nächste vom Sender dieses Segments erwartete Sequenznummer |
| Data Offset | 4 Bit, Länge des TCP-Headers in 32 Bit-Worten (wenn keine Optionen angegeben sind: 5) |
| Control Bits | *URG* »Urgent Pointer« (Dringlichkeitszeiger) ist gültig<br>*ACK* »Acknowledgement Number« ist gültig<br>*PSH* »Push-Funktion« (Anweisung an anderes TCP, Daten sofort an die Anwendungsschicht zu übergeben)<br>*RST* Verbindung zurücksetzen (Reset)<br>*SYN* Sequenznummern synchronisieren, belegt eine Sequenznummer<br>*FIN* Alle Daten sind gesendet, Verbindung beenden, belegt eine Sequenznummer |
| Window | 16 Bit, Anzahl von Datenbytes, welche der Absender dieses Segments im Moment aufnehmen kann |
| Checksum | 16 Bit, Prüfsumme von TCP-Header, »TCP-Pseudo-Header« (Einbeziehung von Ziel-IP, Quell-IP, Protokoll-Nummer und Segmentlänge) sowie des Datenteils, Berechnung nach [RFC-1071] |
| Urgent Pointer | 16 Bit, Zeiger auf dringende Daten innerhalb des Segments. Nur gültig, wenn *URG*-Flag gesetzt ist. |
| Options | variable Länge, Optionen dienen dem Festlegen spezieller Merkmale der Verbindung (wie z.B. »Maximum Segment Size«, MSS) |
| Padding | variable Länge, Füllbytes zum Ergänzen der Länge des Headers auf ein Vielfaches von 32 Bit |
| Data | variable Länge, trägt ein Segment von Nutzdaten (Anwendungsdaten oder Protokolle höherer Ordnung, z.B. HTTP). Die Übertragung von Daten in einem TCP-Segment ist optional. |

**Tabelle 6: Der TCP-Protokollkopf**

haften, verlorengegangenen, duplizierten und in falscher Reihenfolge gelieferten Datagrammen ist. Unzuverlässig bedeutet in diesem Zusammenhang lediglich, dass die Netzwerkschicht keine Garantie für die Zustellung der Datagramme geben kann. Dadurch eröffnen sich neben dem Einsatz als Transportprotokoll zusammen mit dem Internet-Protokoll weitere Einsatzmöglichkeiten. Im Folgenden wird aber nur auf die Verwendung im Zusammenhang mit IP eingegangen. Die Bereitstellung eines zuverlässigen Übertragungskanals wird durch folgende Mechanismen erreicht:

- Grundlegender Datentransfer

TCP übertragt einen Datenstrom zwischen beiden Hosts. Hierfür erfolgt eine Zerlegung dieses Datenstroms in Segmente, die dann als Pakete (so genannte IP-Datagramme) verschickt werden.

- Zuverlässigkeit

TCP muss gegenüber beschädigten, verlorengegangenen, duplizierten oder in falscher Reihenfolge gelieferten Datagrammen unempfindlich sein. Zu diesem Zweck wird jedem Byte im Datenstrom eine Sequenznummer zugeordnet. Jede dieser Sequenznummern muss von der Gegenstelle quittiert werden (*ACK*). Das Ausbleiben einer Bestätigung führt nach einer bestimmten Zeit (timeout) zu einer erneuten Übertragung der Daten (Retransmission). Beim Empfänger dienen die Sequenznummern dazu, die Daten wieder in der richtigen Reihenfolge anzuordnen und duplizierte Segmente zu verwerfen. Zusätzlich wacht ein Checksummen-Mechanismus über die Richtigkeit empfangener Daten.

- Flusskontrolle

Es wird ein Mechanismus bereitgestellt, mit dem der Empfänger bei jeder Bestätigung empfangener Daten dem Sender mitteilt, wie viele Daten er noch aufnehmen kann (Receive Window).

- Multiplexing

Das TCP-Protokoll erweitert die Adressierung von Hosts mittels IP-Adressen um »Ports«. Ein Port ist ein 16-Bit-Wort und dient praktisch dem Multiplexing der Host-IP. Durch Nutzung unterschiedlicher Ports kann ein TCP/IP-Netzknoten Verbindungen mit mehreren Partnern gleichzeitig unterhalten und unterscheiden. Jede Kombination einer Internet-Adresse mit einem Port stellt einen Kommunikationsendpunkt dar und wird im allgemeinen als »Socket« bezeichnet. Für viele Dienste sind Portnummern vordefiniert, um bei Zugriff auf einen unbekannten Server den richtigen Dienst (sog. Standarddienste) zu erreichen. In Tabelle 5 auf Seite 13 ist eine kleine Auswahl dieser Portnummern dargestellt, welche sich besonders für Testzwecke eignen. [RFC-1700]

- Verbindung

Die gerade beschriebenen Mechanismen erfordern das Einrichten einer Verbindung zwischen zwei Hosts und das Verwalten von Informationen über diese Verbindung. Da Verbindungen über ein unzuverlässiges Netzwerksystem aufgebaut werden müssen, kommt ein »Handshake-System« mit zeitgesteuerten Sequenznummern zur Anwendung. Das TCP-Protokoll basiert auf einem Client/Server-Modell. Als Client wird der Host bezeichnet, der den Übertragungskanal öffnet (active open), und der Server ist der auf eine Verbindung wartende Host (passive open). Die Datenübertragung findet voll-duplex bis zum Schließen des Kanals statt.

Eine TCP-Implementierung stellt der übergeordneten Softwareschicht (ap-

```
00000:  01 02 03 04 05 06 00 01 02 B7 88 44 08 00 45 00   ..........·ˆD..E.
00010:  00 30 04 63 40 00 80 06 74 2E C0 A8 00 C9 C0 A8   .0.c@.€.t.À¨.ÉÀ¨
00020:  00 1E 0B F2 00 50 F7 86 37 52 00 00 00 00 70 02   ...ò.P÷†7R....p.
00030:  40 00 85 CD 00 00 02 04 05 B4 01 01 04 02         @....Í.....´....

TCP: Source Port = 0x0BF2
TCP: Destination Port = Hypertext Transfer Protocol
TCP: Sequence Number = 4152768338 (0xF7863752)
TCP: Acknowledgement Number = 0 (0x0)
TCP: Data Offset = 28 (0x1C)
TCP: Reserved = 0 (0x0000)
TCP: Flags = 0x02 : ....S.
     ..0..... = No urgent data
     ...0.... = Acknowledgement field not significant
     ....0... = No Push function
     .....0.. = No Reset
     ......1. = Synchronize sequence numbers
     .......0 = No Fin
TCP: Window = 16384 (0x4000)
TCP: Checksum = 0x85CD
TCP: Urgent Pointer = 0 (0x0)
TCP: Option Type = Maximum Segment Size
     Option Length = 4 (0x4)
     Maximum Segment Size = 1460 (0x5B4)
TCP: Option Nop = 1 (0x1)
TCP: Option Nop = 1 (0x1)
TCP: Option Type = Sack Permitted
     Option Length = 2 (0x2)
```

**Beispiel 6: TCP-Protokollkopf**

plication layer) einen Satz von Methoden und Eigenschaften (bzw. Ereignissen) bereit, mit der diese die Funktionalität des TCP nutzen kann. Wichtige Methoden sind z.B. Öffnen bzw. Schließen eines Kanals und das Senden und Empfangen von Daten.

## Aufbau eines TCP-Segments

In Bild 6 auf Seite 12 und Tabelle 6 (gegenüber) sind die Felder des TCP-Protokollkopfes dargestellt und beschrieben. Beispiel 6 stellt ein TCP-Segment aus der Phase eines Verbindungsaufbaus dar (Segment trägt das *SYN*-Flag und die »Initiale Sequenznummer«).

## Funktionsweise von TCP

Eine TCP-Verbindung durchläuft vom Aufbau bis zum Beenden mehrere Zustände. Diese lassen sich am besten mittels eines »Zustandsgraphen« beschreiben (Bild 7, Seite 16), die Bedeutung dieser Zustände ist Tabelle 7 auf Seite 18 zu entnehmen. Die Übergänge zwischen diesen Zuständen erfolgen als Reaktion auf Ereignisse, wie z.B. Funktionsaufrufe (Verbindung herstellen, Verbindung beenden...), das Vorhandensein von Flags im aktuellen Segment (*ACK, SYN, FIN...*) oder Timeouts. Jeder TCP-Endpunkt muss den dargestellten Zustandsgraphen implementieren. Obwohl dieser schon relativ umfangreich ist, stellt er doch nur einen Überblick über die Arbeitsweise des TCP dar.

Im Folgenden sollen kurz die drei wesentlichsten Phasen einer Verbindung erläutert werden:

- Verbindungsaufbau

Stellen wir uns einen Host B vor, welcher ein Webserver ist, also einen Kanal passiv geöffnet hat und sich somit im TCP-Zustand ***LISTEN*** befindet. Ein Host A möchte nun zu diesem Server eine Verbindung aufbauen, um mit ihm Daten auszutauschen (z.B. eine Webseite abzurufen). Dazu wird im Host A die TCP-Funktion ***ActiveOpen*** aufgerufen. Im Folgenden ist nun zwischen den beiden TCPs eine Verbindung aufzubauen. Das geschieht durch den Austausch und die Synchronisation der Sequenznummern. Dieser Vorgang wird auch als »three-way-handshake« bezeichnet. Jede TCP-Schicht besitzt zu

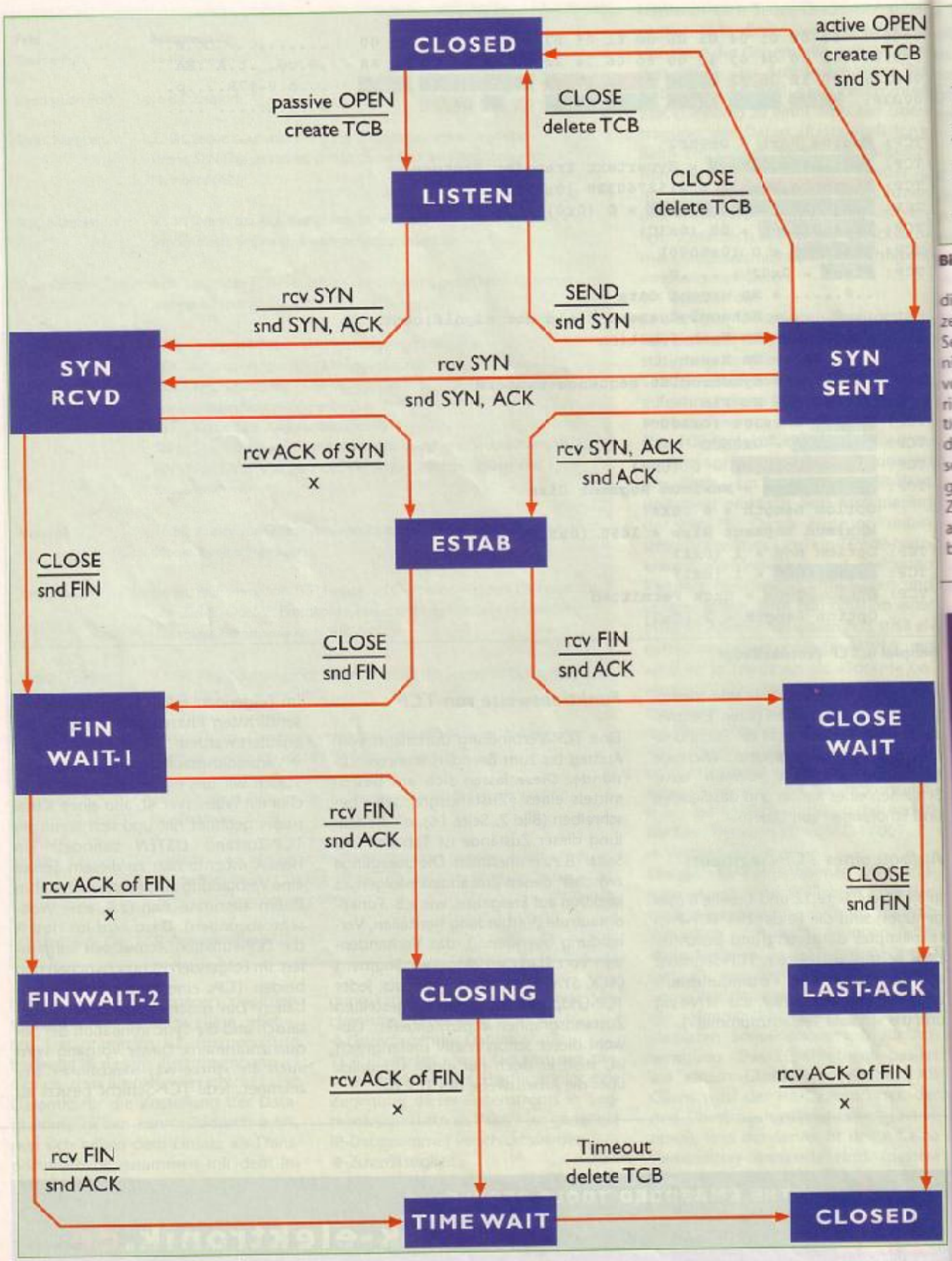



**Bild 7: Der TCP-Zustandsgraph**

| | TCP A | | | | TCP B |
|---|---|---|---|---|---|
| 1. | *CLOSED* | | | | *LISTEN* |
| 2. | *SYN-SENT* | → | <*SEQ*=100><*CTL*=*SYN*> | → | *SYN-RECEIVED* |
| 3. | *ESTABLISHED* | ← | <*SEQ*=300><*ACK*=101><*CTL*=*SYN,ACK*> | ← | *SYN-RECEIVED* |
| 4. | *ESTABLISHED* | → | <*SEQ*=101><*ACK*=301><*CTL*=*ACK*> | → | *ESTABLISHED* |

**Bild 8: TCP-Verbindungsaufbau**

diesem Zweck einen Generator zum Erzeugen einer anfänglichen (initialen) Sequenznummer (ISN). Dieser ist nichts weiter als ein freilaufender Zähler von 32 Bit Breite, welcher mit einer Periodendauer von etwa 4 µs inkrementiert wird. Zum Zeitpunkt des Verbindungsaufbaus ergibt sich die erste zu sendende Sequenznummer des jeweiligen TCPs durch das Auslesen dieses Zählers. In Bild 8 ist der Verbindungsaufbau schematisch dargestellt. Zur besseren Übersicht ist diese Darstellung auf das Notwendigste beschränkt. An beiden Seiten sind die Zustände der TCP-Schichten dargestellt, dazwischen die übertragenen Sequenznummern (*SEQ*), Acknowledgementnummern (*ACK*) und Flags (*CTL*). Host A teilt dem Host B seine initiale Sequenznummer durch ein Segment mit dem Flag *SYN* mit und geht gleichzeitig in den Zustand ***SYN-SENT*** über. Nach dem Empfangen dieses Segments durch Host B wechselt dieser in den Zustand ***SYN-RECEIVED***, gibt durch ein Segment (Schritt 3) seine eigene ISN bekannt (Flag *SYN* gesetzt) und bestätigt gleichzeitig den Empfang der ISN des Hosts A (Flag *ACK* gesetzt, *ACK*=101). Dabei ist zu beachten dass ein *SYN*-Flag eine Sequenznummer belegt und demzufolge durch ein *ACK* = ISN + 1 bestätigt wird. Im Schritt 4 wiederum bestätigt Host A den Empfang der ISN des Hosts B (Flag *ACK* gesetzt, *ACK*=301). Damit ist die Verbindung aufgebaut (***ESTABLISHED***), und mit dem darauffolgenden Schritt kann die Datenübertragung beginnen.

| Zustand | Bedeutung |
|---|---|
| *LISTEN* | Warten auf eingehende Verbindung eines Remote-TCP (*SYN*) |
| *SYN-SENT* | Warten auf einen Verbindungsaufbau nach dem Senden einer Verbindungsaufbau-Anforderung (*SYN*, ggf. *ACK*) |
| *SYN-RECEIVED* | Warten auf die Bestätigung nach dem Senden und Empfangen einer Verbindungsaufbau-Anforderung (*ACK*) |
| *ESTABLISHED* | Kanal ist geöffnet, Daten können gesendet und empfangen werden |
| *FIN-WAIT-1* | Warten auf das Beenden der Verbindung durch das Remote-TCP (*FIN*) oder das Bestätigen der eigenen Verbindungsauflösung (*ACK*) |
| *FIN-WAIT-2* | Warten auf das Beenden der Verbindung durch das Remote-TCP (*FIN*) |
| *CLOSE-WAIT* | Warten auf das Beenden der Verbindung durch die lokale Anwendungsschicht |
| *CLOSING* | Warten auf das Beenden der Verbindung durch das Remote-TCP (*FIN*) |
| *LAST-ACK* | Warten auf das Bestätigen der eigenen Verbindungsauflösung durch das Remote-TCP (*ACK*) |
| *TIME-WAIT* | Warten einer bestimmten Zeitdauer, um sicherzustellen dass das Remote-TCP die Bestätigung (*ACK*) seiner Verbindungsauflösung (*FIN*) erhält |
| *CLOSED* | Keine Verbindung vorhanden |

**Tabelle 7: Zustände eines TCP**

◆ Datenübertragung

Nach dem Verbindungsaufbau und dem Übergang beider TCPs in den Zustand ***ESTABLISHED*** kann nun die Datenübertragung in beide Richtungen beginnen. In Bild 9 ist eine solche Datenübertragung schematisch dargestellt. Hierbei handelt es sich um die Weiterführung des Beispiels aus dem Abschnitt »Verbindungsaufbau«. Im Schritt 5 überträgt Host A an Host B ein 50 Byte umfassendes Datensegment. Daraufhin bestätigt Host B das Empfangen der 50 Byte durch ein Segment (Flag *ACK* gesetzt, *ACK* = 101 + 50 = 151), welches wiederum 50 Datenbyte enthält, die Host B zu Host A übertragen möchte. Nach dem Empfangen dieses Segments durch Host A (Schritt 7) sendet dieser noch einmal ein Segment mit 50 Datenbytes an Host B. Es erfolgt abermals die Bestätigung der eben empfangenen Daten durch Setzen des Flags *ACK* und der »Acknowledgement Number« = 301 + 50 = 351. Im Schritt 8 wird nur ein Bestätigungssegment von Host B zu Host A übertragen, welches selbst keine Daten enthält. Die Datenübertragung erfolgt solange, bis einer der beiden TCPs die Verbindung beendet.

◆ Verbindungsabbau

Host A will die Verbindung zu Host B lösen. Zur Verdeutlichung wird das Beispiel weitergeführt (Bild 10). Host A sendet nun ein Segment mit gesetztem *FIN*-Flag und geht damit in den Zustand ***FIN-WAIT-1*** über. Host B bestätigt den Empfang des *FIN*-Flags und wechselt in den Zustand ***CLOSE-WAIT*** (Schritt 11). Dabei ist zu beachten, dass das *FIN*-Flag genauso wie am Anfang das *SYN*-Flag eine Sequenznummer belegt und somit bestätigt werden muss. Dieser Mechanismus dient dem Schutz der Flags vor Verlust oder Verfälschung. Nach dem Beenden seiner Übertragung schließt nun Host B die Verbindung und teilt dies Host A ebenfalls durch die Übertragung eines *FIN*-Flags mit (Schritt 12). Nachdem die Bestätigung des *FIN*-Flags bei Host B eingegangen ist, kann die Verbindung auf dieser Seite geschlossen werden (Schritt 13). Host A wartet nach dem Senden der Bestätigung noch eine bestimmte Zeit, um eventuelle »Retransmissionen« von Schritt 12 abermals zu bestätigen, und schließt nach Ablauf der Zeit seine Seite (Schritt 14). Die Verbindung kann nun als vollständig gelöst betrachtet werden.

## Hyper Text Transfer Protocol (HTTP)

Das »Hyper Text Transfer Protokoll« (HTTP) lässt sich der Anwendungsschicht (application layer) zuordnen. Es ist ein universelles, zustandsloses und

| | TCP A | | | TCP B |
|---|---|---|---|---|
| 5. | *ESTABLISHED* | →<*SEQ*=101><*ACK*=301><*CTL*=*ACK*><DATA> | → | *ESTABLISHED* |
| 6. | *ESTABLISHED* | ←<*SEQ*=301><*ACK*=151><*CTL*=*ACK*><DATA> | ← | *ESTABLISHED* |
| 7. | *ESTABLISHED* | →<*SEQ*=151><*ACK*=351><*CTL*=*ACK*><DATA> | → | *ESTABLISHED* |
| 8. | *ESTABLISHED* | ←<*SEQ*=351><*ACK*=201><*CTL*=*ACK*> | ← | *ESTABLISHED* |

**Bild 9: TCP-Datentransfer**

| | TCP A | | | TCP B |
|---|---|---|---|---|
| 9. | *ESTABLISHED* | | | *ESTABLISHED* |
| 10. | (Close) | | | |
| | *FIN-WAIT-1* | → | <*SEQ*=201><*ACK*=351><*CTL*=*FIN,ACK*> | → *CLOSE-WAIT* |
| 11. | *FIN-WAIT-2* | ← | <*SEQ*=351><*ACK*=202><*CTL*=*ACK*> | ← *CLOSE-WAIT* |
| | | | | (Close) |
| 12. | *TIME-WAIT* | ← | <SEQ=351><*ACK*=202><*CTL*=*FIN,ACK*> | ← *LAST-ACK* |
| 13. | *TIME-WAIT* | → | <*SEQ*=202><*ACK*=352><*CTL*=*ACK*> | → *CLOSED* |
| 14. | (Timeout) | | | |
| | *CLOSED* | | | |

**Bild 10: TCP-Verbindungsabbau**

objektorientiertes Protokoll. Um seine Funktion zu gewährleisten, ist eine zuverlässige, »streamorientierte« Transportschicht (wie z.B. TCP) erforderlich. HTTP wird hauptsächlich zum Übertragen von HTML-Dokumenten (Hyper Text Markup Language, universelle Seitenbeschreibungssprache) oder multimedialen Inhalten (Bilder, Samples) zwischen Webservern und Internet-Browsern verwendet. Der folgende Überblick bezieht sich auf die Version 1.0 des Protokolls, welche in [RFC-1945] vorgestellt wird.

Das HTTP-Protokol arbeitet nach dem Schema »Request« (Anforderung) und »Response« (Antwort). Im einfachsten Fall stellt ein Client eine Verbindung zu einem Server her und sendet eine Anforderung nach einem bestimmten Inhalt, welcher durch einen »URI« (Uniform Ressource Identifier, Pfad und Name einer Ressource im Internet, ähnlich einem Dateisystem) spezifiziert ist. Der Server überträgt daraufhin das gewünschte Dokument an den Client, und die Verbindung wird wieder beendet. Die Datenübertragung mittels HTTP ist transparent und gut nachvollziehbar, da prinzipiell nur einfache ASCII-Strings ausgetauscht werden.

## HTTP-Request

Werfen wir zunächst einen Blick auf den Aufbau eines solchen »Requests« (Bild 11 auf Seite 21). Das erste Feld enthält die Art des Requests (auch Methode genannt). Im Standard HTTP V1.0 sind drei Methoden definiert (Tabelle 8). Danach folgt ein Leerzeichen (in Bild 11 und 12 mit »SP« bezeichnet). Im nächsten Feld wird der gewünschte URI angegeben. Anschließend wird die Version des HTTP-Protokolls übertragen. Jede Zeile schließt mit einem Wagenrücklauf (»CR«) und Zeilenvorschub (»LF«) ab. Anschließend kann eine beliebige Anzahl von optionalen »Header«-Feldern folgen (farbig dargestellt). Diese dienen beispielsweise dem bekannt Geben der Fähigkeiten des Clienten an den Server.

| Methode | Beschreibung |
|---|---|
| GET | Anfordern einer Datei (Ressource) vom Server |
| HEAD | ähnlich GET, es werden jedoch nur die Header-Informationen der Ressource übertragen, nicht aber die Ressource selbst |
| POST | dient dem Übertragen von Informationen vom Client zum Webserver (z.B. Formulardaten) |

**Tabelle 8: HTTP-Methoden**

Der gesamte Header-Block wird durch ein zusätzliches »CR+LF« abgeschlossen. Im Anschluss daran folgt ein optionaler Datenblock, welcher ebenfalls durch ein zusätzliches »CR+LF« abgeschlossen wird. Dieser Datenblock

```
GET / HTTP/1.1
Accept: */*
Accept-Language: de
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/4.0 (compatible; MSIE 5.01; Windows NT 5.0)
Host: 192.168.0.30
Connection: Keep-Alive
```

**Beispiel 7: HTTP-Request eines Internet-Browsers**

# Embedded Intelligence 2002

## DER KONGRESS ZUR

# Embedded Systems 2002

# Call for Papers

*Die erfolgreichste Veranstaltung Deutschlands im Bereich »Embedded Engineering« lädt erneut alle kreativen und innovativen Entwickler und Designer zum Stelldichein. Präsentieren Sie Ihre Ideen, Lösungen und Erfahrungen auf dem Kongress »Embedded Intelligence 2002«, der parallel zur Fachmesse »Embedded Systems 2002« vom 19. bis 21. Februar 2002 in Nürnberg stattfindet!*

*Stellen Sie Ihre Ideen und Lösungen auf den Prüfstand. Auf keinem anderen Kongress erreichen Sie so viele interessierte und kritische Fachkollegen, mit denen Sie in kollegialer Atmosphäre diskutieren können.*

*Geben Sie Ihre Erfahrungen weiter, präsentieren Sie Ihre Lösungen und Applikationsideen, stellen Sie die Ergebnisse Ihrer Forschungs- und Diplomarbeiten vor. Helfen Sie mit, neue Technologien einer breiten Entwicklerschar näher zu bringen. Nutzen Sie diese Chance!*

*Senden Sie uns bitte Ihren Vorschlag eines praxisorientierten Fachvortrages!*

**Folgende Themenbereiche seien als Anregung genannt:**

- Embedded Designprozess/ Tools und Debugging
- Embedded Internet/ Internet Security
- Wired and wireless Communication
- Softwareentwicklung, -qualität und -sicherheit
- Neue embedded Bausteine und deren Anwendungen
- Embedded Betriebssysteme
- Embedded Signalverarbeitung
- Bussysteme in Embedded-Anwendungen
- Automotive- und Control-Anwendungen
- Interessante Embedded-Applikationen
- Echtzeitsysteme und -anwendungen

**Weitere Themen aus dem Bereich »Embedded Engineering« sind willkommen.**

**Einsendeschluss: 08. Oktober 2001**
Senden Sie die aussagefähige Kurzfassung Ihres Beitrages (maximal drei A4-Seiten, Deutsch oder Englisch) bitte bis zum 08.10.2001 an:

Redaktion DESIGN&ELEKTRONIK
z.Hd. Frau Renate Ester
Gruber Straße 46a, 85586 Poing
Tel. 0 81 21/95-13 42, Fax 0 81 21/95-16 54
Email: rester@design-elektronik.de

**Termine**

| | |
|---|---|
| Einsendung der Zusammenfassung: | 08.10.2001 |
| Benachrichtigung der Autoren: | 06.11.2001 |
| Einsendung des endgültigen Beitrags: | 14.12.2001 |

Der endgültige Beitrag sollte zehn Seiten DIN A4 nicht wesentlich überschreiten

Eine Veranstaltung der
DESIGN&ELEKTRONIK
PRODUKTE UND KNOW-HOW FÜR DEN ELEKTRONIK-ENTWICKLER

Sie werden zu erfahrenen Entwicklern und junger Design-Ingenieuren sprechen. Deshalb ist eine rein technische Abhandlung Ihres Themas zwingend notwendig. Marketing-orientierte Beiträge werden nicht akzeptiert.

**Die Kurzfassung sollte enthalten:**

- Titel des Beitrages
- Themenschwerpunkte, die den Inhalt am besten beschreiben
- Name des Autors
- Post- und Email-Adresse des Autors
- Telefon- und Faxnummer

**Wir freuen uns auf Ihre Beiträge!**

Prof. Dr.-Ing. habil. H. Beikirch, Universität Rostock
Prof.-Dr.-Ing. M. Sturm, HTWK Leipzig

Mitglieder des Kongressbeirates
Embedded Intelligence 2002

| Method | SP | Request-URI | SP | HTTP-Version | CR+LF |
|---|---|---|---|---|---|
| Header: | SP | Value | CR+LF | | |
| Header: | SP | Value | CR+LF | | |
| CR+LF | | | | | |
| Daten | | | | | |
| CR+LF | | | | | |

**Bild 11: HTTP-Request**

dient beispielsweise der »PUT«-Methode zum Übertragen einer Ressource vom Client zum Server.
Beispiel 7 (Seite 19) stellt einen »HTTP-Request« vor. Es wird von einem Client (Internet-Explorer) der URI »/« angefordert (Benutzung der Methode »GET«). Ein Webserver sollte daraufhin seine entsprechende Startseite (Standardseite) übertragen (meist »index.html«).

## HTTP-Response

Als Reaktion auf einen Request eines Clienten antwortet der Webserver mit einem »HTTP-Response«. Dessen Aufbau ist in Bild 12 angegeben. Das erste zu übertragende Feld enthält die Versionsnummer der HTTP-Implementierung. Daran schließt sich der »Status-Code« mit einem dazugehörigen String an. Status-Codes sind 3-stellige Zahlen und werden in mehrere Klassen unterteilt. Sie sind ausführlich in [RFC-1945] beschrieben und dienen beispielsweise der Rückmeldung, ob ein vorangegangener »Request« erfolgreich ausgeführt werden konnte. Genau wie beim Request schließt sich nun ein optionaler Block mit »Header-Feldern« an. Damit kann der Server unter anderem den Typ und Länge des Inhaltes bekannt geben. Im Anschluss an die Header-Felder kann noch ein durch »CR+LF« abgeschlossener Datenblock übertragen werden. Beispiel 8 gibt solch einen »HTTP-Response« an. Im Anschluss an den Kopfteil wird eine HTML-Webseite übertragen (nur die ersten Zeilen sind dargestellt). (cg)

| HTTP-Version | SP | Status-Code | SP | Reason-Phrase | CR+LF |
|---|---|---|---|---|---|
| Header: | SP | Value | CR+LF | | |
| Header: | SP | Value | CR+LF | | |
| CR+LF | | | | | |
| Daten | | | | | |
| CR+LF | | | | | |

**Bild 12: HTTP-Response**

```
HTTP/1.0 200 OK
Content-Type: text/html

<html>
  <head>
    <meta http-equiv="refresh"...
...
```

**Beispiel 8 HTTP-Response eines Webservers, nur die ersten Zeilen sind dargestellt**

**DESIGN&ELEKTRONIK**
**ENTWICKLERFORUM**

http://www.elektroniknet.de
click to register

# Embedded Internet

## Am 11.10.2001 im M,O,C, München

Möchten Sie Entwicklungszeit sparen? Von den Erfahrungen anderer Entwickler auf dem Gebiet »Embedded Internet« profitieren? Dann finden Sie hier die Lösungen!
Immer mehr Entwickler sehen sich mit der Anforderung konfrontiert, die unterschiedlichsten Geräte ans Internet zu bringen. Eine enorme Herausforderung, schließlich sind statt einer sperrigen und vor allem teuren PC-Lösung einfache, kleine, strom- und kostensparende Module mit 8-Bit- bzw. 16-Bit-Mikrocontrollern gefragt, die die benötigte Webfunktionalität komplett bereitstellen. Wenn der Entwickler sich auf die für seine Applikation wesentlichen Möglichkeiten von TCP/IP beschränkt, lässt sich der Aufwand drastisch reduzieren. Doch welche Teile der Protokolle werden unbedingt gebraucht? Was für Bausteine sind mindestens nötig? Bedarf es eines Echtzeit-Betriebssystems? Wie lässt sich der TCP/IP-Stack mit endlichem Aufwand realisieren?

Erschöpfende Antworten finden Sie auf dem DESIGN&ELEKTRONIK-Entwicklerforum »Embedded Internet«. Es erwartet Sie die Vorstellung einer Embedded-Webserver-Komplettlösung im Detail, Sie erfahren vieles über Embedded-Internet-Bausteine, lernen die Vorzüge von Java in Embedded-Internet-Anwendungen kennen und bekommen Know-how bezüglich der Datensicherheit vermittelt. Weitere Themen sind die Verbindung zwischen einem CAN-Netzwerk und dem Internet sowie alternative Wege in das Internet wie WAP oder Powerline.
Auf diesem DESIGN&ELEKTRONIK-Entwicklerforum werden die Probleme nicht nur genannt, sondern auch gelöst. Ein Besuch des Forums bringt Ihnen im Bereich der Internetanbindung mit Sicherheit den entscheidenden Wettbewerbsvorteil. Eine Gelegenheit, die Sie sich auf keinen Fall entgehen lassen sollten! Also: Anmeldung ausfüllen und faxen. Oder Sie melden sich online an unter: **www.elektroniknet.de**.

Altium – Think it, Design it, Build it
Evolved for the eWorld ADVANTECH
HK ELEKTRONIK
Gesytec
GETembedded!
FS Forth-Systeme GmbH – ein Unternehmen der ES Embedded Solutions AG
SSV
euros
SMA
HITACHI Inspire the Next
Fraunhofer Institut Integrierte Schaltungen
Garz & Fricke
MICRO CONSULT
JUMPtec ...always a Jump ahead!
KEIL SOFTWARE
WindRiver
SND SMART NETWORK DEVICES

Eine Veranstaltung der

**DESIGN&ELEKTRONIK**

PRODUKTE UND KNOW-HOW FÜR DEN ELEKTRONIK-ENTWICKLER

# KONGRESSPROGRAMM

| | | |
|---|---|---|
| **09:00–10:00** | **Einführung**<br>Einsatz von Embedded Internet auf Standardplattformen oder anwendungsoptimiert? | ***Peter Heusinger, Fraunhofer Institut Integrierte Schaltungen*** |
| | **Kurs 1: Mini-Webserver auf Mikrocontroller-Basis** | |
| 10:00–10:45 | embedded Webserver auf PIC-Basis | ***Prof. Dr. Francesco Volpe, FH Aschaffenburg*** |
| 10:45–11:30 | easyWEB - Webserver auf MSP430-Basis | ***Prof. Dr. Matthias Sturm, HTWK Leipzig*** |
| 11:30–12:00 | Implementierung eines Web-Servers mit einem Low-Cost-Mikrocontroller | ***Harald Kreidl, Motorola*** |

**12:00 bis 13:30 Mittagspause, Ausstellung**

| | | |
|---|---|---|
| **Kurs 2:** | **Embedded-Internet-Bausteine** | |
| 13:30–14:00 | Skalierbare Embedded Internet Plattform auf Basis der SuperH RISC Familie | ***Dr. Manfred Schlett, Hitachi Europe*** |
| 14:00–14:30 | Designing a low-cost single-chip solution for an embedded web-server application | ***Dr. Christoph Baumhof, Hyperstone*** |
| 14:30–15:00 | Ubiquitäre Kommunikation – die Entwicklung eines Internet Prozessors | ***Bert Lutje Berenbroek, Ubicom Inc.*** |
| 15:00–15:30 | Ausstellung | |

| | | |
|---|---|---|
| **Kurs 3:** | **Embedded-Internet-Applikationen** | |
| 13:30–14:00 | Embedded Security | Volker ***Wittelsberger, HSP*** |
| 14:00–14:30 | Schlüsselkomponenten von Embedded Internetlösungen | ***Thomas Batt, Microconsult*** |
| 14:30–15:00 | Webserver in einem CAN-Knoten auf MSP430-Basis | ***Jean Randhahn, FH Stralsund,*** ***Prof. Dr. Helmut Beikirch, Universität Rostock*** |
| 15:00–15:30 | Controlling a CAN-network using an embeded webserver | ***Stewart Cording, National Semiconductor*** |

**15:30 bis 16:00 Kaffeepause, Ausstellung**

| | | |
|---|---|---|
| **Kurs 4:** | **Java in embedded internet Anwendungen** | |
| 16:00–16:30 | Embedded Web-Server mit Java | ***Carsten Reinartz, Fraunhofer Institut IIS*** |
| 16:30–17:00 | Java-Technologie vereinfacht Informationsmanagement in der Industrieautomation | ***Peter Duchemin, Smart Network Devices*** |
| 17:00–17:30 | Service-Gateway-Software zur intelligenten Vernetzung von Geräten | ***Dr. Dimitar Valtchev, ProSyst Software*** |

| | | |
|---|---|---|
| **Kurs 5:** | **Alternative Wege ins Internet** | |
| 16:00–16:30 | WAP-integration von Embedded Systemen | ***Klaus D. Walter, SSV*** |
| 16:30–17:00 | Embeded Internet über Powerline | ***Prof. Dr. Francesco Volpe, FH Aschaffenburg*** |
| 17:00–17:30 | Einsatzmöglichkeiten PC-basierter Internettechniken, z.B. in der Automatisierungstechnik | ***Rudolf Barth, jUMPtec*** |

Änderungen vorbehalten

# Anmeldung FAX 081 21/95 16 54

**Teilnahme-Gebühr: DM 490,–**

**Ich melde mich verbindlich an:**

**Vormittag:**
**Einführung + Kurs 1**

**Nachmittag:**
☐ **Kurs 2 oder** ☐ **Kurs 3**
☐ **Kurs 4 oder** ☐ **Kurs 5**

Firma

Vorname Name

Straße

PLZ Ort

Telefon Fax

Email

Datum Unterschrift EI/DE/EX

Die Preise verstehen sich zuzügl. der gesetzl. MwSt. In diesem Betrag enthalten sind Tagungsunteragen (Kongress) sowie Mittagsessen und Pausengetränke. Studenten (50% Rabatt) bitte Immatrikulationsbescheinigung beilegen. Die Rechnungsstellung erfolgt mit der Anmeldebestätigung. Bei Stornierung der Anmeldung bis 10 Tage vor Veranstaltungsbeginn erheben wir eine Bearbeitungsgebühr von DM 100,– (zzgl. MwSt.), bei späterer Absage oder Nichterscheinen wird die gesamte Tagungsgebühr fällig.

Info: Hilde Buchner, Tel.: 08121/95-1345, Fax: 08121/95-1654, HBuchner@design-elektronik.de · **WEKA-Fachzeitschriften-Verlag, 85586 Poing, Gruber Straße 46a**

*TCP/IP-Knoten auf Mikrocontrollerbasis*

# Ein Mikrocontroller wird Internetserver

**Zur Implementierung eines kompletten TCP/IP-Protokollstapels werden erhebliche Mengen Arbeitsspeicher und Rechenzeit von einem Mikrocontrollersystem benötigt. Die Wahl eines Hochleistungs-Mikrocontrollers (32-Bit-Klasse, über 30 MHz, mehr als 5 KByte RAM) wird jedoch meist aus Kostengründen abgelehnt. Der Wunsch nach einer ressourcenschonenden Anbindung von weniger leistungsfähigen Mikrocontrollern (16-Bit-Klasse und darunter) an das Internet ist somit stärker denn je. Um dieses Ziel zu erreichen, ist eine geeignete, schlagkräftige Kombination von Mikrocontroller und Software nötig.**

Betrachten wir zunächst noch einmal den Aufbau des TCP/IP-Protokollstapels (Bild 13). Die Aufgabe der Datenverarbeitung und Übermittlung auf der untersten Schicht (network layer) wird üblicherweise von einem Ethernet-Controller übernommen. Dieser implementiert die Protokolle nach IEEE 802.3 und stellt somit die Schnittstelle zwischen einem Mikroprozessor bzw. Mikrocontroller und dem Übertragungsmedium dar. Die Bearbeitung der übergeordneten Protokollschichten erfolgt zweckmäßigerweise durch Einsatz eines Mikrocontrollers. Die dort vorhandenen, Chip-internen Ressourcen an Arbeits- und Festwertspeicher führen zu einem minimalen Aufwand an äußerer elektrischer Beschaltung.

## Das Konzept

Der sicherlich interessanteste notwendige Teil ist die Entwicklung einer geeigneten Mikrocontroller-Software. Dazu müssen die im vorangegangenen Abschnitt »Internet-Protokolle – Die Grundlagen« beschriebenen Protokolle soweit abgerüstet werden, dass sie einerseits die unbedingt für die Kommunikation erforderliche Funktionalität bereitstellen, andererseits aber auch sehr sparsam im Umgang mit Speicher und Rechenzeit sind.

### *Wahl eines Mikrocontrollers*

Bei der Auswahl eines geeigneten Bausteins fiel die Wahl auf die MSP430-Serie von Texas Instruments (Blockschalt-

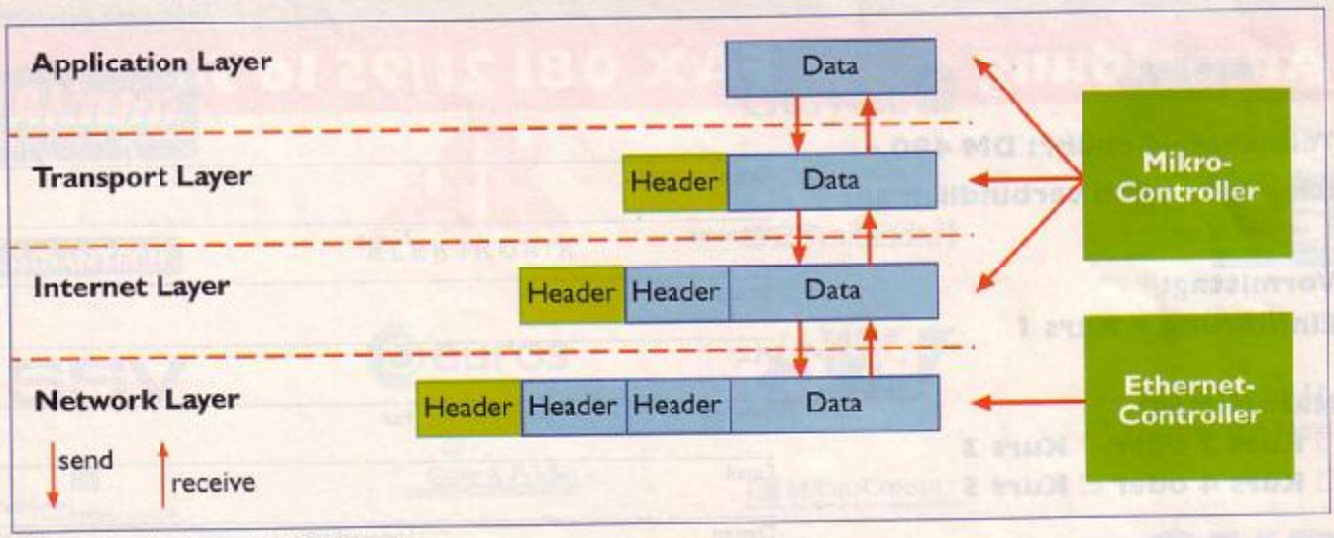



**Bild 13: Handhabung des TCP-Stapels**

bild siehe Bild 14) als Vertreter einer sehr leistungsfähigen Reihe von 16-Bit-Mikrocontrollern. Zu deren herausragendsten Eigenschaften zählen die extrem niedrige Stromaufnahme, die RISC-Architektur (Reduced Instruction Set Code) mit einer Befehlszykluszeit von 125 ns (bei Maximaltakt), der 12-Bit-A/D-Wandler und die vielfältigen Konfigurationsmöglichkeiten der integrierten Timer und Counter. Die MSP430-Serie eignet sich somit optimal für eine Programmierung in der Hochsprache C.

Ein weitere, für die Entwicklung von Projekten sehr wichtige Funktion ist die einfache In-Circuit-Programmierbarkeit der Flash-Modelle mittels eines Target-Interface (Flash Emulation Tool, FET). Das Interface wird dazu transparent in den Source-Level-Debugger »C-Spy« der integrierten Entwicklungsumgebung »Embedded Workbench« von IAR Systems eingebunden. Mit diesem Paket aus Hard- und Software ist eine sehr effiziente Software-Entwicklung möglich.

Die verschiedenen Derivate der MSP430-Serie bieten eine gute Skalierbarkeit von Preis und Leistung entsprechend der geforderten Aufgabe. In diesem Projekt kommt der Typ »MSP430F149« zum Einsatz, welcher neben den eben dargestellten Eigenschaften über einen Flash-Programmspeicher von 60 KByte und einen Arbeitsspeicher von 2 KByte Größe verfügt [MSP430DS].

### Wahl eines Ethernet-Controllers

Bei der Auswahl eines Ethernet-Controllers sind verschiedene Gesichtspunkte zu beachten. Zum einen sind bestimmte Forderungen durch den Einsatz des MSP430F149 bedingt, zum anderen stellt die Applikation einige Ansprüche an die Hardware. Für den Einsatz in einem Embedded-Internetknoten ist beispielsweise ein hoher Integrationsgrad im Baustein selbst (sowohl analog als auch digital), das Vorhandensein von FIFO-Puffern zur automatischen Zwischenspeicherung von Ethernet-Frames, Möglichkeiten des Stromsparens und ein einfach zu handhabendes Bus-Interface wünschenswert.

Die am Markt befindlichen Ethernet-Controller sind fast ausschließlich für den Einsatz auf PC-Netzwerkkarten gedacht und dementsprechend meistens mit einem PCI-Bus-Interface ausgestattet. Eine der wenigen Ausnahmen stellt der »CS8900A« (Blockschaltbild siehe Bild 15 auf Seite 25) des Herstellers Crystal (Cirrus Logic) dar. Dieser Netzwerkkarten-Controller ist zwar zum Anschluss an einen PC-ISA-Bus vorbereitet, doch lässt sich dieses Bus-Interface (im Gegensatz zu PCI) sehr einfach mit allen gängigen Mikrocontrollersystemen verbinden. Die Lieferbarkeit dieses Bausteins in einer 3,3-V-Variante ermöglicht außerdem einen direkten Anschluss an den MSP430F149 [Crystal].

Um die zu entwerfende Baugruppe universell einsetzen zu können, d.h. nicht nur in der später beschriebenen Anwendung als Webserver, erschien es noch sinnvoll, ein RS232-Interface vorzusehen und alle unbenutzten Portpins über Steckverbinder zur Verfügung zu stellen.

## Die Hardware

Der folgende Abschnitt stellt die Schaltung der Webserver-Baugruppe vor. Diese besteht im wesentlichen aus den gerade besprochenen Ethernet- bzw. Mikrocontrollern sowie deren zugehöriger externer Beschaltung. Zur Übersicht dient das Blockschaltbild der Baugruppe in Bild 16 auf Seite 26. Der Stromlaufplan der Schaltung sowie ein Plot sämtlicher Lagen der vierlagigen Multilayer-Platine inklusive Bestückungsplan und Bauteil-Stückliste sind im Anhang ab Seite 45 zu finden. Sämtliche Angaben von Signalnetz-Namen und Bausteinbezeichungen beziehen sich auf diese Dokumente.

Die wichtigste zu lösende Aufgabe war die Kopplung von MCU (*IC1*) und Ethernet-Controller (*IC2*). Da der MSP430F149 über kein externes Bus-Interface verfügt, muss das zur Adressierung und Datenübertragung notwendige Taktschema softwaremäßig durch Benutzung von Portpins bereitgestellt werden. Der Ethernet-Controller lässt sich in drei verschiedenen Betriebsarten betreiben: im »Memory«- und »I/O«-Modus sowie als »DMA-Slave«. Dabei fiel die Wahl auf den Einsatz des I/O-Modus, da sich der CS8900A in diesem Fall mit einer minimalen Anzahl vor Signalleitungen an den Mikrocontroller anschließen lässt. Im Detail sind nur ein 8 Bit breiter, bidirektionaler Datenbus, ein 4 Bit breiter Adressbus und ein aus zwei Signalen bestehender Steuerbus notwendig. Dabei kann über das low-aktive Signal »IOR« (MCU-Pin P3.6) ein Lese- und

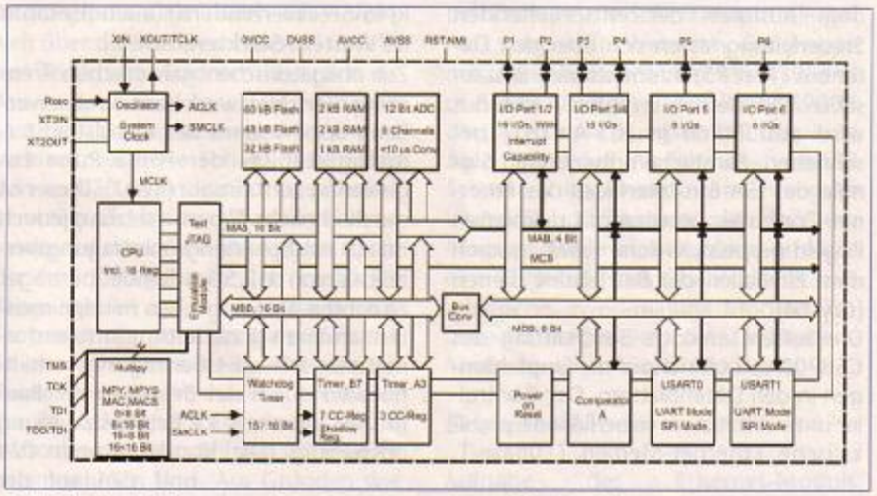



**Bild 14 Block-Diagramm der Low-Power-Controllerfamilie MSP430x14x von Texas Instruments**

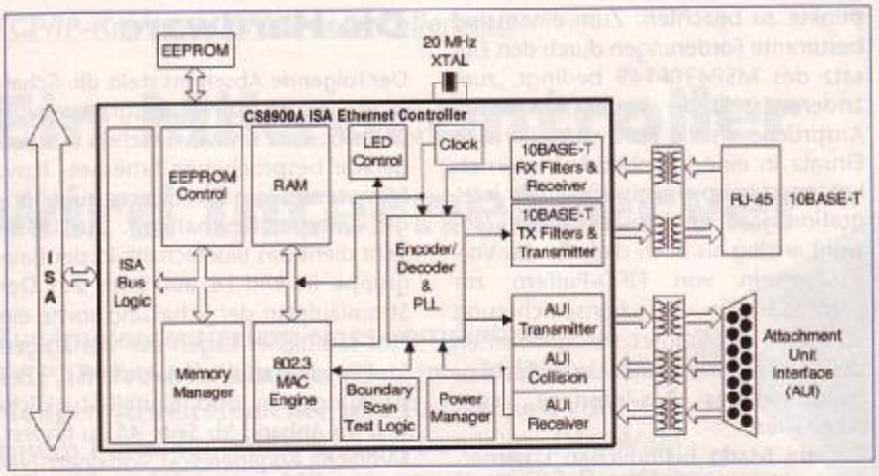



**Bild 15: Block-Diagramm des Ethernet-Controllers CS8900A von Crystal**

über das ebenfalls low-aktive Signal IOW (Pin P3.7) ein Schreibvorgang ausgelöst werden. Nach dem Anlegen der gewünschten I/O-Adresse (0 bis 15) an den Adressbus (Pins P3.0 bis P3.3) und dem Auslösen der entsprechenden Steuerleitung lassen sich über den Datenbus (Port P5) Informationen austauschen. Der Betrieb im 8-Bit-I/O-Modus wird ausführlich in [CS-AN181] beschrieben. Sämtliche unbenutzten Signale des ISA-Bus-Interfaces des Ethernet-Controllers werden mit definierten Pegeln gespeist, welche teilweise auch dem Einstellen der Betriebsart dienen [Crystal].

Die äußere analoge Beschaltung des CS8900A stützt sich auf die Empfehlungen in den Datenblättern. Der Controller unterstützt zwar verschiedene physikalische Ethernet-Medien (10Base-T, 10Base-2, AUI), jedoch beschränkt sich diese Realisierung auf das heute am häufigsten eingesetzte »10Base-T«. Die Verbindung zum Hub erfolgt dabei mittels standardisiertem Patchkabel mit RJ45-Steckverbindern (auch bekannt als Western-Steckverbinder).

Zur obligatorischen galvanischen Trennung von Netzwerk und Webserver-Baugruppe kommt der Impulstransformator »E2023« der Firma Pulse Engineering zum Einsatz (*IND1*). Dieser ist durch ähnliche Typen ersetzbar, jedoch ist auf ein Spannungsübertragungsverhältnis von 1:2,5 im Sendeübertrager zu achten. Das verglichen mit den meisten anderen Ethernet-Impulstransformatoren höhere Übertragungsverhältnis wird durch das Betreiben der Baugruppe mit nur 3,3 V Betriebsspannung notwendig. Die Kondensatoren *C24* und *C25* auf der Netzwerkseite des Transformators können bei Verwendung einer geschirmten RJ-45-Buchse *X2* bestückt werden. Um in diesem Fall eine Verbindung zwischen Masse-Potential *GNDA* und Netzwerkkabelschirm bzw. metallischem Steckergehäuse herstellen zu können, wurde der Lötstopp-Lack in der Umgebung der Befestigungsbohrung neben der RJ-45-Buchse ausgespart.

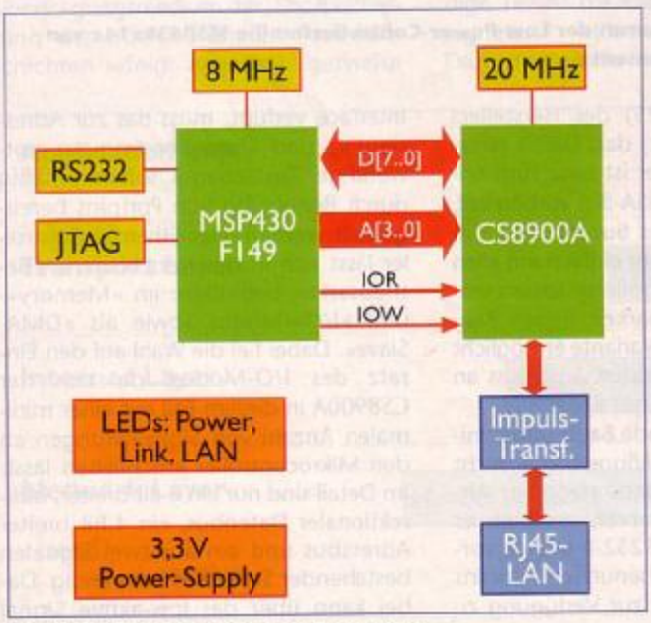



**Bild 16: Blockschaltbild der Baugruppe**

Die Erzeugung des für die Funktion des LAN-Controllers notwendigen Taktes übernimmt ein 20-MHz-Schwingquarz (*Q1*). Das beim Anlegen der Versorgungsspannung nötige Reset-Signal wird durch die R/C-Kombination *R9/C17* bereitgestellt. Dabei ist zu beachten, dass der für den Anschluss an den ISA-Bus gedachte CS8900A im Gegensatz zum MSP430F149 ein high-aktives Resetsignal benötigt [CS-AN83] [Pulse].

Der Ethernet-Controller besitzt verschiedene Anschlüsse mit erhöhter Treiberfähigkeit. An zwei dieser Anschlüsse sind Leuchtdioden zur Darstellung des Netzwerkstatus angeschlossen. Pin 100 (*LANLED*) treibt eine rote Leuchtdiode (*D1*) zur Anzeige von Netzwerkaktivität (Daten werden durch den CS8900A empfangen bzw. gesendet), eine gelbe LED (*D2*) an Pin 99 (*LINKLED*) dient hingegen der Signalisierung, dass zwischen dem Netzwerkcontroller und einem Hub eine Verbindung hergestellt wurde.

Die äußere Beschaltung des Mikrocontrollers MSP430F149 umfasst neben der bereits beschrieben Anbindung des LAN-Controllers ein JTAG-Interface, eine RS232-Schnittstelle, einen Schwingquarz sowie eine Reset-Schaltung [MSP430DS].

Für das In-Circuit-Debugging und die Programmierung der MCU nutzt die Schaltung das JTAG-Interface, dessen Signale (*TCLK, TCK, TDI, TDO/TDI* und *TMS*) über einen 14-poligen Pfostensteckverbinder (*X6*) zugänglich sind. Dieser Steckverbinder ist für den direkten Anschluss des von Texas Instruments erhältlichen »MSP430 Flash Emulation Tool« vorgesehen.

Der Pegelwandler »ADM3202ARU« von Analog Devices (*IC3*) erzeugt die genormten Pegel des RS232-Interface. Als zusätzliche Bauteile zum Chip, der sich mit einer Versorgungsspannung von 3,3 V begnügt, sind lediglich vier Kondensatoren mit einer Kapazität von je 100 nF nötig. Die Signale *RxD* und *TxD* der RS232-Schnittstelle sind über einen 10-poligen Pfostensteckverbinder (*X5*) zugänglich. Dieser ist so beschaltet, dass eine direkte Verbindung zu einer 9-poligen seriellen Schnittstelle eines PCs möglich ist. Dazu ist ein einfaches, 1:1 durchgeschaltetes Kabel notwendig, welches sich beispielsweise sehr effizient in Schneid-Klemm-Technik anfertigen lässt [ADM3202].

Um die maximal mögliche Leistung aus der eingesetzten MCU herauszuholen,