

Diplomarbeit

zur Erlangung des akademischen Grades

Diplom-Ingenieur

Entwicklung einer Internetkamera und deren Anbindung über PPP und TCP/IP an das ISDN

angefertigt von

cand. Ing. Thorsten Thormählen

bei

Prof. Dr. rer. nat. G. Zimmer

Fachgebiet

Elektronische Bauelemente und Schaltungen

an der

Gerhard-Mercator-Universität-Gesamthochschule Duisburg

Duisburg, Dezember 1999

Vorwort des Verfassers

Diese Diplomarbeit entstand im Rahmen des Eigenforschungsprojektes „Bildsensorvernetzung“ der Abteilung System- und Anwendungstechnik im Fraunhofer Institut für Mikroelektronische Schaltungen und Systeme (IMS) in Duisburg in der Zeit vom 14. Juni bis zum 14. Dezember 1999. Institutsleiter ist Herr Prof. Dr. G. Zimmer, dem ich für die Übertragung dieser Diplomarbeit danke.

Im IMS in Duisburg konnte ich die Arbeit unter sehr guten technischen Voraussetzungen und angenehmer Arbeitsatmosphäre erstellen. Besonders bedanken möchte ich mich bei meinen beiden Betreuern, den Herren Dipl.-Ing. M. Schäfer und Dipl.-Ing. H. J. Schliepkorte, für ihre wertvollen Hinweise und konstruktive Kritik.

Duisburg, den 14. Dezember 1999
Thorsten Thormählen

Inhaltsverzeichnis

1	Einleitung	12
2	Grundlagen	14
2.1	OSI-Referenzmodell	15
2.1.1	Überblick	15
2.1.2	Schichten des OSI-Referenzmodells	16
2.1.3	Vertikale Kommunikation	16
2.1.4	Horizontale Kommunikation	17
2.1.5	Vor- und Nachteile des OSI-Modells	18
2.2	Kommunikationsprotokolle	20
2.2.1	Überblick	20
2.2.2	PPP	20
2.2.3	TCP/IP	25
2.2.4	Sockets	35
2.2.5	HTTP	36
2.3	ISDN	38
2.3.1	ISDN Basisanschluss	38
2.3.2	S ₀ -Bus	39
2.3.3	ISDN-Protokollarchitektur im OSI-Schichtenmodell	39
2.4	JPEG-Standard	40
2.4.1	Sequenzieller DCT-Modus	42
2.4.2	Progressiver DCT-Modus	46
2.4.3	Hierarchisch-progressiver Modus	46
2.4.4	Verlustfreier Modus	46
2.5	VxWorks	47
2.5.1	Task	47
2.5.2	Interrupts	48
2.5.3	Kommunikation zwischen Tasks	48
3	Konzeption und Realisierung der Internetkamera	51
3.1	Einführung	52
3.2	Auslesen der Bilddaten aus dem SRAM	54
3.2.1	Erweiterung des Verilog-Programms	55
3.2.2	FPGA-Steuerung durch den MPC860	56
3.2.3	Der Memory-Controller	60
3.2.4	Die Register des MPC860	60
3.2.5	Speicherorganisation des MBX-Boards	60

3.2.6	Programmieren des Memory-Controllers	62
3.3	Bilddatenkompressor	62
3.3.1	JPEG-Komprimierer	65
3.4	Internetserver	66
3.4.1	Synchronisation und Geschwindigkeitsoptimierung	67
3.4.2	Speichermanagement	67
3.5	Kommunikationsprotokolle	71
3.5.1	PPP	72
3.5.2	TCP/IP über die Socket-Schnittstelle	72
3.6	ISDN	73
3.6.1	Bitübertragungsschicht	73
3.6.2	HDLC im D-Kanal mit SCC und TSA	77
3.6.3	D-Kanal Protokolle	81
3.6.4	Anbindung der D-Kanal Protokolle und PPP an die Bitübertragungsschicht	81
3.6.5	Bewertung der ISDN-Schnittstelle der Internetkamera	85
4	Verifikation des Systems	86
4.1	Visualisierung in einem Internetbrowser	87
4.1.1	HTML	87
4.1.2	META-Tag	88
4.1.3	JavaScript	89
4.1.4	Java-Applet	91
4.2	Inbetriebnahme der Internetkamera und Aufbau der Verbindung	91
4.2.1	Starten der Kameraanwendung	91
4.2.2	Internet oder Ethernet	92
4.2.3	Serielle Schnittstelle	93
4.2.4	ISDN	93
5	Zusammenfassung und Ausblick	96
A	Compact Disk zur Diplomarbeit	102

Abbildungsverzeichnis

2.1	Schichtendarstellung des OSI-Referenzmodells	15
2.2	Dienststruktur des OSI-Modells am Beispiel des Basisdienstes „Transaktion“	17
2.3	Transportmechanismus für Nutz- und Steuerinformationen zwischen Partnerinstanzen	18
2.4	Kommunikationsabläufe als Zeitfolgediagramm	19
2.5	HTTP, TCP/IP, PPP im OSI-Referenzmodell	20
2.6	Allgemeine Struktur der PPP-Kapselung	21
2.7	Phasendiagramm einer PPP-Verbindung	22
2.8	HDLC-Rahmenformat	23
2.9	PPP-Rahmenformat	23
2.10	TCP/IP-Referenzmodell	25
2.11	IPv4-Protokollkopf	27
2.12	Internet-Adresstypen	28
2.13	Beispiel eines Netzwerks - Verbund mehrerer TCP/IP-Netze mittels Router	29
2.14	Fragmentierung eines IP-Pakets	29
2.15	TCP-Protokollkopf	30
2.16	TCP-Verbindungsmanagement als Zustandsautomat	33
2.17	Schematische Darstellung des Sliding-Window-Prinzips als Zeitfolgediagramm	34
2.18	Beispiel einer ISDN-Hausinstallation	39
2.19	Schichtenmodell der ISDN-Protokolle	40
2.20	JPEG-Kodierer im sequenziellen DCT-Modus	42
2.21	JPEG-Dekodierer	42
2.22	Auswertungsmuster zum Umsortieren der zweidimensionalen DCT-Koeffizienten	45
2.23	JPEG-Kodierer im verlustfreien Modus	46
2.24	Taskzustände und Übergänge	47
2.25	Kommunikation mittels Message Queues	49
3.1	Die verwendete Hardware der Internetkamera	52
3.2	Konzept des Systems „Internetkamera“	53
3.3	Datentransport vom SRAM in den Hauptspeicher des MBX- Boards	54
3.4	Auslesen des Speichers im Doppelpufferbetrieb	58
3.5	Auslesen des Speichers bei Verwendung einer Speicherbank	59
3.6	Entfernen des Flash und Umstecken des Jumpers J4	61
3.7	Timing Diagramm für Lese- und Schreibzyklus des MPC860	63
3.8	Taskarbeitung ohne Timeout	68
3.9	Taskarbeitung mit Timeout	68
3.10	Speicherorganisation der einzelnen Module	69
3.11	Speicherorganisation der einzelnen Module mit Doppelpufferbetrieb	70

3.12	Aufgabenverteilung bei der Implementierung der Kommunikationsprotokolle	71
3.13	D-Kanal- und Kommunikationsprotokolle bei der ISDN-Schnittstelle	74
3.14	Der MC145574 Schicht-1 Baustein und das Communication Prozessor Module des MPC860 .	75
3.15	Timing des 8-Bit-IDL	76
3.16	Speicherstruktur der SCCs	79
3.17	Schnittstellenfunktionen zwischen LAPD bzw. PPP und Bitübertragungsschicht	82
3.18	Aufbau der Schnittstelle zur ISDN-Datenübertragung zw. LAPD und Bitübertragungsschicht .	83
3.19	Aufbau der Schnittstelle zur ISDN-Datenübertragung zwischen PPP und Bitübertragungsschicht	84
4.1	Visualisierung des Kamerabildes in einem Internetbrowser	87
4.2	Periodisches Aktualisieren des Kamerabildes mit Hilfe eines Java-Applets	90
4.3	Java-Applet der Internetkamera	92
4.4	Aufbau einer TCP/IP Verbindung mit Internetbrowser als Client	93
4.5	Aufbau einer seriellen Verbindung über PPP und TCP/IP unter Windows NT 4.0	94
4.6	Statusreport der seriellen Verbindung über PPP und TCP/IP unter Windows NT 4.0	95
4.7	Aufbau einer ISDN-Verbindung zur Internetkamera über PPP und TCP/IP	95
A.1	Verzeichnisstruktur der CD	103

Tabellenverzeichnis

2.1	Flags im TCP-Protokollkopf	31
2.2	Analogie zwischen dem Nutzen einer Socket-Schnittstelle und Telefonieren	36
2.3	Anfragemethoden des HTTP	37
2.4	Statusklassen in der HTTP-Antwort	38
2.5	Prädiktoren für den verlustfreien Modus	47
3.1	Verwendete Signale beim Auslesen des SRAMs	55
3.2	Der Aufbau des 32-Bit-Registers	56
3.3	Verwendete Chip-Select-Leitungen	61
3.4	Unterstützte Grafikformate der Generationen von Internetbrowsern	65
3.5	SI-RAM Einträge für 8-Bit-IDL	76
3.6	Zuordnungstabelle des Parameter-RAMs	78
3.7	IDL-Signale auf dem 860/COMM Expansion Connector	79

Abkürzungsverzeichnis

AHDLC	Asynchronous High-Level Data Link Control
ARP	Address Resolution Protocol
ARPANET	Advanced Research Projects Agency NETWORK
AUTH	Authentifizierungsprotokoll
BC	B-Channel
BD	Buffer Descriptor
BR	Base Register
BSD	Berkeley System Distribution
BSP	Board Support Package
Cache	spezieller temporärer Zwischenspeicher für schnelle Datenzugriffe
CCITT	Comité Consultatif International
CGI	Common Gateway Interface
CIF	CALTECH Intermediate Format
CMOS	Complementary Metal-Oxide Semiconductor
CPM	Communication Processor Module
CRC	Cyclic Redundancy Check
CS	Chip-Select
DCT	Discrete Cosinus Transformation
DFÜ	Daten Fern Übertragung
DIVO	Digitale Ortsvermittlung
DNS	Domain Name Service
DRAM	Dynamic Random Access Memory
DSP	Digital Signal Processor
DSS1	Digital Subscriber Signalling System No.1
EAP	Extensible Authentication Protocol
EIA	Electronic Industries Association
EPROM	Erasable Programmable Read-Only Memory
FhG	Fraunhofer Gesellschaft
FPGA	Field Programmable Gate Array
FTP	File Transfer Protocol
GCI	General Circuit Interface
GIF	Graphics Transfer Protocol
GiK	Gesellschaft für innovative Kommunikationstechnik mbH
GPCM	General-Purpose Chip-Select
H.261	ITU-T H.261

H.320	ITU-T H.320
HDLC	High-Level Data Link Control
HTML	HyperText Markup Language
HTTP	HyperText Transfer Protocol
IAE	ISDN-Anschluss-Einheit
ID	Identifikation
IDL	Interchip Digital Link
IDCT	Inverse Discrete Cosinus Transformation
IDMA	Independent Direct Memory Access
IDU	Information Data Unit
IMM	Internal Memory Map
IMMR	Internal Memory Map Register
IMS	Institut für Mikroelektronische Schaltungen und Systeme
INT	Interrupt
I/O	Input/Output
IOM-2	ISDN-Oriented Modular rev 2.2
IP	Internet Protocol
IPCP	Internet Protocol Control Protocol
IPX	Internet Exchange Package
ISDN	Integrated Services Digital Network
ISO	International Standardization Organization
ISR	Interrupt Service Routine
ITU	International Telecommunication Union
JPEG	Joint Photographic Experts Group
LAN	Local Area Network
LAPD	Link Access Protocol of the D-Channel
LCP	Link Control Protocol
LSB	Least Significant Bit
LZW	Lemple, Welch, Zic (Namen der Entwickler des Komprimierungsalgorithmus von Unisys)
MBX-Board	Mikroprozessorboard für eingebettete Systeme der Firma Motorola
MC145574	Schicht 1 Baustein von Motorola
MF	More Fragments Flag
MPC860	Motorola PowerPC 860
MPEG	Moving Experts Group
MS	Microsoft (US-amerikanischer Softwarehersteller)
MSB	Most Significant Bit
NCP	Network Control Protocol
NT	Network Termination
NVRAM	Non-Volatile Random Access Memory
OR	Option Register
OSI	Open System Interconnection
PAP	Password Authentication Protocol
PC	Personal Computer
PCI	Protocol Control Information

PCM	Pulse Code Modulation
PDU	Protocol Data Unit
PNG	Portable Network Graphics Format
PPP	Point-to-Point-Protocol
QUICC	Quad Integrated Communication Controller
Reg	Register
RFC	Request For Comment
RGB	Rot, Grün, Blau
ROM	Read-Only Memory
RS232	Related EIA Standard (entspricht V-24-Schnittstelle)
SAP	Service Access Point
SATNET	SATellite NETwork
SCC	Serial Communication Controller
SCP	Serial Control Port
SDRAM	Synchronous Dynamic Random-Access Memory
SDU	Service Data Unit
SI	Serial Interface
SMC	Serial Management Controller
SMTP	Simple Mail Transfer Protocol
SPAP	Shiva Password Authentication Protocol
SPI	Serial Peripheral Interface
SRAM	Static Random Access Memory
TCP	Transmission Control Protocol
TDM	Time Division Multiplexed
TELNET	Telnet Virtual Terminal Protocol
TSA	Time Slot Assigner
UPM	User-Programmable Machine
UNIX	= UNICS Uniplexed Information and Computing System
URL	Uniform Resource Locator
WOSA	Windows Open Service Architecture
ZZF	Zentralamt für Zulassungen im Fernmeldewesen

Glossar englischer Fachbegriffe

Board	Platine, (Steck-) Karte
Boot	Urladen
Buffer	Puffer, Speicher
Bus	Verbindungsleistung, Kanal
Client	Arbeitsplatzrechner, Dienstanutzer, Kunde
Compiler	Übersetzungsprogramm
Controller	Steuereinheit, Steuerwerk
Debug	Programmfehler suchen und beseitigen
Delay	Verzögerung
Delayed	Verzögert
Driver	Treiber
Escape	Funktionsunterbrechung
Execute	Ausführen
Firewall	Datenschutzeinrichtung gegen Angriffe von aussen
Flag	Steuerelement, Anzeiger
Flash	Aufblitzen, Blinken
Frame	Rahmen, Blockaufbau
Gateway	Intelligenter Netzknoten, Kommunikationsserver
Grant	Gewähren
Handshake	Anforderungs-Quittungsverfahren
Header	Anfangskennsatz
Host	Wirtsrechner, Großrechner
Interface	Schnittstelle
Integer	ganze Zahl
Interrupt	Unterbrechnung, Unterbrechen
Jumper	Brücke, spezielle Steckkontakte
Master	Meister
Memory	Speicher
Memory Map	Speichertabelle
Message Queue	Nachrichtenwarteschlange
Offset	Versetzung
Pended	Anstehend
Pin	Steckstift, Stecker
Pixel	Bildelement (engl. Abkürzung für „picture element“)
Pointer	Zeiger

Port	Leitweg
Preemptive	Präventiv
Ready	Fertig
Request	Anfrage
Router	Intelligenter Netzknoten, Wegweiser
Routing	Wegewahl
Schedule	Zeitplan
Scheduling	Terminplanung
Server	Zentralrechner, Dienstleister, Dienstseinheit
Slave	Sklave, abhängiger Rechner
Socket	Schnittstelle, Buchse
Stuffed	Füllung, Polstermaterial
Suspended	Aufgeschoben
Tag	Formatierbefehl, Etikett
Task	Auftrag, Aufgabe
Thread	Auftrag, Aufgabe (ähnlich wie Task)
Timeout	Zeitabschaltung
Timer	Zeitgeber
Timing	Zeitverlauf, Zeitüberwachung

Kapitel 1

Einleitung

Die Kommunikation mit Hilfe des Internetprotokolls TCP/IP über Netzwerke wie ISDN, Ethernet oder das weltweite Internet ist ein alltäglicher Vorgang im modernen Berufs- und Privatleben geworden. Durch die Verwendung von leistungsstarken eingebetteten Systemen können kostengünstige Multimediaanwendungen für diese Netzwerke entstehen.

Gegenstand dieser Diplomarbeit ist eine ISDN-Internetkamera, die als autonomes System operieren kann. Für ihren Betrieb muss kein zusätzlicher Rechner als Server eingesetzt werden. Der eingebaute Mikrocontroller übernimmt die Aufgabe eines Internetserver. Die Bilder der Kamera können über Ethernet, ISDN oder serielle RS232-Schnittstelle versendet werden. Über diese Übertragungsmedien kann vom jedem PC eine Verbindung zur Internetkamera über das Internetprotokoll TCP/IP hergestellt werden. Der Anwender kann die Bilder mit Hilfe eines Internetbrowsers abrufen, welcher heutzutage zur Standardsoftware eines PCs gehört.

Die Übertragung von Live-Bildern von den verschiedensten Lokalitäten der Welt durch Internetkameras ist ein beliebter und viel genutzter Dienst im Internet. Bei der Verwendung des Internets als Übertragungsmedium, kann die Übertragungsrate und damit die Wiederholfrequenz der Kamerabilder stark schwanken. Dies ist vor allem für Überwachungszwecke nachteilig. Mit Hilfe einer PPP-Verbindung über ISDN kann die Internetkamera in ein lokales TCP/IP-Rechnernetz integriert werden. Bei einer direkten ISDN-Verbindung zum Server der Kamera kann dem Benutzer eine feste Datenrate während der gesamten Übertragung garantiert werden. Durch die Anschlußmöglichkeiten über Ethernet und ISDN ist die ISDN-Internetkamera sehr variabel einsetzbar und der Standort frei wählbar.

Für die Kamera wird ein spezieller CMOS-Bildsensor verwendet, der in der Abteilung Signalverarbeitung und Systementwurf des IMS entstanden ist. In der vorangegangenen Studienarbeit mit dem Thema „Konzeption und Realisierung einer Entwicklungsplattform für eine ISDN-Kamera“ ist die Hardware der Internetkamera realisiert worden. Im Rahmen dieser Diplomarbeit wurde die Software für den MPC860 Mikrocontroller von Motorola mit integriertem Kommunikationsprozessor entwickelt. Dieser Mikrocontroller ist bisher im IMS noch nicht eingesetzt worden. Durch den leistungsstarken Controller und die Verwendung des Echtzeitbetriebssystems VxWorks der Firma Wind River stellt die Internetkamera eine solide Plattform für die Entwicklung von weiterführenden Anwendungen in der Automatisierungs- und Kommunikationstechnik dar.

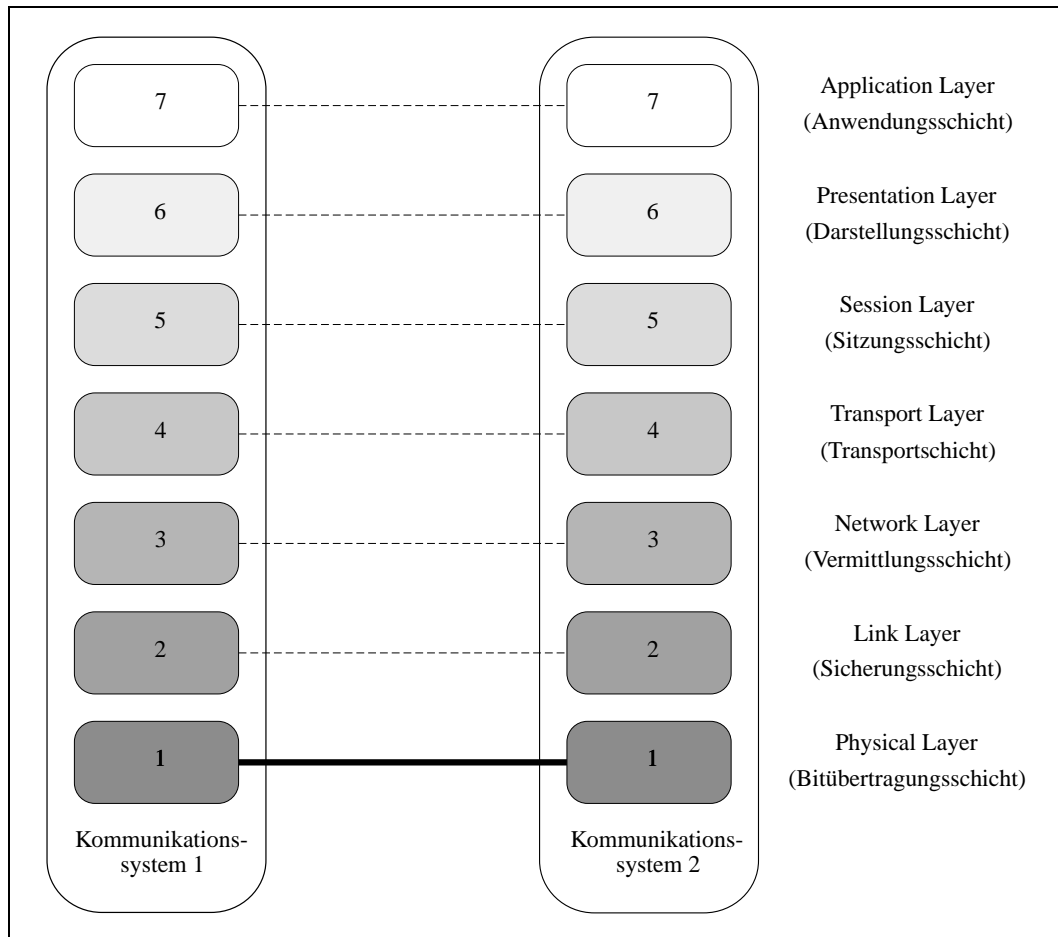
Die hier vorliegende Dokumentation der Arbeit stellt zuerst die erforderlichen Grundlagen zum OSI-Referenzmodell, allen verwendeten Kommunikationsprotokollen, ISDN, JPEG und dem Echtzeitbetriebssystem VxWorks vor. Danach wird die Konzeption und die Realisierung der Internetkamera beschrieben. Anschließend ist der Hard- und Softwaretest dargestellt. Dazu werden die Bilder der Kamera in einem Internetbrowser periodisch visualisiert und verschiedene Verfahren zur Erstellung einer Internetseite mit Kamerabildern verglichen. Abschließend werden die Ergebnisse zusammengefasst und ein Ausblick auf zukünftige Einsatzmöglichkeiten gegeben.

Kapitel 2

Grundlagen

2.1 OSI-Referenzmodell

Abbildung 2.1: Schichtendarstellung des OSI-Referenzmodells



Quelle: frei nach [28], Seite 32

2.1.1 Überblick

Unter einem offenen Kommunikationssystem werden Hard- und Softwareimplementierungen verstanden, die sich an Standards oder de facto Definitionen halten, und damit die freie und leichte Verbindung der Lösungen verschiedener Hersteller erlauben. Das Gegenteil eines offenen Systems ist eine proprietäre oder herstellereigene Implementierung.

Die im Rahmen dieser Diplomarbeit entwickelte Internetkamera ist ein offenes System. Es setzt sich aus mehreren Modulen zusammen und für seine Kommunikation mit der Außenwelt nutzt es verschiedene Protokolle. Um die Rolle und Aufgabe dieser Protokolle im Gesamtsystem zu beschreiben, ist es hilfreich, sie in ein standardisiertes Kommunikationsmodell einzuordnen. Als Kommunikationsmodell wird in dieser Diplomarbeit das OSI¹-Referenzmodell verwendet.

¹OSI = Open Systems Interconnection

Das OSI-Referenzmodell, auch Schichtenmodell genannt, ist im ISO²-Standard 7498-1 und im ITU³-T-Standard X.200 zu finden. Es dient zur Beschreibung von Kommunikationsprozessen in offenen Systemen.

2.1.2 Schichten des OSI-Referenzmodells

Das OSI-Referenzmodell definiert sieben Beschreibungsebenen, die sogenannten Schichten (engl. layer). Diese sind in Abbildung 2.1 zu sehen. Im Folgenden werden die Aufgaben der einzelnen Schichten kurz vorgestellt (Quelle: [7], Seite 6):

1. Die Bitübertragungsschicht stellt eine ungesicherte Systemverbindung zur Übertragung von Bits zur Verfügung.
2. Die Sicherungsschicht realisiert eine fehlerüberwachte, sichere Systemverbindung, das heißt, eine Übertragung von Datenblöcken mit Fehlerprüfzeichen unabhängig vom physikalischen Medium.
3. Die Vermittlungsschicht stellt eine Verbindung zwischen Endsystemen zur Verfügung. In dieser Schicht wird die Wahl des Weges vorgenommen, der möglicherweise über Transitsysteme führt.
4. In der Transportschicht werden beliebige Datenmengen zwischen zwei Endsystemen ausgetauscht, die nicht notwendigerweise direkt verbunden sind.
5. In der Sitzungsschicht werden Hilfsdienste zur Koordinierung einer Kommunikationsbeziehung zur Verfügung gestellt. Dabei wird vor allem ein Synchronisationsmanagement realisiert.
6. Die Darstellungsschicht stellt Mittel zum darstellungsunabhängigen Austausch von Daten, sowie zur Datenkompression und Verschlüsselung zur Verfügung.
7. In der Anwendungsschicht kommunizieren Anwendungen miteinander.

Die Realisierung einer Kommunikationsschicht in einem System wird als Instanz bezeichnet. Allgemein kann zwischen horizontaler und vertikaler Kommunikation der Instanzen unterschieden werden. Horizontale Kommunikation wird über Protokolle, vertikale über die Nutzung von Diensten realisiert.

2.1.3 Vertikale Kommunikation

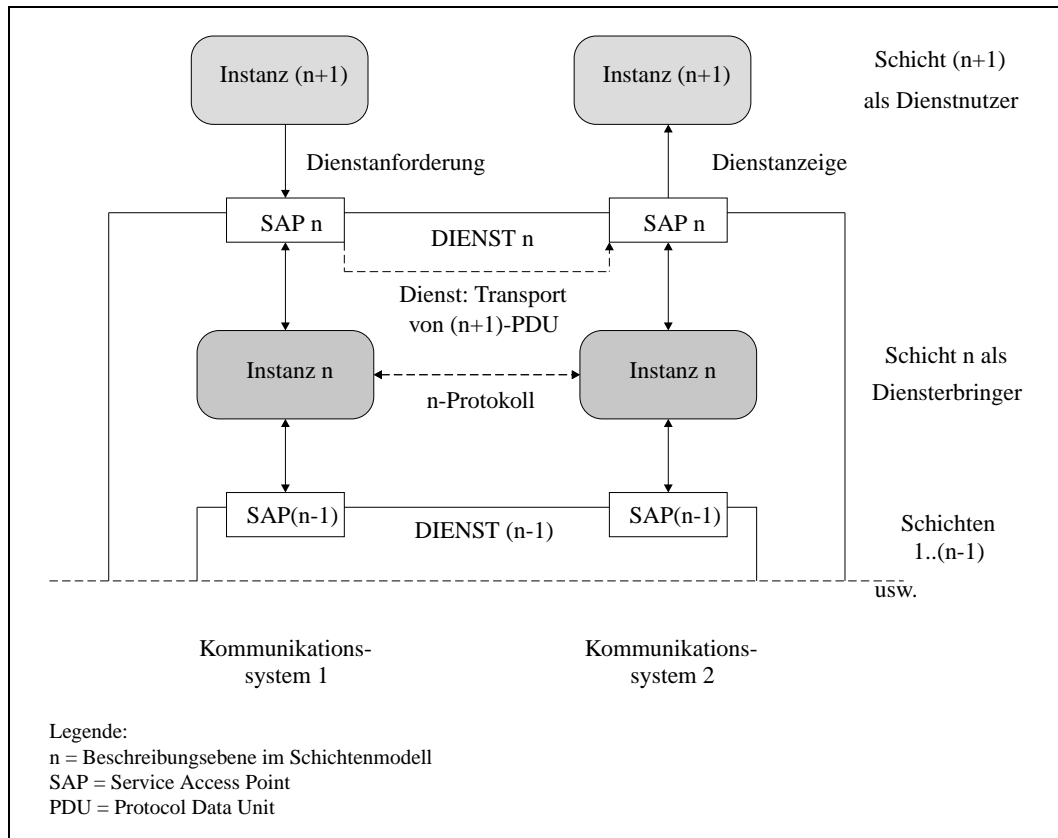
Vertikale Kommunikation ist der Nachrichtenaustausch einer Instanz mit der Instanz aus der nächst höheren oder nächst niedrigeren Schicht in einem Kommunikationssystem. Wie in Abbildung 2.2 gezeigt, geschieht dies über SAPs⁴, d. h. mit Hilfe von Schnittstellen, über die auf einen Dienst zugegriffen werden kann. Die niedrigere Instanz bietet der darüber liegenden einen Dienst (engl. service) an, wobei sie sich selbst nur auf Dienste der nächst niedrigeren Instanz stützt. Dabei ist Schicht n+1 immer der Dienstanwender der Schicht n, dem Dienstanbieter.

²ISO = International Standardization Organization

³ITU = International Telecommunication Union

⁴SAP = Service Access Point

Abbildung 2.2: Dienststruktur des OSI-Modells am Beispiel des Basisdienstes „Transaktion“



Quelle: frei nach [8]

Neben Verbindungsauf- und -abbau besteht eine wichtige Dienstleistung jeder Schicht darin, Nachrichteneinheiten in Form von PDUs⁵, von einem Quell-SAP zum gewünschten Ziel-SAP zu transportieren (siehe Abbildung 2.2). PDUs werden aus den PCIs⁶ und den SDUs⁷ zusammengesetzt. Die SDUs dienen zum Datentransport für die höheren Schichten des Kommunikationssystems.

2.1.4 Horizontale Kommunikation

Über die PCIs entsteht mit Hilfe des in Abbildung 2.3 gezeigten Transportmechanismus ein Nachrichtenaustausch zwischen den Partnerinstanzen der gleichen Kommunikationsschicht. Das wird als horizontale Kommunikation bezeichnet.

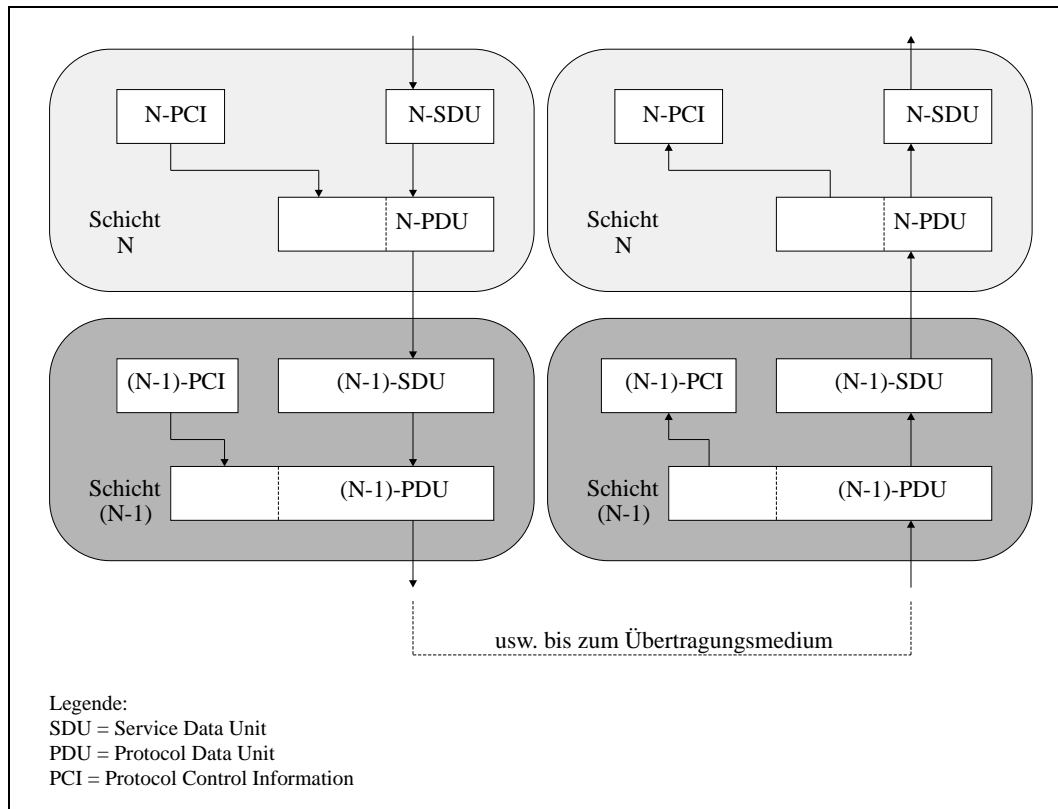
Bei der horizontalen Kommunikation existiert nur zwischen den Bitübertragungsschichten eine physikalische Verbindung. Die anderen Schichten kommunizieren über ihre spezifischen Protokolle mit ihrem Partner. Es entsteht eine virtuelle Verbindung (gestrichelte Linien in Abbildung 2.1 und 2.2). Durch diese virtuelle Verbindung findet eine Abstraktion des Übertragungsmediums von den niedrigeren zu den höheren Schichten statt.

⁵PDU = Protocol Data Unit

⁶PCI = Protocol Control Information

⁷SDU = Service Data Unit

Abbildung 2.3: Transportmechanismus für Nutz- und Steuerinformationen zwischen Partnerinstanzen



Quelle: [28], Seite 18

Bei der horizontalen Kommunikation werden vier Nachrichtentypen zwischen den Instanzen genutzt:

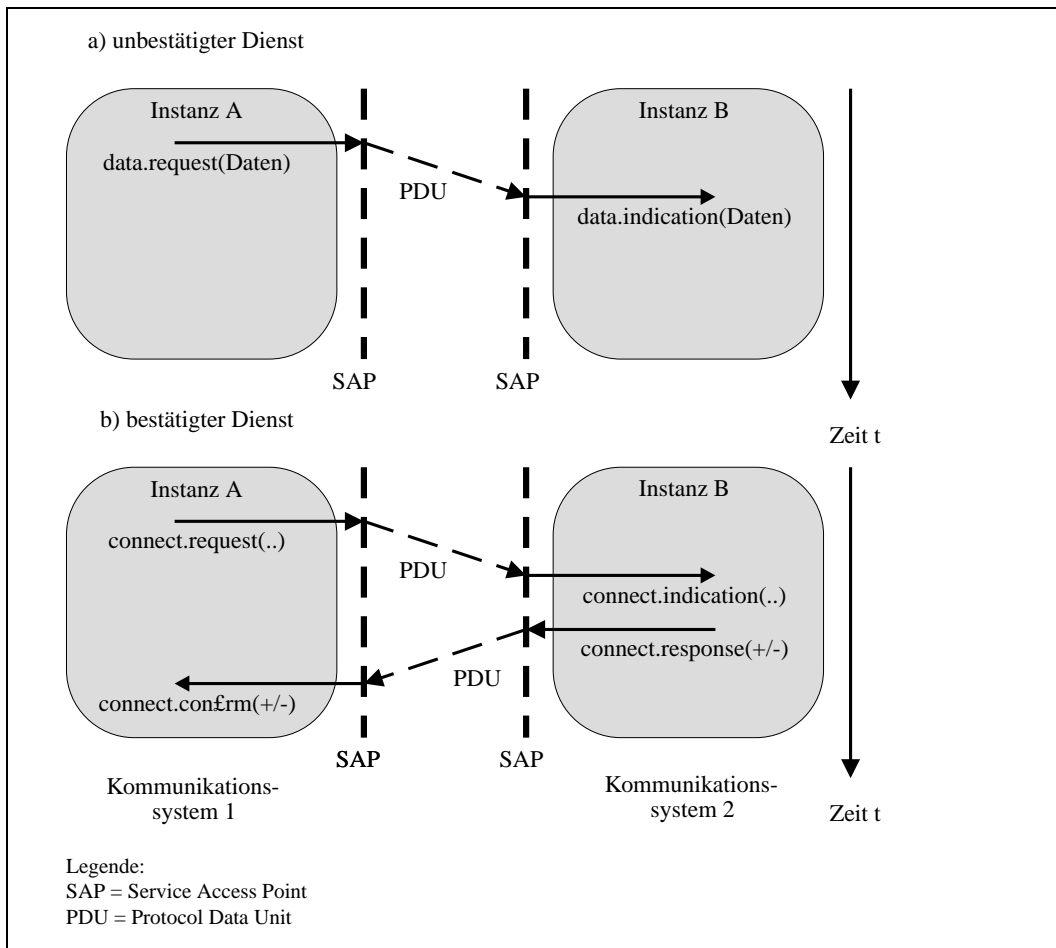
- Dienstanforderung (request)
- Dienstanzeige (indication)
- Dienstbeantwortung (response)
- Dienstbestätigung (confirmation)

Es wird zwischen dem unbestätigten und bestätigten Dienst unterschieden. Ein Zeitfolgediagramm der entsprechenden Kommunikationsabläufe wird in Abbildung 2.4 gezeigt. Ein Dienstanforderung von System 1 führt zu einer Dienstanzeige im System 2. Bei einem betätigten Dienst sendet System 2 daraufhin eine Dienstbeantwortung. Dies führt zu einer Dienstbestätigung im System 1.

2.1.5 Vor- und Nachteile des OSI-Modells

- + Sind die SAPs benachbarter Schichten definiert, kann ein Kommunikationssystem modular aufgebaut werden. Durch ein offenes Kommunikationssystem können die einzelnen Schichten unabhängig von einander realisiert werden.
- + Eine Aufgabenverteilung zwischen Entwicklern wird vereinfacht. Softwareentwickler auf höheren Schichten brauchen keine Hardwarekenntnisse.

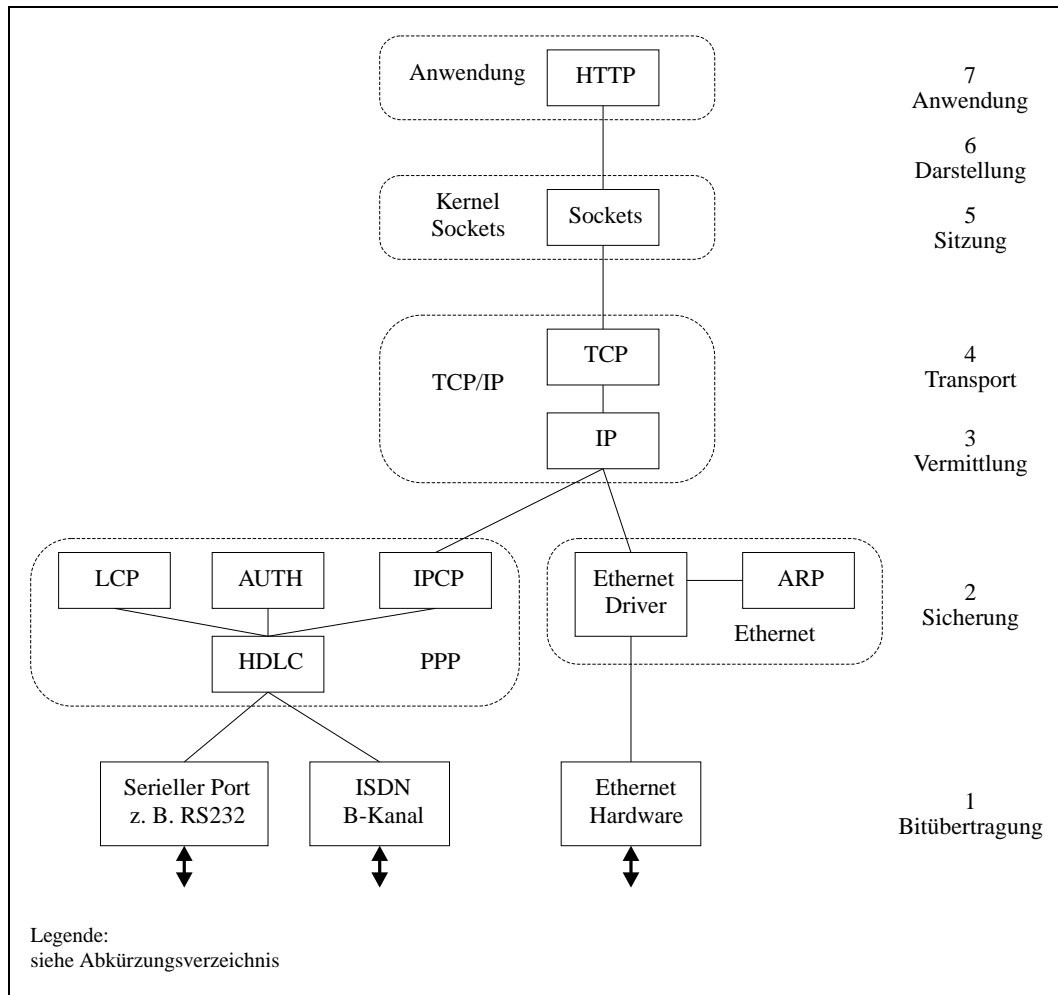
Abbildung 2.4: Kommunikationsabläufe als Zeitfolgediagramm für einen unbestätigten und einen bestätigten Dienst



Quelle: [8], Seite 15

- + Verschiedenartige Protokolle, die für die gleiche Schicht geschrieben sind, können durch einander ersetzt werden. Protokolle von verschiedenen Anbietern können genutzt werden. Es kann so schnell auf Entwicklungsbedürfnisse reagiert werden.
- + Wiederverwertbarkeit der Protokolle in anderen Systemen wird erleichtert.
- + Die Definition von Schnittstellen bildet eine Grundlage zur Verifikation der Schichten und damit zum schnelleren Finden von Fehlern im Gesamtsystem.
- + Möglichkeit zur Abstraktion durch Protokoll- und Dienstbeschreibung.
- Der in Abbildung 2.3 gezeigte Transportmechanismus verursacht einen großen Overhead, weil jedes Protokoll seine Nachrichten den eigentlichen Nutzdaten hinzufügt.
- In der Realität ist die Zuordnung von Protokollen zu einer speziellen Schicht des OSI-Modells nicht notwendigerweise eindeutig.

Abbildung 2.5: HTTP, TCP/IP, PPP im OSI-Referenzmodell



Quelle: [4]

2.2 Kommunikationsprotokolle

2.2.1 Überblick

Abbildung 2.5 ordnet die Protokolle HTTP, TCP/IP und PPP in das OSI-Referenzmodell ein. Die Protokolle TCP, IP und PPP sollen laut Aufgabenstellung dieser Diplomarbeit zum Aufbau eines Kommunikationssystems für eine Internetkamera verwendet werden. Im folgenden Abschnitt werden daher die Grundzüge der verwendeten Kommunikationsprotokolle beschrieben.

2.2.2 PPP

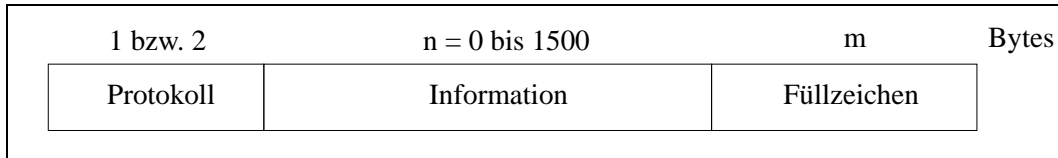
Mit Hilfe des PPP⁸ werden Netzwerkprotokolle höherer Schichten, wie z. B. IP⁹ oder IPX¹⁰ über eine Punkt-zu-Punkt-Verbindung transportiert. Die Spezifikation des PPP, die in RFC1661 [34] zu finden ist, beschreibt

⁸PPP = Point-to-Point-Protocol

⁹IP = Internet Protocol

¹⁰IPX = Internet Packet Exchange

Abbildung 2.6: Allgemeine Struktur der PPP-Kapselung



Quelle: [34], Seite 3

folgende Bereiche:

- Eine Methode zur Verkapselung von Paketen der Protokolle höherer Schichten.
- Die Definition des LCPs¹¹.
- Die Definition einer Familie mit NCPs¹².

2.2.2.1 Verkapselung anderer Protokolle

Abbildung 2.6 zeigt die vom PPP verwendete Kapselung. Damit wird ein Mechanismus bereit gestellt, mit dem verschiedene Protokolle der Vermittlungsschicht gleichzeitig über die selbe Verbindung übertragen werden können.

Der Wert des ersten Feldes gibt Auskunft über das im Informationsblock gekapselte Protokoll. Dabei können Protokolle der Vermittlungsschicht, das verwendete Protokoll des NCP oder Protokolle der Sicherungsschicht (wie z. B. LCP) durch den Wert gekennzeichnet sein. Die aktuelle Zuordnung der Werte zu den Protokollen ist in RFC1340 [31] zu finden. Das Feld Information enthält die eigentlichen Nutzdaten der Protokolle. Durch das letzte Feld kann die Information durch eine beliebige Anzahl von Füllbytes ergänzt werden. Durch den Einsatz von Füllbytes können Datenpakete zur leichteren Weiterbearbeitung auf ein Vielfaches ihrer Grundgröße gebracht werden.

2.2.2.2 Network Control Protocol (NCP)

Das PPP stellt eine Familie von NCPs zur Verfügung. Jedes davon kümmert sich um die spezifischen Aufgabenstellungen eines bestimmten Protokolls aus der Vermittlungsschicht, wie z. B. das Zuordnen und Verwalten von Adressen bei Punkt-zu-Punkt-Verbindungen. Das NCP für das Vermittlungsschichtprotokoll IP heißt IPCP¹³ und wird in RFC1332 [12] beschrieben.

2.2.2.3 Link Control Protocol (LCP)

Das LCP stellt sicher, dass die PPP-Umgebung flexibel genug bleibt, um auf eine Vielzahl verschiedener Plattformen übertragen zu werden. Es enthält folgende Funktionen:

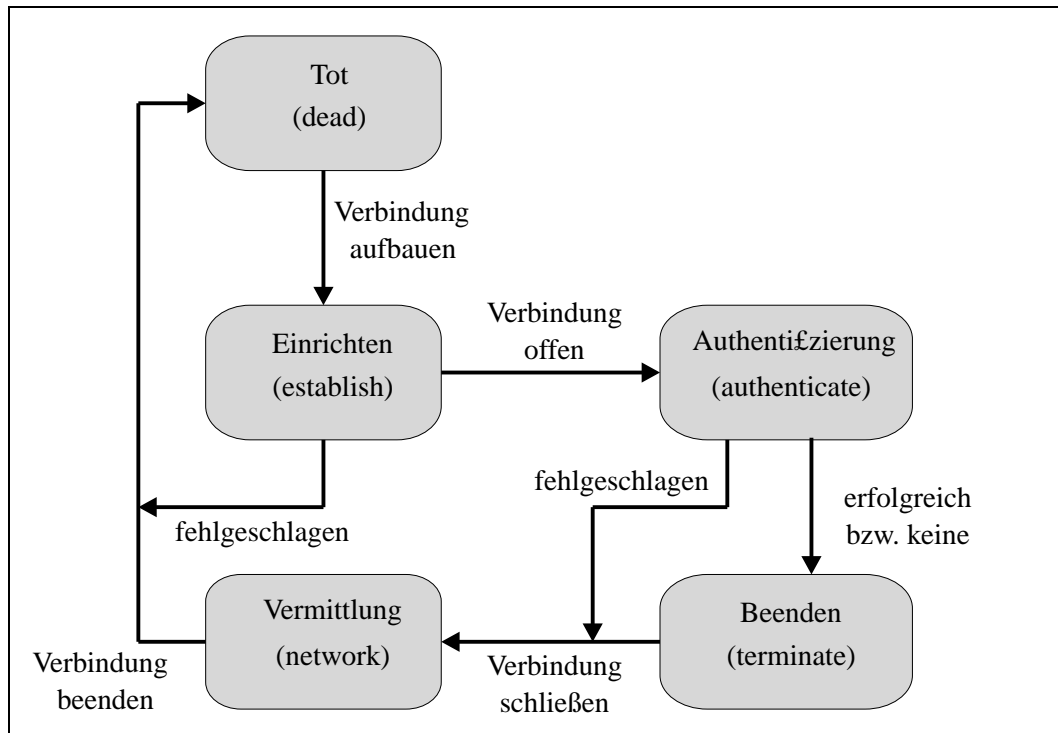
- Verbindungen auf- und abzubauen
- Automatisch das für eine Verkapselung eingesetzte Format aushandeln

¹¹LCP = Link Control Protocol

¹²NCP = Network Control Protocol

¹³IPCP = Internet Protocol Control Protocol

Abbildung 2.7: Phasendiagramm einer PPP-Verbindung



Quelle: [25]

- Verschiedene Maximalgrößen für die versendeten Daten behandeln
- Fehler bei der Konfiguration erkennen
- Authentifizierung eines Hosts am anderen Ende der Verbindung (optional)

2.2.2.4 Aufbau einer PPP-Verbindung

Abbildung 2.7 zeigt das Phasendiagramm für eine PPP-Verbindung.

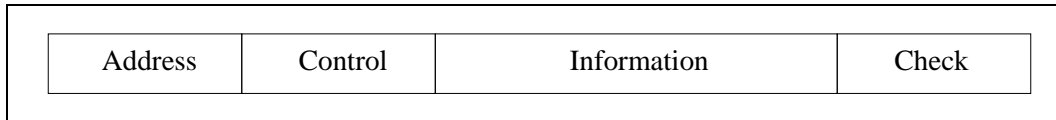
Phase dead: Dies ist der Grundzustand. Die Verbindung beginnt und endet hier. Wenn die Verbindung der Bitübertragungsschicht aufgebaut ist, wird in die nächste Phase gewechselt.

Phase establish: Beide Kommunikationspartner verwenden LCP-Pakete, um die Parameter für die Verbindung zu vereinbaren. Alle Konfigurationen enthalten zuerst die Standardwerte. Durch den Austausch von LCP-Paketen zur Options-Konfiguration können diese geändert werden. Danach wird die Verbindung mit Hilfe von LCP-Paketen aufgebaut und getestet. Alle Nicht-LCP-Pakete werden in dieser Phase verworfen.

Phase authenticate: Standardmäßig wird keine Authentifizierung durchgeführt. Wird sie gewünscht, muss dafür in der vorhergehenden Phase ein Protokoll ausgehandelt werden. Verbreitete Protokolle sind:

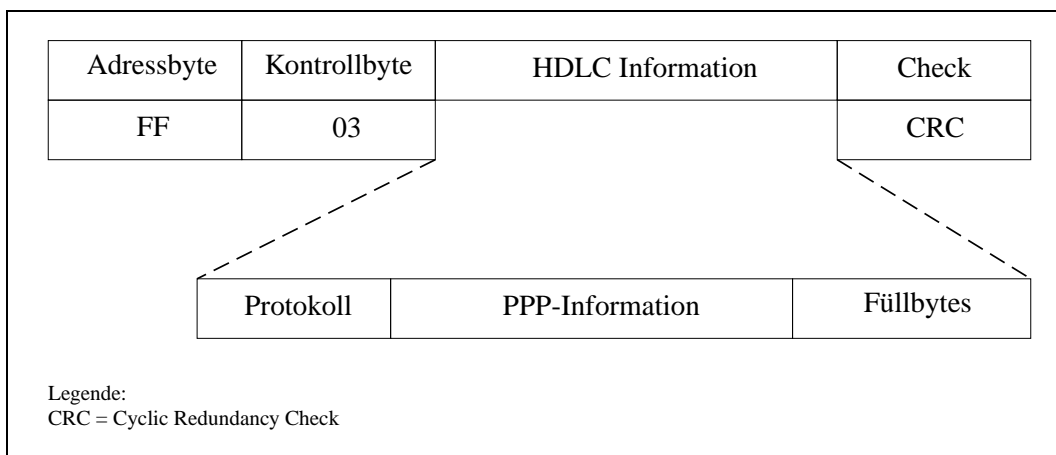
- PAP (Password Authentication Protocol)
- CHAP (Challenge-Handshake Authentication Protocol)

Abbildung 2.8: HDLC-Rahmenformat



Quelle: [4], Seite 10

Abbildung 2.9: PPP-Rahmenformat



Quelle: [4], Seite 12

- MS-CHAP (Microsoft Challenge-Handshake Authentication Protocol)
- EAP (Extensible Authentication Protocol)
- SPAP (Shiva Password Authentication Protocol)

Phase network: Die Vermittlungsschichtprotokolle werden über die zugehörigen NCPs konfiguriert. Ist die Konfiguration abgeschlossen, können Datenpakete aus der Vermittlungsschicht übertragen werden.

Phase terminate: PPP kann die Verbindung jederzeit beenden. Zum Beenden werden entsprechende LCP-Pakete versendet. PPP informiert die Vermittlungsschicht über den Verbindungsabbau. PPP sollte auch die Bitübertragungsschicht zum Trennen der Verbindung auffordern, besonders im Falle einer fehlgeschlagenen Authentifizierung.

2.2.2.5 HDLC im PPP

PPP verwendet eine spezielle Form des HDLC¹⁴-Protokolls, das in RFC1662 [35] beschrieben ist. Dabei wird der normale HDLC-Rahmen, der in Abbildung 2.8 gezeigt wird, folgendermaßen verändert:

- Das Adressfeld enthält immer den Hexadezimalwert FF. (FF bedeutet im HDLC-Protokoll, dass dieser Rahmen für alle Stationen bestimmt ist.)
- Das Kontrollfeld enthält immer 03. (03 steht im HDLC-Protokoll für „nicht nummerierte Information“.)

¹⁴HDLC = High-Level Data Link Control

Damit ergibt sich das Rahmenformat gemäß Abbildung 2.9. Das HDLC-Informationenfeld enthält die gekapselten PPP-Informationen. Die Kapselung der PPP-Informationen wurde bereits in Abbildung 2.6 gezeigt.

2.2.2.6 AHDLC

AHDLC¹⁵ wird für asynchrone Punkt-zu-Punkt-Verbindungen genutzt. AHDLC arbeitet byteweise und nutzt dabei zur Steuerung die beiden Hexadezimalwerte 7E und 7D. Das Byte 7E kennzeichnet den Anfang und das Ende eines AHDLC-Rahmens. 7D ist ein Escape-Zeichen, das zum sogenannten Byte-stuffing verwendet wird. Wenn ein Empfänger auf den Wert 7D im Datenstrom stößt, wird dieser Wert dem Strom entnommen und mit dem darauf folgenden Byte eine Exklusiv-Oder-Operation (XOR) mit dem Hexadezimalwert 20 durchgeführt. Der Sender fügt dieses Escape-Zeichen vor die Werte 00 bis 1F, 7D und 7E ein. So wird zum Beispiel ein 7E im Datenstrom als 7D 5E versendet und 7D als 7D 5D. Auf diese Weise wird eine transparente Datenübermittlung realisiert.

Das folgende Beispiel zeigt einen PPP-Rahmen, auf den der AHDLC-Mechanismus angewendet werden soll:

FF 03 C0 21 01 00 7E 0E 7D 21 31 01

Die Bytefolge auf der seriellen Leitung sieht mit AHDLC folgendermaßen aus:

7E FF 7D 23 C0 21 7D 21 7D 20 7D 5E 7D 2E 7D 5D 21 31 7D 01 7E
--

2.2.2.7 Bit-synchrones HDLC

Bit-synchrones HDLC wird bei den meisten Telekommunikationsschnittstellen, wie z. B. auch ISDN, verwendet. Anders als AHDLC wird es meist durch eine Hardware realisiert. Die Rahmenkennung 7E bleibt gleich. Es wird jedoch kein Escape-Zeichen genutzt, sondern nach jeder Bitfolge von fünf aufeinander folgenden Einsen eine Null eingefügt. Da die Rahmenkennung $7E_{(16)} = 01111110_{(2)}$ sechs aufeinander folgende Einsen besitzt, ist sie jederzeit im Datenstrom zu erkennen. Dieser Mechanismus wird Bit-stuffing genannt.

Als Beispiel dient die Bytefolge von oben:

FF 03 C0 21 01 00 7E 0E 7D 21 31 01

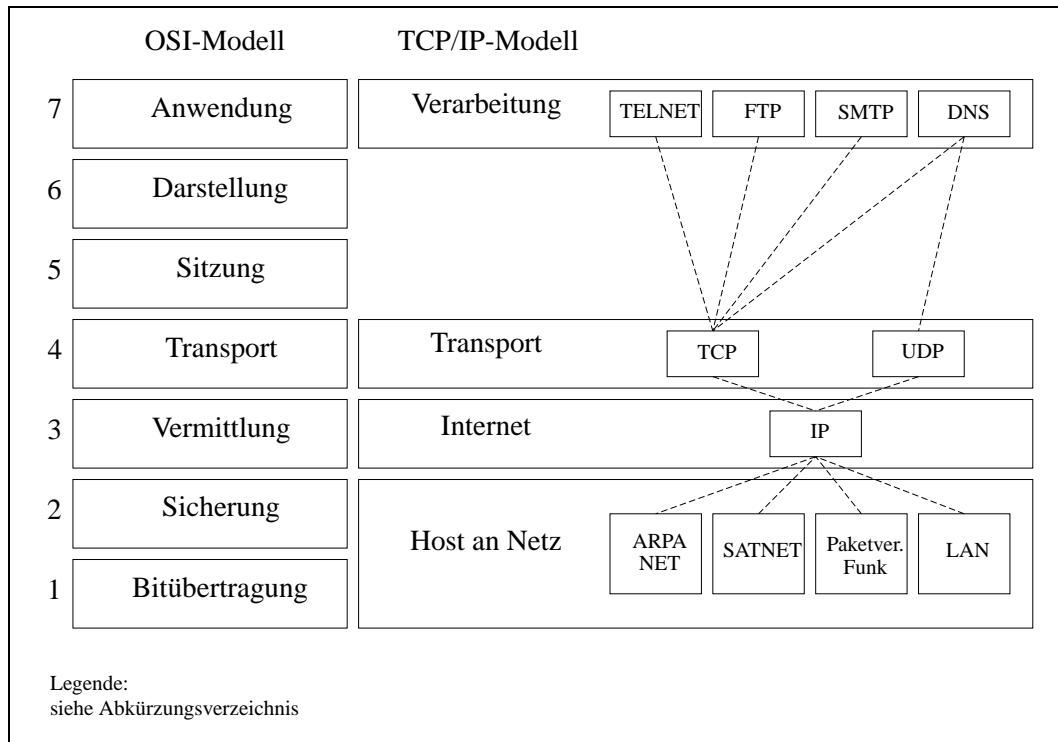
Dargestellt als Bitfolge (HDLC sendet das LSB¹⁶ zuerst):

11111111	11000000	00000011	10000100
10000000	00000000	01111110	01110000
10111110	10000100	10001100	10000000

Nach Bit-synchronem HDLC (S = eingefügte Null):

01111110	11111S11	111S0000	00000000
11100001	00100000	00000000	00011111
S1001110	00010111	110S1000	01001000
11001000	00000111	1110	

Abbildung 2.10: TCP/IP-Referenzmodell



2.2.3 TCP/IP

Im einfachsten Falle kann unter TCP/IP¹⁷ die Verwendung des Protokolls IP in der Vermittlungsschicht und des Protokolls TCP in der Transportschicht eines Kommunikationssystems verstanden werden.

Der Begriff enthält aber eigentlich mehr. Unter TCP/IP wird eine ganze Familie von Kommunikationsprotokollen zusammengefasst. Dabei besitzt TCP/IP eine Architektur, deren Referenzmodell ähnlich dem OSI-Schichtenmodell ist. Abbildung 2.10 zeigt das OSI- und TCP/IP-Modell im Vergleich. Beim TCP/IP-Modell existiert oberhalb der Transportschicht nur noch die Verarbeitungsschicht, während das OSI-Modell noch zwei zusätzliche Schichten definiert. In der Vermittlungsschicht wird im TCP/IP-Modell stets IP verwendet.

In Abbildung 2.10 sind ebenfalls die ursprünglichen Protokolle und Netze des TCP/IP-Modells zu finden. Im Laufe der Jahre wurden viele andere Protokolle der Verarbeitungsschicht hinzugefügt (z. B. auch das HTTP¹⁸, welches in Kapitel 2.2.5 beschrieben wird).

Bei der Definition der TCP/IP-Architektur wurden folgende Ziele verfolgt (Quelle: [32], Seite 27):

- Unabhängigkeit von der verwendeten Netzwerktechnologie sowie der Architektur der Hostrechner

¹⁵AHDLC = Asynchronous High-Level Data Link Control

¹⁶LSB = Least Significant BIT

¹⁷TCP/IP = Transmission Control Protocol/Internet Protocol

¹⁸HTTP = HyperText Transfer Protocol

- Universelle Verbindungsmöglichkeiten im gesamten Netzwerk
- Ende-zu-Ende-Quittungen
- Standardisierte Anwendungsprotokolle

Die TCP/IP-Protokollfamilie hat sich als de facto Marktstandard im Bereich der Kommunikation zwischen Rechnern der verschiedenen Herstellern etabliert. Sie wird heute im weltweiten Internet angewendet.

In den zwei folgenden Abschnitten werden die Protokolle IP und TCP vorgestellt.

2.2.3.1 Internet Protokoll (IP)

IP ist ein verbindungsloses, ungesichertes Protokoll der Vermittlungsschicht. Hauptaufgaben von IP sind die Adressierung von Rechnern und das Fragmentieren bzw. Defragmentieren von Datenpaketen. IP versucht, Pakete so gut wie möglich dem nächsten Rechner im Netzwerk zu übermitteln. Ist dieser nicht der Zielrechner des Pakets, schickt die dortige IP-Implementierung das Paket weiter, dient also der Vermittlung. Das Paket durchläuft auf seinem Weg meist mehrere Vermittlungen bis es den Zielrechner erreicht. Jeder Rechner im Netz muss deshalb IP verstehen. Wegweiser in IP-Netzen, die an Kreuzungen von Netzen die Daten richtig weiterleiten, werden Router genannt.

Derzeit existieren folgende Versionen von IP:

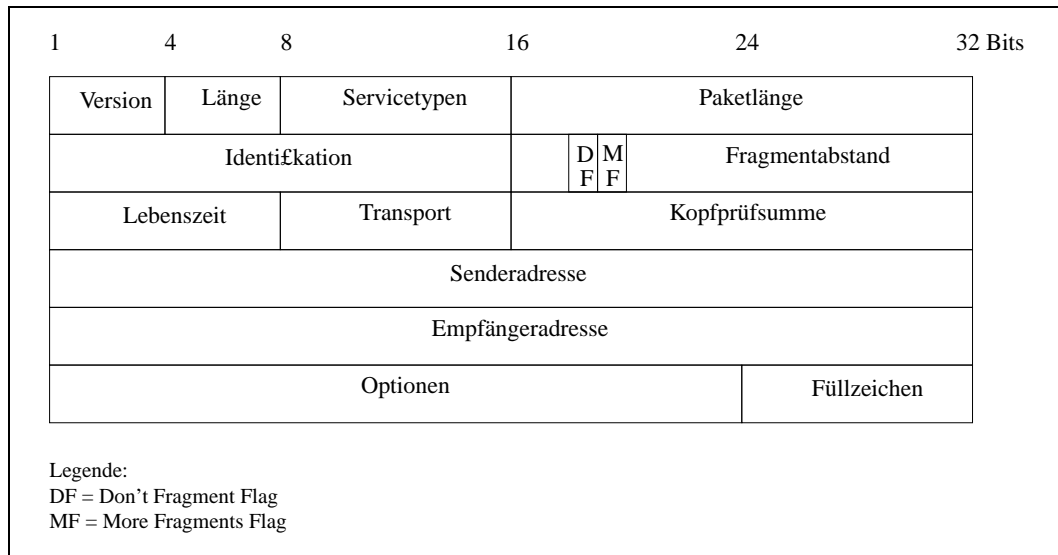
- IPv4 (Internet Protokoll Version 4) beschrieben in RFC 1716
- IPv6 (Internet Protokoll Version 6) beschrieben in RFC 1883
- IPng (Internet Protokoll Next Generation) beschrieben in RFC 1550 (heutzutage in der Regel als Synonym für IPv6 zu sehen)

Dabei sind IPv6 bzw. IPng nicht kompatible Weiterentwicklungen von IPv4. Deshalb wird im Folgenden zuerst das derzeit gebräuchliche IPv4 dargestellt.

Abbildung 2.11 zeigt den von IPv4 verwendeten Protokollkopf. Die einzelnen Felder haben folgende Bedeutung:

- Versionsnummer: Diese vier Bits breite Feld enthält die Versionsnummer des IP-Protokolls (IPv4 = 4).
- Länge: Länge des IP-Kopfes in 32-Bit Worten.
- Servicetypen: Angabe für den IP-Protokollautomaten, die Nachricht nach einem bestimmten Kriterium zu behandeln. Solche Kriterien sind Priorität, Wartezeit, Durchsatz und Zuverlässigkeit.
- Paketlänge: Enthält die Länge des Paketes in Bytes inklusive des Protokollkopfes.
- Identifikation: Das Feld dient zur eindeutigen Identifikation. Es wird z. B. bei der Defragmentierung der Pakete genutzt, um alle Teile einer Fragmentkette identifizieren zu können.
- Fragmentierungsflags: Zwei Bits namens DF (Don't Fragment) und MF (More Fragments) steuern die Behandlung des Pakets im Falle der Fragmentierung. Ein gesetztes DF Bit verbietet die Fragmentierung. Das gesetzte MF Bit signalisiert, dass noch weitere Teilpakete folgen.

Abbildung 2.11: IPv4-Protokollkopf



Quelle: [4], Seite 12

- Fragmentabstand: Enthält die Lage der in diesem Paket gespeicherten Nachricht in Bezug auf den Beginn der Gesamtnachricht in Einheiten von 8 Bytes.
- Lebenszeit: Der absendende Teilnehmer gibt hier an, wie lange das Paket im Netz verweilen darf, bevor es verworfen wird.
- Transport: Enthält die Identifikationsnummer des Transportprotokolls, dem dieses Paket zugestellt werden muss (TCP = 6)
- Sender- und Empfängeradresse: Die 32 Bit langen Internetadressen des Senders und des Empfängers.
- Optionen und Füllzeichen: Für Spezialaufgaben (Netzwerkmanagement, Sicherheit) kann der IP-Kopf um Optionen und Füllbytes erweitert werden.

Eine Internet-Adresse besteht immer aus der Netz-Identifikation und der Host-Identifikation. Bei IPv4 existieren vier verschiedene Typen von Internet-Adressen, die jeweils eine verschiedene Anzahl von Bits für die Netz-ID und die Host-ID verwenden (siehe Abbildung 2.12). Alle Rechner eines Netzwerks haben die gleiche Netzwerk-ID. Die Host-ID identifiziert eindeutig einen bestimmten Rechner innerhalb des Netzwerks. Vorsicht ist bei Host-IDs geboten, die alle Bits auf 1 oder 0 gesetzt haben, da diese Bit-Kombinationen für spezielle Funktionen reserviert sind.

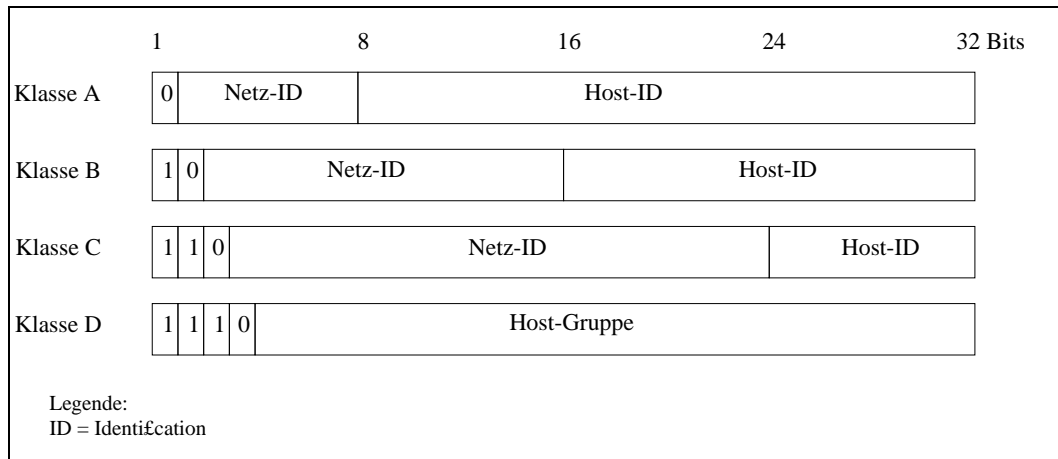
Als Beispiel dient folgende Internet-Adresse (dargestellt in der üblichen „dotted decimal notation“):

192.44.3.200₍₁₀₎ = 1100000000.00101100.00000011.11001000₍₂₎

Da die ersten beiden Bits gesetzt sind, handelt es sich hierbei um eine Klasse C Internet-Adresse (siehe Abbildung 2.12). Daher lautet die Netz-ID dieses Beispiels 192.44.3 und die Host-ID 200.

Soll mit Hilfe von IP ein Paket an einen Rechner gesendet werden, der die gleiche Netz-ID besitzt, ist das Routing sehr einfach. Das Pakete wird von einem Rechner des Netzwerks zum nächsten weitergeleitet bis der

Abbildung 2.12: Internet-Adresstypen



Quelle: [32], Seite 38

Zielrechner erreicht wird. Ist der Empfänger ein Rechner mit einer anderen Netz-ID, schickt IP das Paket an einen Router (oft auch Gateway genannt), der für die Weiterleitung zuständig ist. Ein Router hat mindestens zwei Netzwerkschnittstellen über die er an verschiedene IP-Teilnetze angeschlossen ist. Jeder Router hat eine sog. Routing-Tabelle, in der Netz-IDs Netzwerkschnittstellen zugeordnet werden. Ist die Netz-ID nicht vorhanden, wird das Paket zum Vorgabe-Router geleitet, der ausführlichere Tabellen hat. Drei Netzwerke mit unterschiedlichen Netz-IDs, die mit Hilfe von zwei Routern verbunden sind, zeigt Abbildung 2.13.

Die Fragmentierung des IP-Protokolls dient dazu, Pakete über Netzwerke mit unterschiedlichen Paketgrößen zu verschicken. Die Pakete können in einem Netzwerk über verschiedene Wege versendet werden. Aufgrund der unterschiedlichen Paketlaufzeiten müssen die Pakete nicht in der korrekten Reihenfolge empfangen werden. Die richtige Reihenfolge kann nur bei einer Punkt-zu-Punkt-Verbindung gewährleistet werden.

Jedes Fragment einer Nachricht erhält einen eigenen IP-Protokoll-Kopf (siehe Abbildung 2.14). Das Identifikationsfeld enthält in jedem neuen IP-Kopf eines Fragments die gleiche Kennung. Damit wird eine spätere Zuordnung gewährleistet. Die Lage der einzelnen Fragmente innerhalb der Gesamtnachricht wird durch das Fragmentabstandsfeld gekennzeichnet. Das MF-Bit zeigt an, dass noch weitere Pakete folgen. Durch das Setzen des DF-Bits kann ein Absender verhindern, dass ein Paket zerteilt wird.

Folgendes Beispiel (aus [32], Seite 43) dient der Anschaulichkeit:

Ausgangssituation:

Maximale Paketlänge des Netzwerks: 128 Bytes Anzahl der zu versendenden Datenbytes: 300 Bytes Identifikation des Paketes: 2354 DM-Bit: nicht gesetzt Keine Optionen

Resultat nach Fragmentierung:

Abbildung 2.13: Beispiel eines Netzwerks - Verbund mehrerer TCP/IP-Netze mittels Router

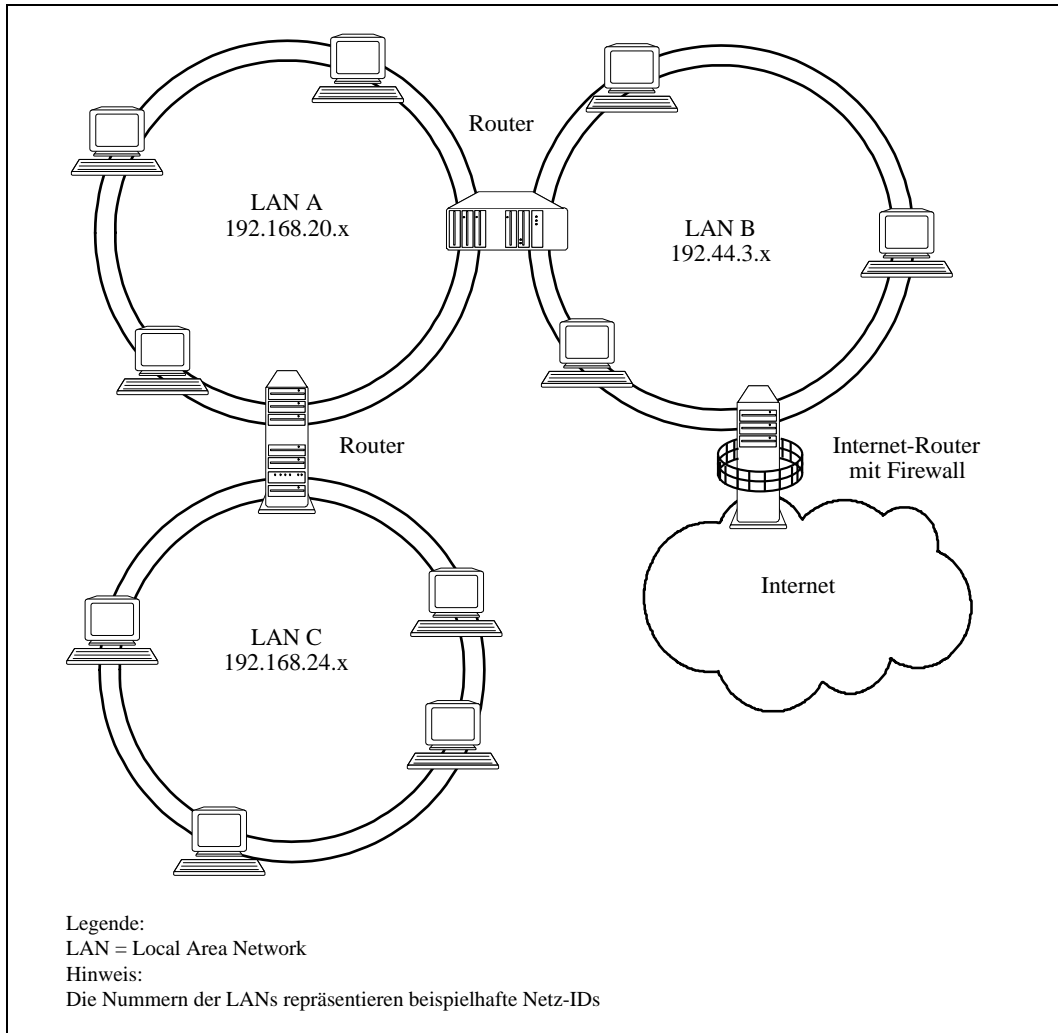
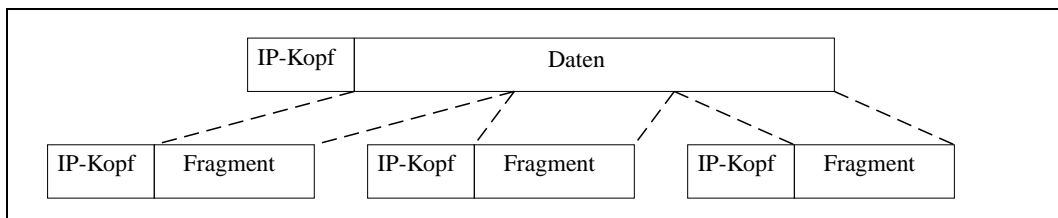


Abbildung 2.14: Fragmentierung eines IP-Paketes

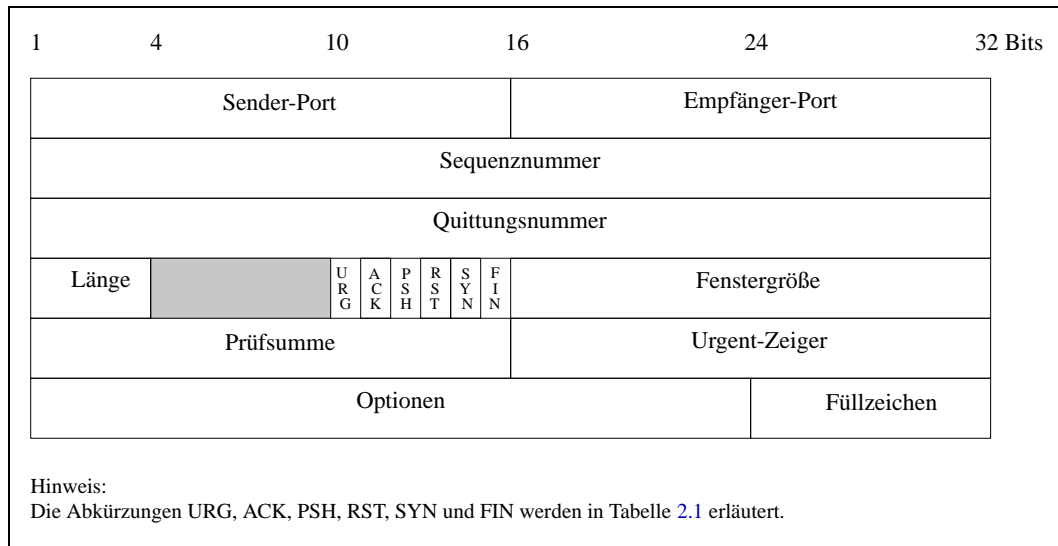


Quelle: [32], Seite 43

Fragment1: Paketlänge=124, Abstand=00, MF-Bit=1, ID = 2354
Fragment2: Paketlänge=124, Abstand=13, MF-Bit=1, ID = 2354
Fragment3: Paketlänge=112, Abstand=26, MF-Bit=0, ID = 2354

Dabei wird der Fragmentabstand in Einheiten von 8 Byte angegeben. Daher müssen sämtliche Pakete mit

Abbildung 2.15: TCP-Protokollkopf



Quelle: [32]

Ausnahme des jeweils letzten eine ganzzahlig durch 8 teilbare Länge haben. Für das Beispiel ergibt sich also:

$$\text{Abstand} = \text{trunc}\left(\frac{\text{max. Paketlänge} - \text{Kopf}}{8 \text{ Bytes}}\right) = \text{trunc}\left(\frac{128 \text{ Bytes} - 20 \text{ Bytes}}{8 \text{ Bytes}}\right) = 13$$

$$\text{Paketlänge} = \text{Abstand} * 8 \text{ Bytes} + \text{Kopf} = 13 * 8 \text{ Bytes} + 20 \text{ Bytes} = 124$$

Das weiterentwickelte Internetprotokoll Version 6 besitzt folgende Unterschiede und Erweiterungen zu Version 4:

1. Erweiterte Adressierung. Die Adressbreite wird von 32 auf 128 Bit vergrößert. Damit stehen weit mehr Adressen und eine umfangreichere Adresshierarchie zur Verfügung.
2. Vereinfachtes Kopfformat.
3. Verbesserte Unterstützung für Erweiterungen und Optionen.
4. Datenflussorientiertes Markieren. Diese neue Funktionalität von IPv6 dient zum Markieren von Paketen, für die der Absender ein bestimmtes Verarbeitungsverfahren vorgesehen hat. Ein Beispiel hierfür ist ein Echtzeitdienst, in dem zeitkritische Daten besonders behandelt werden.
5. Authentifizierung und Verschlüsselung. Dient dem Bedarf nach Sicherheit und Datenintegrität im Internet.

2.2.3.2 Transmission Control Protocol (TCP)

TCP liegt in den Schichten oberhalb von IP, der Transportschicht. TCP setzt ausschließlich auf IP auf. Das TCP arbeitet im Gegensatz zum IP verbindungsorientiert, was heißt, dass vor dem Übertragen von Datenpaketen eine bidirektionale virtuelle Verbindung auf und später wieder abgebaut wird. Während in den Schichten unterhalb vom TCP kaum Überlegungen hinsichtlich möglicher Verluste ganzer Datenpakete angestellt werden,

Tabelle 2.1: Flags im TCP-Protokollkopf

Kürzel	engl. Bedeutung	Funktion
URG	Urgent	Gesetzt, wenn der Urgent-Zeiger genutzt wird
ACK	Acknowledge	Quittungsnummer ist gültig
PSH	Push	Der Empfänger wird dazu aufgefordert, die Daten der Anwendung bei Ankunft bereitzustellen und sie nicht erst zwischenspeichern
RST	Reset	Rücksetzen der Verbindung oder Antwort auf ein ungültiges Segment
SYN	Synchronize	Verbindungsaufbau wird gewünscht: SYN muss bestätigt werden. SYN = 1, ACK = 0 für Verbindungsanfrage SYN = 1, ACK = 1 für Empfangsbestätigung
FIN	Finish	Wird genutzt um eine Verbindung einseitig abzubauen. FIN muss bestätigt werden. FIN = 1, ACK = 0 Verbindungsabbau-Anfrage FIN = 1, ACK = 1 Bestätigung

wird dies hier berücksichtigt. Das TCP überträgt aus Sicht des Dienstnutzers einen kontinuierlichen, aus Bytes bestehenden Datenstrom in beliebige Richtungen. Dabei legt es die Blockgrößen und das Weiterleiten von Daten selbst fest. Die Daten werden gesichert durch:

- Sequenznummern
- Prüfsummenbildung mit Empfangsquittungen
- Quittung mit Zeitüberwachung
- Segmentwiederholung nach Quittungszeitablauf

Ein TCP-Segment besteht aus dem festen Protokollkopf von 20 Bytes, Optionen und Daten. Abbildung 2.15 zeigt den TCP-Protokollkopf. Die einzelnen Felder enthalten folgende Informationen:

- Sender- und Empfänger-Port: Zwei 16 Bits breite Portnummern, welche die Endpunkte der virtuellen Verbindung bezeichnen. Hostrechner können mehrere TCP-Verbindungen über verschiedene Kanäle (engl. ports) aufbauen. Jeder Dienst einer höheren Schicht muss eine Portnummer für die Kommunikation festlegen. Über diese Nummer adressiert der Kunde (engl. client) den Server, der den Dienst bereitstellt. Teilweise sind für bestimmte Anwendungen, wie z. B. TELNET oder FTP¹⁹, vordefinierte Portnummern festgelegt.
- Sequenz- und Quittungsnummer: Jeweils ein 32 Bit Wort, das die Anzahl der Daten-Bytes angibt, die über die Verbindung gesendet wurden. Die Sequenznummer gilt für die Senderichtung, die Quittungsnummer für die Empfangsrichtung.
- Datenabstand: Länge des TCP-Protokollkopfes in 32-Bit-Worten. Dient zur Bestimmung des Datenbeginns.

¹⁹FTP = File Transfer Protocol

- **Flags:** Mit Hilfe dieser sechs Flags werden Aktionen in der TCP-Protokollmaschine ausgelöst (siehe Tabelle 2.1).
- **Fenstergröße:** Enthält die Zahl der Bytes, die der Empfänger in seinem Datenpuffer augenblicklich aufnehmen kann. Mit dieser Angabe steuert der TCP-Empfänger den Datenfluss, eine Fenstergröße von 0 würde den TCP-Sender stoppen.
- **Prüfsumme:** Bildet eine Prüfsumme aus den Informationen im TCP-Protokollkopf, den Daten und einem Teil des IP-Protokollkopfes. Dieser Pseudo-IP-Protokollkopf besteht aus Internet-Sende- und Empfangsadressen, Transportprotokollkennung und Länge des Datenpakets.
- **Urgent-Zeiger:** Zusammen mit der Sequenznummer ergibt der Urgent-Pointer einen Zeiger auf ein Datenbyte. Somit wird der Anfang eines Datenpaketes referenziert, das besonders wichtig ist. Dieser Mechanismus dient zum Signalisieren eines außergewöhnlichen Zustands, z. B. ein Programmabbruch oder Ähnliches.
- **Optionen:** Dieses Feld bietet die Möglichkeit, zusätzliche Funktionen bereitzustellen, die im normalen Header nicht verfügbar sind. Die wichtigste Option erlaubt jedem Host, seine maximale Größe der TCP-Nutzdaten zu spezifizieren, die er annehmen kann.

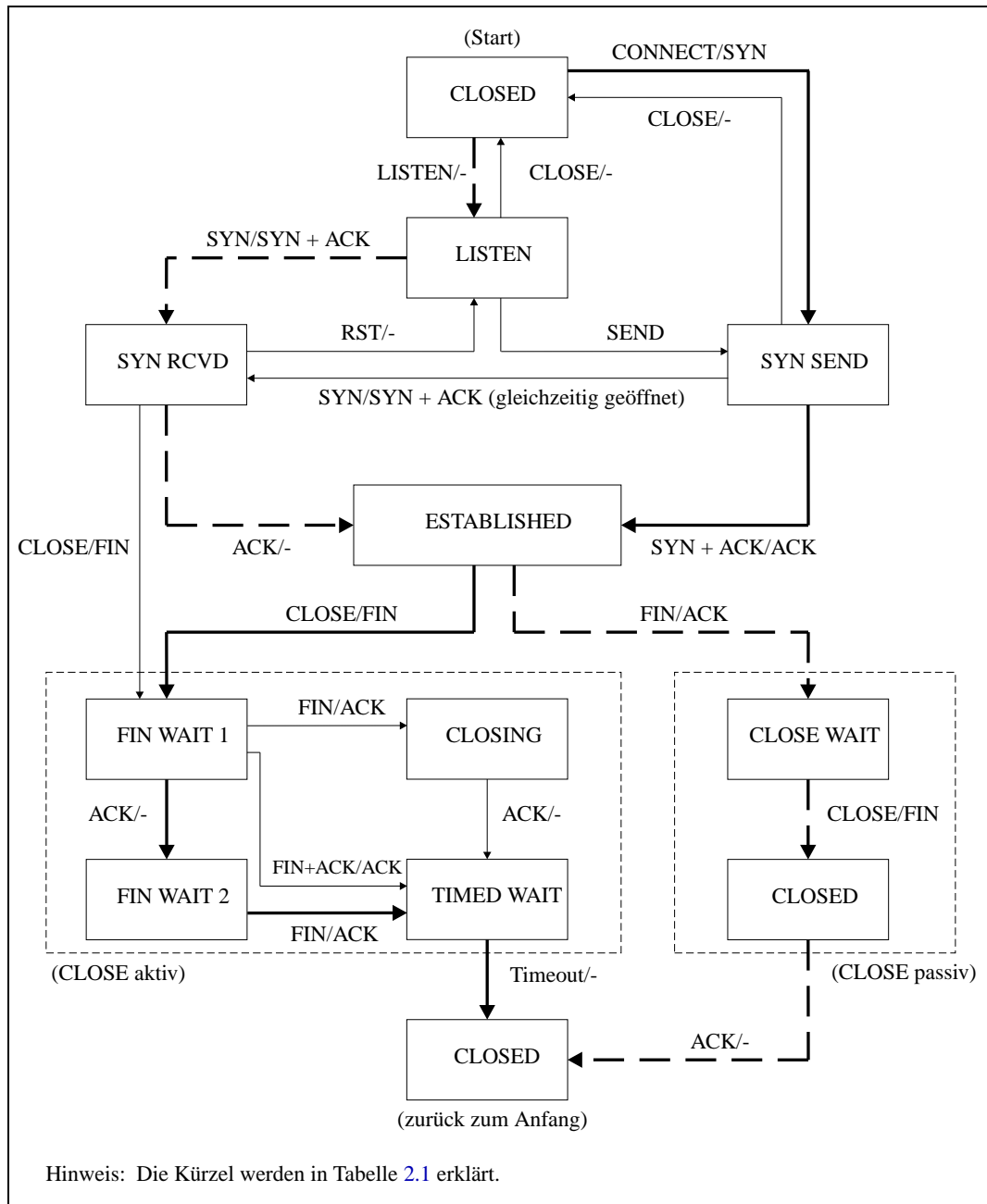
Abbildung 2.16 zeigt das TCP-Verbindungsmanagement in Form einer Zustandsmaschine. Dabei kennzeichnet die fette Linie den normalen Pfad des Clients und die fett gestrichelte Linie den Pfad des Servers. Feine Linien kennzeichnen ungewöhnliche Ereignisse. Anhand dieses Diagramms soll der Auf- und Abbau einer TCP-Verbindung verdeutlicht werden. Jeder Übergang ist mit einem Ereignis/ Aktion-Paar gekennzeichnet. Die Ereignisse CONNECT, LISTEN, SEND und CLOSE sind Systemaufrufe aus höheren Schichten. Die Aktion besteht im Senden eines Segments mit gesetztem Steuerelement (Flag), wie SYN, ACK, FIN, RST. Keine Aktion ist durch - gekennzeichnet.

Zuerst wird der Pfad des Clients verfolgt. Im Startzustand CLOSED erfolgt ein CONNECT-Aufruf der Clientmaschine. TCP sendet ein SYN-Segment und wechselt in den Zustand SYN SENT. Darauf wird auf den Empfang eines SYN+ACK des Servers gewartet und dies wiederum mit einem ACK bestätigt. Es wird also der Empfang einer Bestätigung nochmals bestätigt. Dieser Mechanismus wird 3-Wege-Handshake genannt. Er erkennt sowohl den Verlust einer Nachricht, als auch den Verlust einer Bestätigung. Durch die Verwendung von Sequenz- und Quittungsnummern kann gewährleistet werden, dass es sich nicht um irgendein ACK handelt, sondern um die Antwort des Servers auf die vorangegangene Anfrage. Nach erfolgreichem 3-Wege-Handshake befindet sich der Automat im Zustand ESTABLISHED. Nun können Daten übertragen werden.

Nach der Datenübertragung beendet die Anwendung die Verbindung durch die CLOSE-Operation. Der Client sendet ein FIN-Segment und wechselt in den Zustand FIN WAIT 1. Kommt ein ACK vom Server, wird die Verbindung in einer Richtung geschlossen und der Zustand wechselt zu FIN WAIT 2. Schließt auch die andere Seite die Verbindung, wird ein FIN vom Client empfangen, das bestätigt wird. Wenn beide Seiten geschlossen sind, wartet TCP die maximale Lebensdauer eines Pakets ab, um sicherzustellen, dass alle Pakete empfangen wurden.

Der Pfad des Servers beginnt mit einer LISTEN-Operation der Anwendung. Der Server wartet auf ein SYN vom Client und nach einem erfolgreichen 3-Wege-Handshake befindet er sich im Zustand ESTABLISHED. Die Verbindung ist aufgebaut.

Abbildung 2.16: TCP-Verbindungsmanagement als Zustandsautomat

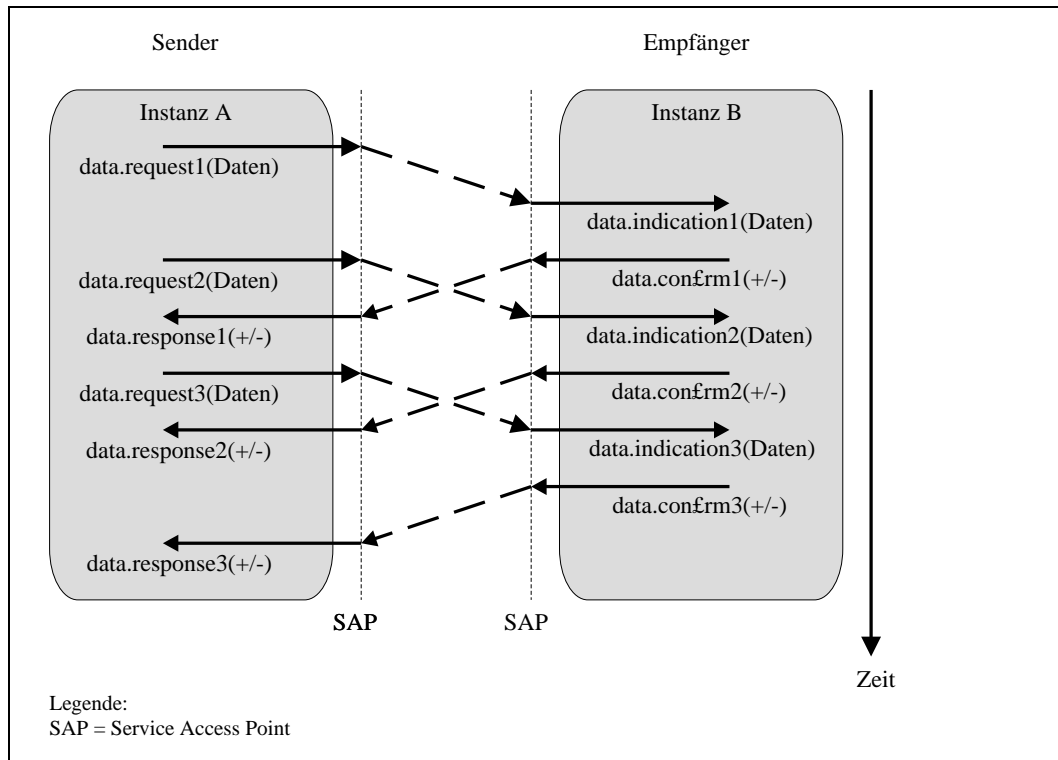


Quelle: [39], Seite 553

Hat der Client kein Interesse mehr an einer Datenübertragung, empfängt der Server FIN und sendet sein ACK. Wird der Server seitens der Anwendung ebenfalls geschlossen, sendet er ein FIN zum Client. Kommt die Bestätigung des Clients an, schließt der Server die Verbindung und löscht den Datensatz.

Das TCP-Protokoll verwendet das Prinzip des „Sliding Window“. Einfache Protokolle senden z. B. einen Datenblock und warten auf Bestätigung, bevor sie ein neues Datenpaket verschicken. Das Sliding-Window-

Abbildung 2.17: Schematische Darstellung des Sliding-Window-Prinzips als Zeitfolgediagramm



Quelle: [39]

Prinzip erlaubt jedoch, mehrere Segmente ohne Bestätigung zu versenden. Die Anzahl der Bytes, die versendet werden dürfen, wird durch die Fenstergröße festgelegt. Während des Sendevorgangs können gleichzeitig Quitungen empfangen werden, die neue Fenstergrößen festlegen. Diese Aufgabe übernimmt das Feld Fenstergröße im TCP-Protokollkopf. Dieses Prinzip ist vor allem bei Übertragungsstrecken mit großer Laufzeit sinnvoll, da sonst die Übertragungsbandbreite durch das Warten des Senders auf die Bestätigung nicht genutzt werden kann. Abbildung 2.17 zeigt das Sliding-Window-Prinzip in einem Zeitfolgediagramm.

Die Zeitüberwachung spielt im TCP-Protokoll eine große Rolle. Es werden folgende Timer gestartet, um die Abläufe zu überwachen:

- Segmentwiederholungs-Timer, dynamische Zeitdauer, veranlasst die Wiederholung eines Segments nach Ablauf des Timers.
- Persistenz-Timer, ca. 5 Sekunden, überprüft die Gegenseite durch Senden eines Minipaketes, wenn das Empfangsfenster gleich null ist.
- Quiet-Time-Timer, ca. 30 Sekunden, gibt Ports nach Abbau einer Verbindung erst nach Ablauf des Timers wieder frei.
- Keep-Alive-Timer, ca. 45 Sekunden, überprüft den Zustand der Gegenseite durch Senden eines Paketes.
- Idle Timer, ca. 360 Sekunden, bricht die Verbindung nach Ablauf ab, wenn der Keep-Alive-Timer keine Antwort erhalten hat.

Durch die dynamische Manipulation der Fenstergröße und Zeitüberwachung durch Timer, wird im TCP-Protokoll eine Überlastungsüberwachung realisiert. Bei Aufbau einer Verbindung muss eine geeignete Fenstergröße gewählt werden. Der Empfänger kann ein Fenster z. B. auf Grundlage seiner Puffergröße spezifizieren. Hält sich der Sender an diese Größe, können Übertragungsprobleme nicht aufgrund eines Pufferüberlaufs am Empfangsende auftreten. Ein Timeout im TCP ist deshalb entweder auf einen Paketverlust oder auf eine Überlastung des Netzes zurückzuführen. Häufen sich die Timeouts, müssen die Fenstergröße und die Timeouts, die eine Wiederholung des Segments bewirken, entsprechend angepasst werden. Dadurch wird einer Überlastung entgegengewirkt.

Das TCP ist ein kompliziertes Protokoll mit mehreren Optionen und vielen Erweiterungen. Für den Einsatz von TCP als Kommunikationsprotokoll für diese Internetkamera sind diese Zusatzfunktionen uninteressant. Sie werden deshalb an dieser Stelle nicht beschrieben.

2.2.4 Sockets

Die Socket-Schnittstelle dient als standardisierte Programmierschnittstelle zwischen dem Anwendungsprogramm und TCP/IP. Neben der rechnerübergreifenden Kommunikation von Prozessen über TCP/IP, wird sie auch zur lokalen Interprozesskommunikation genutzt.

Die Socket-Schnittstelle kommt aus der UNIX-Welt. Mit dem 4.2BSD²⁰ wurden die Sockets in UNIX²¹ eingeführt. Eine allgemein verfügbare Spezifikation, die auf den UNIX-Sockets basiert, wurde von der Firma Microsoft unter der Bezeichnung WinSock in das WOSA²² aufgenommen und wurde damit ein etablierter Standard.

Ein großer Vorteil der Verwendung von Sockets ist ihre Homogenität, d. h. eine Kommunikation über Sockets ist unabhängig von dem Standort der Prozesse, dem Betriebssystem, der Rechnerarchitektur und dem Übertragungsmedium.

Der Aufbau einer TCP/IP Verbindung aus einem Anwenderprogramm wird durch die Verwendung von Sockets sehr einfach. Der Anwendungssoftware stehen die in Tabelle 2.2 gezeigten TCP-Systemaufrufe zur Verfügung.

Zum Aufbau einer Verbindung werden vom Client und Server mit Hilfe des Funktionsaufrufes `socket()` eine Instanz der Socket-Schnittstelle erstellt und das Transportprotokoll festgelegt. Durch den Systemaufruf `bind()` legt der Server den lokalen Host und den lokalen Port fest. Damit ist die Adresse festgelegt, auf welcher der Server dem Client antwortet. Nach dem Aufruf der Funktionen `listen()` und `accept()` wartet der Server auf ein `connect()` des Clients. Der Client bekommt mit dem `connect()`-Aufruf implizit eine lokale Adresse zugewiesen und übergibt sie dem Server. Die Verbindung ist nun aufgebaut und Client und Server können mit `read()`- und `write()`-Funktionen bidirektionalen, byteorientierten Datenaustausch betreiben. Durch den Systemaufruf `close()` wird die Verbindung abgebaut.

Tabelle 2.2 vergleicht eine Socket-Verbindung zwecks Anschaulichkeit mit einem Telefongespräch.

Ausführliche Grundlagen und Programmbeispiele zur Socketprogrammierung sind in [3] zu finden. Hier wurde die Socket-Schnittstelle nur so weit beschrieben, wie es zum Verständnis der Zusammenhänge in dieser

²⁰BSD = Berkeley System Distribution

²¹UNIX = UNICS = Uniplexed Information and Computing System

²²WOSA = Windows Open Service Architecture

Tabelle 2.2: Analogie zwischen dem Nutzen einer Socket-Schnittstelle und Telefonieren

Prozess 1 Server	Prozess 2 Client	Funktion	Analogie beim Telefonieren
socket()	socket()	Socket erstellen	Telefonhörer abheben
bind()		Adresse dem Socket zuweisen	Telefonnummern zuweisen
listen ()		Erlaubnis erteilen, mit dem Socket Verbindung aufzunehmen	Erlaubnis erteilen, an- gerufen zu werden
	connect()	Verbindungswunsch mitteilen	Telefonnummer wählen
accept()		Verbindung komplett aufbauen	Anruf beantworten
write()	write()	Daten über die Schnittstelle senden	Reden
read()	read()	Daten empfangen	Zuhören
close()	close()	Socket schließen	Außergehen

Quelle: [27], Seite 223

Diplomarbeit erforderlich ist.

2.2.5 HTTP

Das HTTP²³ liegt in der Anwendungsschicht des OSI-Referenzmodells. HTTP ist das gemeinsame Protokoll von Clients und Servern im Internet, über das Dokumente ausgetauscht werden. Dabei kann HTTP beliebige Datenformate transportieren.

Für die Übertragung eines Dokumentes über das HTTP-Protokoll wird mit Hilfe von Sockets eine TCP-Verbindung zwischen Client und Server aufgebaut. Dabei wird standardmäßig der Port 80 verwendet.

Der Client sendet eine Anfrage (engl. request), die folgendermaßen aufgebaut ist:

<Methode>	<Request-URL>	<HTTP-Version>
<Kopfzeile (Header)>		
<Objektdaten (Entity)>		

Der Server sendet daraufhin eine Antwort (engl. response). Sie enthält das angeforderte Dokument oder eine Fehlermeldung. Die Antwort hat folgende Struktur:

<HTTP-Version>	<Statuskode>	<Text>
<Kopfzeile (Header)>		
<Objektdaten (Entity)>		

Nach erfolgter Anfrage und Antwort wird die Verbindung abgebaut. Das Senden eines HTTP-Dokuments soll für den Server eine minimale Belastung in Form von Speicherbelegung und Rechenzeit darstellen. Daher wird die Verbindung in den meisten Fällen nach jedem Dokument wieder abgebaut, damit der Server in seinen Grundzustand zurückkehren kann.

Im Folgenden sollen die Elemente der Client-Anfrage und der Server-Antwort vorgestellt werden.

²³HTTP = HyperText Transfer Protocol

Tabelle 2.3: Anfragemethoden des HTTP

<Methode>	Beschreibung
GET	Anfrage zum Lesen eines Dokuments
HEAD	Anfrage zum Lesen der Informationen über das Dokument
PUT	Anfrage zum Speichern eines Dokuments
POST	Anfrage zum Speichern und Zuordnen eines Dokuments
DELETE	Entfernen eines Dokuments
LINK	Verbinden zweier Ressourcen
UNLINK	Löst die Verbindung zwischen zwei Ressourcen

In Tabelle 2.3 sind die standardisierten <Methoden> in HTTP Version 1.0 einer Client-Anfrage aufgelistet. Die häufigste Methode ist GET, mit der ein Client ein Dokument vom Server abrufen.

Mit Hilfe der <Request-URL> wird dem Server mitgeteilt, welches Dokument mit der Methode behandelt werden soll. Im Gegensatz zu einem Request-URL besteht eine typische URL²⁴ normalerweise aus 3 Teilen:

- dem Protokoll (z. B. http://, ftp://, mailto://, gopher://, telnet://)
- dem DNS²⁵-Namen des Host. Mit Hilfe des DNS wird der Name des Hosts in seine numerische IP-Adresse umgesetzt.
- dem Dokumentnamen

Als Beispiel-URL dient die Web-Page des Fraunhofer Instituts:

<http://www.fhg.de/german/index.html>

Das Übertragungsprotokoll dieser Beispiel-URL ist HTTP.

Der DNS-Name www.fhg.de entspricht der IP-Adresse 153.96.68.2.

Der Dokumentname lautet `/german/index.html`.

Da dem HTTP-Server das Protokoll und seine eigene IP-Adresse bekannt sind, besteht eine Request-URL nur aus dem Dokumentnamen. Die Request-URL des Beispiels lautet damit: `/german/index.html`.

Zur Zeit wird normalerweise die Version 1.0 des HTTP-Protokolls verwendet. Deshalb trägt das Element <HTTP-Version> die Kennung HTTP/1.0.

Mit Hilfe von <Kopfzeilen (Header)> werden nähere Angaben zur Anfrage (Antwort) des Clients (Servers) gemacht. Es werden vier verschiedene Headertypen unterschieden:

- Allgemeiner Header, enthält Angaben über die Nachricht, die gerade übertragen wird. Verwendete Kodeworte sind Date, Forwarded, Message-ID, MIME-Version.
- Request Header, macht Informationen zur Anfrage, der gewünschten Antwort, und zum Benutzer selbst. (Accept, Accept-Charset, Accept-Encoding, Accept-Language, Authorization, From, If-Modified-Since, Pragma, Referer, User, Agent).

²⁴URL = Uniform Resource Locator

²⁵DNS = Domain Name Service

Tabelle 2.4: Statusklassen in der HTTP-Antwort

Statuskode	Beschreibung
1xx	Information
2xx	Aktion ausgeführt
3xx	Weitere Maßnahme erforderlich
4xx	Client-Fehler
5xx	Server-Fehler

Quelle: frei nach [11], Seite 189

- Response Header, erhält Angaben zur Antwort. (Public, Retry-After, Server, WWW-Authenticate).
- Entity Header, macht Angaben zu den übertragenen Objektdaten. (Content-Type, Content-Length, Content-Language, Content-Encoding, Content-Transfer-Encoding, Last-Modified, Expires, Location, URI, Link, Title, Version, Derived-From).

Der <Statuskode> ist eine dreistellige Zahl, die den Client über Erfolg oder Misserfolg seiner Anfrage informiert. Der Server sendet zusätzlich zum Kode einen erläuternden <Text>, den der Client dem Benutzer anzeigen kann. Eine Liste mit Statuskodes ist z. B. in ([11], Seite 189) zu finden. Allgemein werden verschiedene Statusklassen unterschieden (siehe Tabelle 2.4). Die erste Ziffer des Statuskodes bestimmt die Klasse.

2.3 ISDN

ISDN²⁶ ist ein digitales Telekommunikationssystem. Mit ISDN ist es möglich, Dienste wie Sprach-, Text-, Bild- und Datenkommunikation über ein öffentliches Telefonnetz anzubieten.

2.3.1 ISDN Basisanschluss

Ein ISDN-Basisanschluss besitzt zwei Nutzkanälen (B-Kanäle) mit einer Bandbreite von je 64 KBit und einem Steuerkanal (D-Kanal) von 16 KBit. Ein B-Kanal ist ein bittransparenter, synchroner, leitungsvermittelter Übertragungskanal. Über ihn können PCM²⁷-kodierte Sprachdaten oder andere Nutzdaten übertragen werden. Der D-Kanal dient zur Übertragung der Daten von D-Kanal-Protokollen, die z.B. den Auf- und Abbau einer Verbindung regeln.

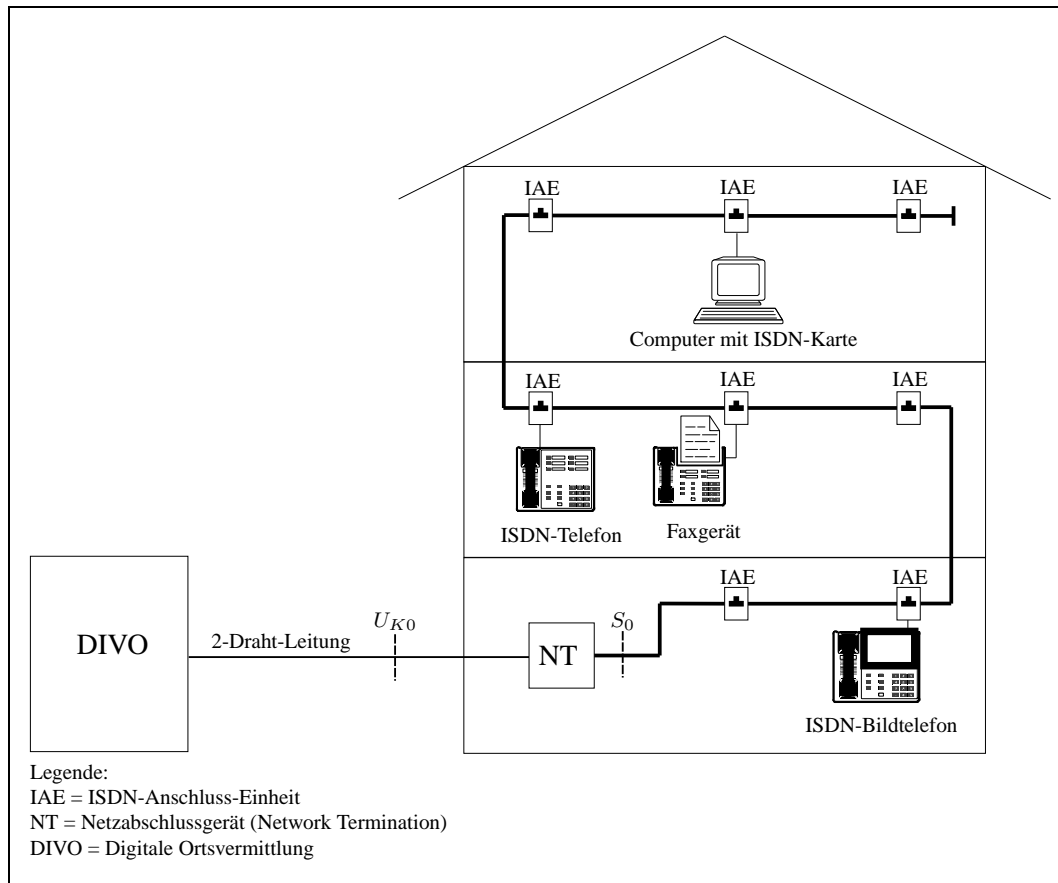
Eine typische Hausinstallation eines ISDN-Basisanschlusses in Form eines Buses ist in Abbildung 2.18 zu sehen. Die digitale Ortsvermittlung (DIVO) wird über eine 2-Draht-Leitung mit einem Netzabschlussgerät (NT²⁸) im Haus verbunden. Dabei kann die alte 2-Draht-Leitung des analogen Telefonanschlusses verwendet werden. Die ISDN-Anschluss-Einheiten (IAE) sind über den S_0 -Bus mit dem NT verbunden. An jede IAE kann ein beliebiges ISDN-Endgerät angeschlossen werden.

²⁶ISDN = Integrated Services Digital Network

²⁷PCM = Pulse Code Modulation

²⁸NT = Network Termination

Abbildung 2.18: Beispiel einer ISDN-Hausinstallation



Quelle: [1]

2.3.2 S₀-Bus

Der S_0 -Bus ist eine 4-Draht-Leitung, bestehend aus einem Sende- und einem Empfangsaderpaar. Die zwei B-Kanäle und der D-Kanal werden im Zeitmultiplex zu einem Bitstrom gebündelt, der nur ein Leitungspaar des S_0 -Buses benötigt (siehe Abbildung 2.19). Somit ist ein Vollduplexbetrieb möglich, bei dem ein Endgerät gleichzeitig über das eine Aderpaar sendet und über das andere empfängt.

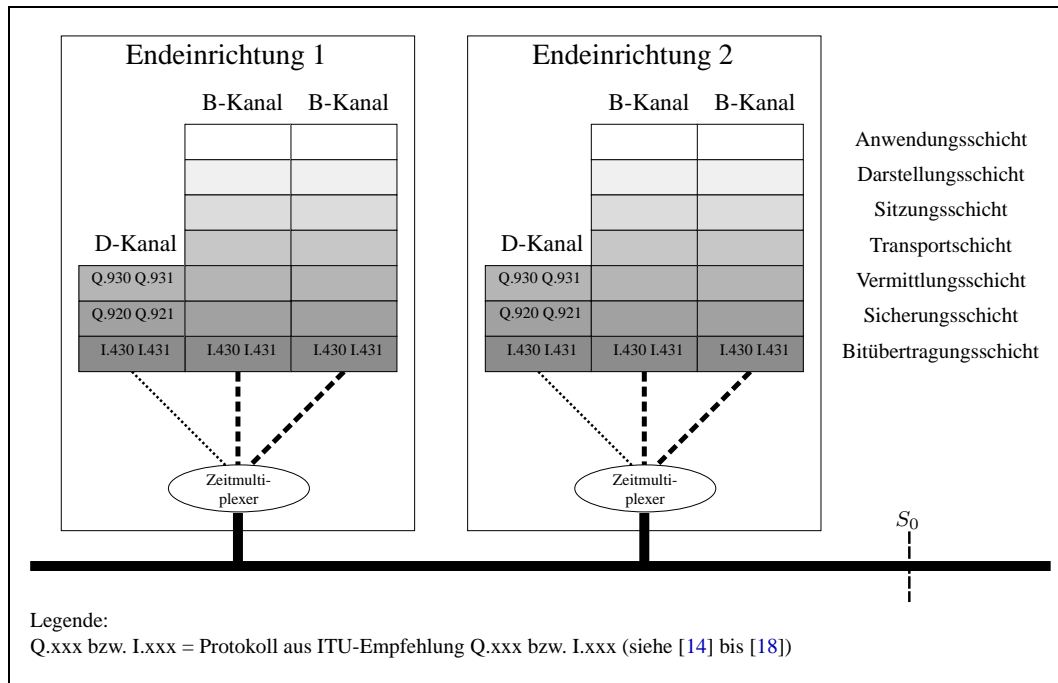
Für ein normales Telefongespräch (Sprachkommunikation) wird nur ein B-Kanal benötigt. Da der ISDN-Basisanschluss 2 B-Kanäle besitzt, können gleichzeitig 2 Gespräche über ISDN geführt werden. Der D-Kanal muss in diesem Fall von zwei Endgeräten geteilt werden. Ein Endgerät, das Steuerinformationen versenden will, muss deshalb den D-Kanal zuerst anfordern (Request/Grant-Mechanismus).

2.3.3 ISDN-Protokollarchitektur im OSI-Schichtenmodell

Jedes Gerät, welches das ISDN-Protokoll versteht, kann an den S_0 -Bus angeschlossen werden. Das ISDN-Protokoll beruht auf Empfehlungen der ITU²⁹. Es besteht aus D-Kanal- und B-Kanal-Protokollen. Die D-Kanal-Protokolle können auf die unteren drei Schichten des OSI-Referenzmodell aus Abschnitt 2.1 abgebildet wer-

²⁹ITU = International Telecommunication Union

Abbildung 2.19: Schichtenmodell der ISDN-Protokolle



Quelle: frei nach [1], Seite 21

den. In Abbildung 2.19 sind zwei Endgeräte mit ihren Protokollschichten zu sehen. In den einzelnen Schichten sind jeweils die ITU-Empfehlungen für das Protokoll angegeben. Dabei entfällt auf die einzelnen Schichten des D-Kanalprotokolls folgende Funktion (Quelle [1], Seite 22):

- **Bitübertragungsschicht.** Diese Schicht ist für die Übertragung der Steuerinformationen im D-Kanal verantwortlich.
- **Sicherungsschicht.** Diese Schicht übernimmt die gesicherte Übermittlung der Steuerinformationen und der eventuell im D-Kanal übertragenen Daten.
- **Vermittlungsschicht.** Innerhalb dieser Schicht wird die eigentliche Benutzersignalisierung realisiert. Dazu gehören die Funktionen, die sowohl zum Auf- und Abbau von ISDN-Verbindungen als auch zur Realisierung von ISDN-Dienstmerkmalen erforderlich sind.

Die Steuerung für die beiden B-Kanäle wird nur innerhalb der ersten Schicht bereit gestellt.

2.4 JPEG-Standard

Der JPEG³⁰-Standard „Digital Compression and Coding of Continuous-Tone Still Images“ ist ein Einzelbildkompressionsstandard, der aus einer gemeinsamen Arbeit des CCITT³¹ (jetzt ITU) und der ISO³² entstanden ist.

³⁰JPEG = Joint Photographic Experts Group

³¹CCITT = Comité Consultatif International Télégraphique et Téléphonique

³²ISO = International Standardization Organization

Bei der Übertragung über eine Punkt-zu-Punkt Verbindung oder über ein Netzwerk ist eine Komprimierung der Kamerabilder sinnvoll. Durch Reduktion der Datenmenge wird die Netzbelastung in einem Netzwerk reduziert und die mögliche Bildwiederholfrequenz bei einer Übertragung über eine Punkt-zu-Punkt-Verbindung erhöht.

Der JPEG-Standard wird an diesem Kapitel beschrieben, da er später für die Kompression der Kamerabilder verwendet werden soll. Für die optimale Konfiguration einer JPEG-Implementierung ist die Einstellung mehrerer Parameter erforderlich. Dafür ist ein grundlegendes Verständnis des JPEG-Algorithmus hilfreich.

Der JPEG-Standard wurde in zwei Teilen veröffentlicht:

- Requirements and Guidelines (ITU/CCITT T.81) [5]
- Compliance Testing (ITU/CCITT T.83) [6]

Die nun folgende Kurzbeschreibung bezieht sich auf T.81. Das dort beschriebene Komprimierungsverfahren ist auf digitale Einzelbilder mit beliebiger Zeilen- und Spaltenanzahl anwendbar, die eines der folgenden Farbformate besitzen:

- Graustufen
- Farbkomponenten RGB³³
- Luminanz-/Farbdifferenzkomponenten YUY oder YIQ

Die YUV- und YIQ-Darstellung kann aus den 3 Farbkomponenten (R)ot, (G)rün und (B)lau nach folgender Formel berechnet werden:

$$\begin{bmatrix} Y \\ U \\ V \end{bmatrix} = \begin{bmatrix} 0.299 & 0.587 & 0.114 \\ -0.146 & -0.288 & 0.434 \\ 0.617 & -0.517 & -0.1 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

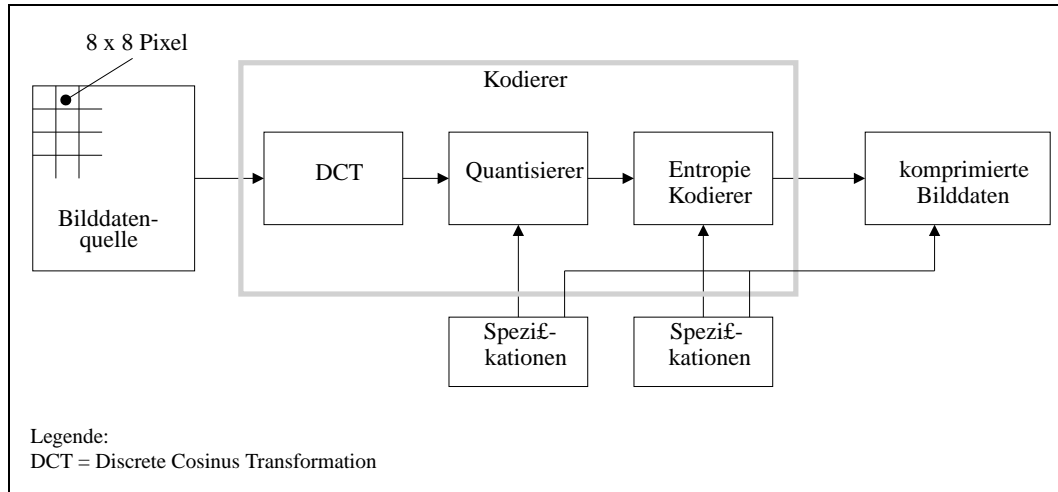
$$\begin{bmatrix} Y \\ I \\ Q \end{bmatrix} = \begin{bmatrix} 0.299 & 0.587 & 0.114 \\ 0.597 & -0.277 & -0.321 \\ 0.213 & -0.523 & -0.309 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

Es gibt vier JPEG-Kompressionsverfahren:

- Sequenzieller DCT³⁴-Modus
- Progressiver DCT-Modus
- Hierarchisch-progressiver Modus
- Verlustfreier Modus

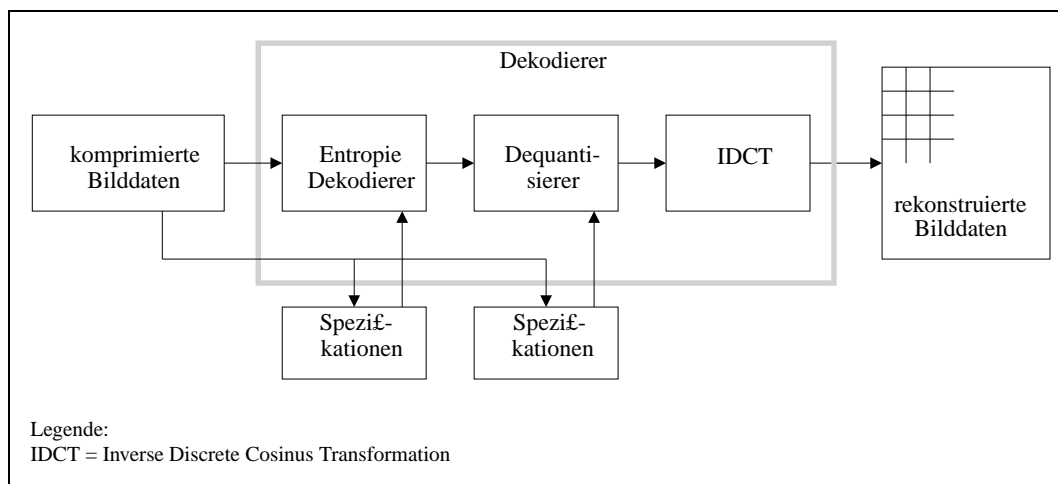
Ein JPEG-Implementierung nach H.81 muss nicht alle Modi unterstützen.

Abbildung 2.20: JPEG-Kodierer im sequenziellen DCT-Modus



Quelle: [41]

Abbildung 2.21: JPEG-Dekodierer



Quelle: [41]

2.4.1 Sequenzieller DCT-Modus

Dies ist die am weitesten verbreitete Variante. Die Vorgehensweise zur Kompression (Dekomprimierung) ist in Abbildung 2.20 (2.21) gezeigt. Die Komprimierung funktioniert wie folgt:

Das Bild wird in Blöcke von jeweils 8 x 8 Pixel aufgeteilt. Auf jedem dieser Blöcke wird eine diskrete Kosinus-Transformation (DCT) durchgeführt:

$$F_{uv} = (1/4)C(u)C(v) \sum_{i=0}^7 \sum_{j=0}^7 f(i, j) \cos((2i + 1)u\pi/16) \cos((2j + 1)v\pi/16)$$

³³RGB = Rot, Grün, Blau

³⁴DCT = Discrete Cosinus Transformation

Der Wert in der linken oberen Ecke ist der Gleichanteil F_{00} der DCT. Je weiter die Werte in Richtung rechte untere Ecke liegen, desto höher ist die zugehörige Frequenz (vgl. Abbildung 2.22).

An diesem Zahlenbeispiel ist die Reduzierung der Datenmenge durch die DCT zu erkennen. Die Matrix enthält viele Nullen und die „Energie“ ist in der linken oberen Ecke gebündelt.

Nach der DCT werden die Bilddaten quantisiert. Jedem aus der DCT resultierenden Wert der Matrix F_{uv} wird ein korrespondierender Quantisierungs-Koeffizient Q_{uv} zugeordnet. Der quantisierte Wert Sq_{uv} berechnet sich dann wie folgt:

$$Sq_{uv} = \text{abs}\left(\frac{F_{uv}}{Q_{uv}}\right)$$

Bei der Wahl der Quantisierungs-Koeffizienten wird eine Frequenzgewichtung benutzt, so dass die tieffrequenten Koeffizienten üblicherweise genauer quantisiert werden. Es wird dabei angenommen, dass das Auge gegenüber Fehlern bei höheren Frequenzen weniger anfällig ist.

Nach der Quantisierung werden die Werte gerundet. Durch das Runden entstehen Informationsverluste, die abhängig vom Grad der Quantisierung sind. Je weniger die Werte gerundet werden, desto mehr Speicherplatz muss für die Speicherung des Wertes verwendet werden. Das führt zu einer größeren Datei.

Das folgende Beispiel für mögliche Quantisierungs-Koeffizienten Q_{uv} ist aus H.83 entnommen.

08	06	05	08	12	20	26	30
06	06	07	10	13	29	30	28
07	07	08	12	20	29	35	28
07	09	11	15	26	44	40	31
09	11	19	28	34	55	52	39
12	18	28	32	41	52	57	46
25	32	39	44	52	61	60	51
36	46	48	49	56	50	52	50

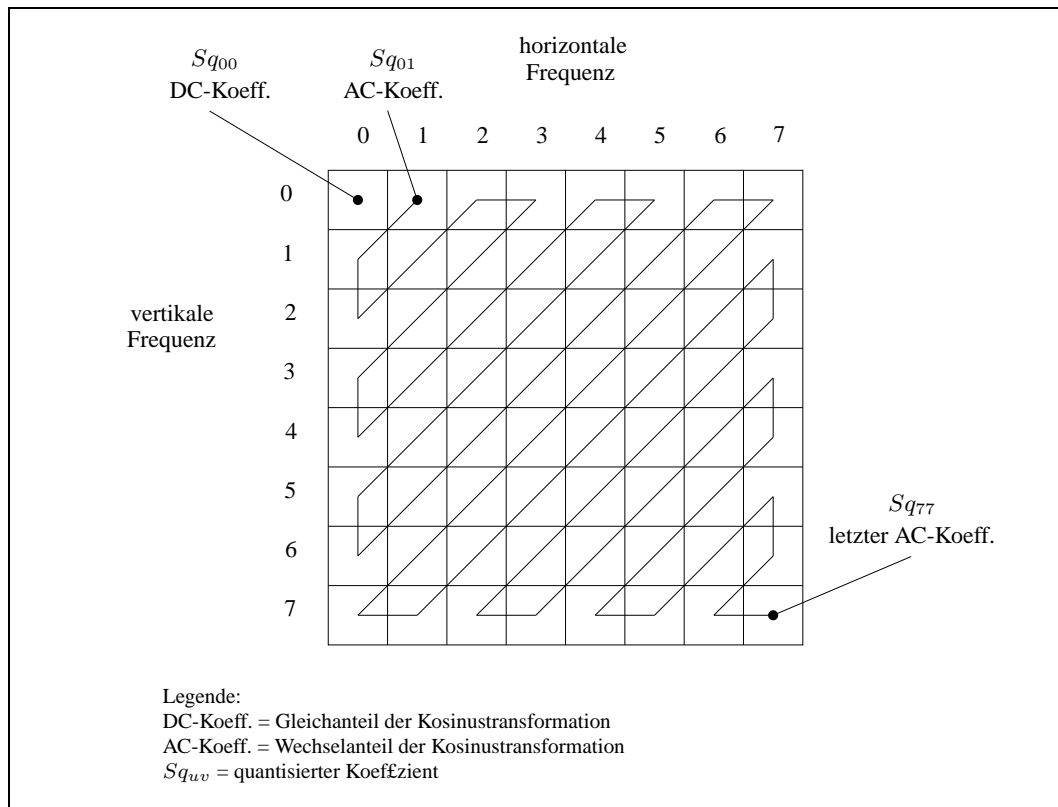
Die Koeffizienten Q_{uv} für die tieffrequenten Signale sind in der linken oberen Ecke und der höherfrequenten Signale in der rechten unteren Ecke. Je kleiner Q_{uv} desto genauer wird der zugehörige Wert F_{uv} quantisiert. Es wird also die linke obere Ecke am genauesten quantisiert.

Nach der DCT und der Quantisierung der 8 x 8 Pixelmatrix werden die 63 AC-Koeffizienten (Sq_{01} bis Sq_{77}) nach dem in Abbildung 2.22 gezeigten Schema ausgewertet. Beim DC-Koeffizienten Sq_{00} , dem Gleichanteil der DCT, wird die Differenz zum Gleichanteil des vorangegangenen 8 x 8 Blockes gebildet. Diese Differenzbildung stellt die einzige Abhängigkeit zwischen den Blöcken dar, aus denen das Gesamtbild zusammengesetzt ist.

Im nächsten Schritt werden die Daten laufrängenkodiert. Wie im obigen Beispiel gezeigt, werden viele quantisierte Koeffizienten zu null. Daher werden die nicht verschwindenden Koeffizienten durch die Anzahl der ihnen vorangehenden Nullen und dem Wert des Koeffizienten selbst kodiert.

Nun folgt eine weitere Verschlüsselung durch Entropiekodierung des Datenstroms. Dies bedeutet, dass den häufig vorkommenden Werten kurze Kodeworte zugeordnet und längere für selten auftretende Werte verwendet werden. Ein Verfahren dazu ist die Huffman-Kodierung. Mit folgendem einfachen Verfahren, das [2] entnommen ist, kann ein Huffman-Kode erzeugt werden:

Abbildung 2.22: Auswertungsmuster zum Umsortieren der zweidimensionalen DCT-Koeffizienten in ein ein-dimensionales Feld



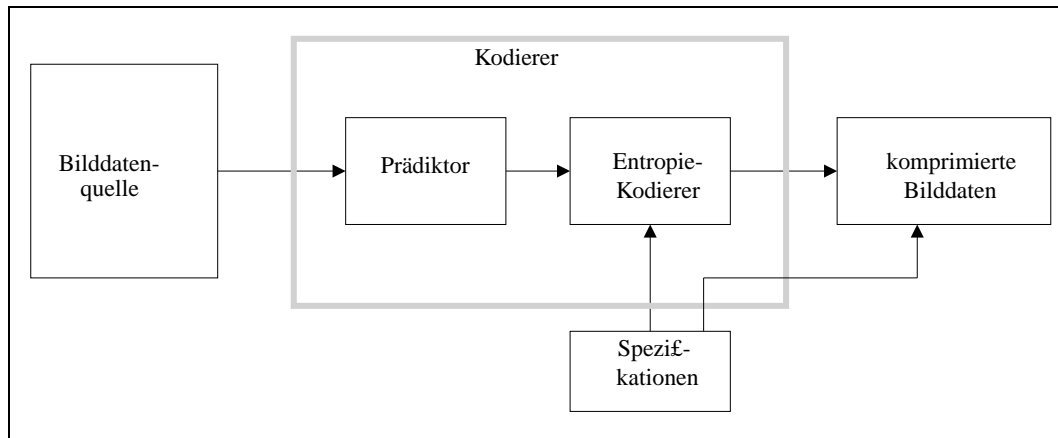
Quelle: [10]

1. Ordne die Zeichen nach der Wahrscheinlichkeit ihres Auftretens.
2. Bilde aus den beiden Zeichen mit kleinstem Gewicht (Wahrscheinlichkeit) einen Baum, dessen Kinder mit null bzw. eins markiert werden.
3. Entferne diese beiden Zeichen aus der Liste und füge stattdessen ein neues Zeichen für den soeben konstruierten Baum hinzu. Das Gewicht dieses Zeichens ist die Summe der Gewichte der beiden Kinder.
4. Bilde aus beiden Zeichen mit dem kleinsten Gewicht in der aktualisierten Liste einen Baum, dessen Kinder mit null bzw. eins markiert werden. Dieser Baum kann zwei andere Zeichen oder auch ein Zeichen und den soeben konstruierten Baum enthalten.
5. Wiederhole diese Prozedur, bis ein einziger großer Baum gebildet ist.

Der JPEG-Standard stellt einen Satz von Huffman-Kodes zur Verfügung. Es kann jedoch ein beliebiger Huffman-Kode verwendet werden. Dieser muss im Header der komprimierten Bilddatei beschrieben werden.

Zusätzlich werden im Header der JPEG-Datei Informationen über die Abmessung des Bildes, die Art der Kodierung und die Quantisierungsniveaus abgelegt.

Abbildung 2.23: JPEG-Kodierer im verlustfreien Modus



Quelle: [41]

2.4.2 Progressiver DCT-Modus

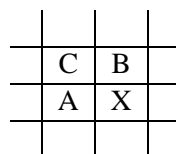
Der progressive DCT-Modus erlaubt einen Bildaufbau mit sukzessive wachsender Qualität. Beim sequenziellen Modus ist dies nicht möglich, da das Bild blockweise bearbeitet und übertragen wird. Dagegen werden beim progressiven DCT-Modus zuerst die Gleichanteile aller Blöcke, dann die niederfrequenten Koeffizienten und zuletzt die hochfrequenten Werte übertragen. Dem Empfänger ist es daher möglich, das Bild schrittweise genauer zu rekonstruieren. Dabei entsteht keine höhere Bitrate als im sequenziellen Modus.

2.4.3 Hierarchisch-progressiver Modus

In der Anfangsphase des sukzessiven Bildaufbaus sind starke Blockeffekte im dekodierten Bild bei der Verwendung des einfachen progressiven DCT-Modus zu erkennen. Durch den hierarchisch-progressiven Modus soll dies verhindert werden. In der 1. Stufe wird das Bild um den Faktor 16 unterabtastet und nach dem bekannten Verfahren kodiert und übertragen, so dass bereits die 1. Stufe eine annehmbare Bildqualität besitzt. Die folgenden Stufen tasten das Bild immer feiner ab. Bei dieser Anwendung steigt die Übertragungsrate gegenüber dem progressiven Modus um bis zu 30 Prozent.

2.4.4 Verlustfreier Modus

Da die DCT-Kodierung zur verlustlosen Übertragung nicht geeignet ist, wird anstatt der DCT ein Prädiktor benutzt. Ein Prädiktor errechnet einen Schätzwert für den Bildpunkt X aus den benachbarten Bildpunkten (A, B und C).



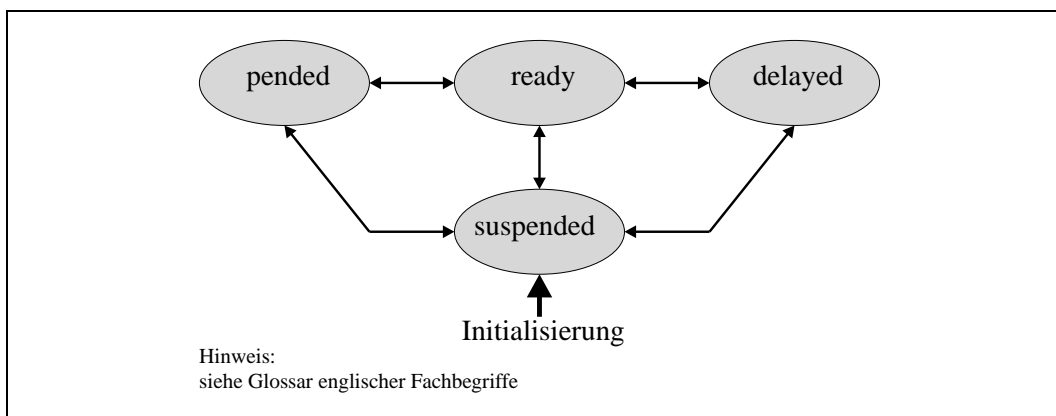
Dabei können die Prädiktoren aus Tabelle 2.5 verwendet werden. Der Schätzwert des Prädiktors wird von dem wahren Wert des Bildpunktes X subtrahiert und die Differenz wird mit Hilfe einer Entropiekodierung weiterverarbeitet. Abbildung 2.23 zeigt den verlustfreien Kodierer.

Tabelle 2.5: Prädiktoren für den verlustfreien Modus

Wert	Prädiktor
0	kein Prädiktor
1	$X = A$
2	$X = B$
3	$X = C$
4	$X = A + B - C$
5	$X = A + (B - C)/2$
6	$X = B + (A - C)/2$
7	$X = (A + B)/2$

Quelle: [5]

Abbildung 2.24: Taskzustände und Übergänge



Quelle: [26], Seite 28

Die Entropiekodierung ist identisch mit dem in Abschnitt 2.4.1 beschriebenen Verfahren. Der verlustfreie Modus des JPEG-Standards erzeugt typischerweise Komprimierungsraten von ca. 2:1.

2.5 VxWorks

VxWorks ist ein Echtzeitbetriebssystem der Firma „Wind River Systems“, das für die Programmierung der Internetaufnahmegeräte verwendet wird. In diesem Abschnitt sollen diejenigen Grundlagen der Echtzeitprogrammierung vorgestellt werden, die bei der Kamera Anwendung finden.

2.5.1 Task

Bei einem Echtzeitbetriebssystem besteht eine Anwendung in der Regel aus mehreren unabhängigen Programmteilen. Werden diese Programmteile vom Prozessor ausgeführt, werden sie Task genannt. Jeder Task benötigt dazu Systemressourcen, wie z.B. Rechenzeit oder Speicherplatz.

Ein Task kann fünf Zustände einnehmen:

- EXECUTING. Der Task wird ausgeführt. Er besitzt die Rechenleistung des Prozessors.

- **READY.** Der Task wartet auf keine Systemressourcen außer der Rechenzeit und kann ausgeführt werden.
- **PENDED.** Der Task ist blockiert, weil er auf eine Systemressource wartet.
- **DELAYED.** Der Task schläft für eine bestimmte Zeitdauer.
- **SUSPENDED.** Der Task kann nicht ausgeführt werden. Zustandswechsel sind jedoch möglich. Dieser Zustand wird hauptsächlich zum so genannten Debuggen verwendet.

Jedem Task, der sich im Zustand **READY** befindet, kann durch Zuweisen von Rechenzeit in den Zustand **EXECUTING** wechseln. In Abbildung 2.24 werden die anderen möglichen Übergänge zwischen den Zuständen gezeigt. In VxWorks existieren zwei Prinzipien wie die Rechenzeit des Prozessors unter den einzelnen Tasks verteilt wird:

- **Preemptive-Priority-Scheduling.** Jeder Task besitzt eine Priorität. Der Task mit der höchsten Priorität wird ausgeführt. Andere Tasks mit niedriger Priorität werden notfalls unterbrochen.
- **Round-Robin-Scheduling.** Bei Tasks mit gleicher Priorität, wird die Rechenzeit gerecht unter den Tasks aufgeteilt. Jeder Task bekommt einen Zeitschlitz zugeordnet, in dem er ausgeführt wird.

2.5.2 Interrupts

VxWorks unterstützt, das schnelle Auswerten von Hard- und Softwareinterrupts durch ISRs³⁵. VxWorks kann jedem Interrupt eine C-Funktion zuordnen, die beim Eintreffen eines Interrupts aufgerufen wird. Da mit Hilfe der ISRs sofort auf externe Ereignisse reagiert werden muss, werden diese C-Funktionen nicht im Rahmen einer Task abgearbeitet, sondern in einer speziellen Interruptumgebung. Die Ausführung des aktuellen Tasks wird dazu ausgesetzt. Es findet jedoch kein Zustandswechsel des Tasks statt.

Um die Echtzeitfähigkeit des Systems nicht einzuschränken, sollten die ISR möglichst zeitunkritische Befehle und Funktionsaufrufe enthalten. In VxWorks Programmer's Guide [26], Seite 93, sind die Routinen aufgelistet, die in einer ISR aufgerufen werden können. Benötigt eine ISR zum Abarbeiten der C-Funktion zu viel Zeit, können die darauffolgenden Interrupts möglicherweise nicht ausgewertet werden und gehen deshalb verloren.

2.5.3 Kommunikation zwischen Tasks

Bei der Internetkamera werden drei Hilfsmittel zur Kommunikation zwischen Task verwendet:

- Gemeinsam genutzte Speicherbereiche
- Semaphoren³⁶.
- Message Queues

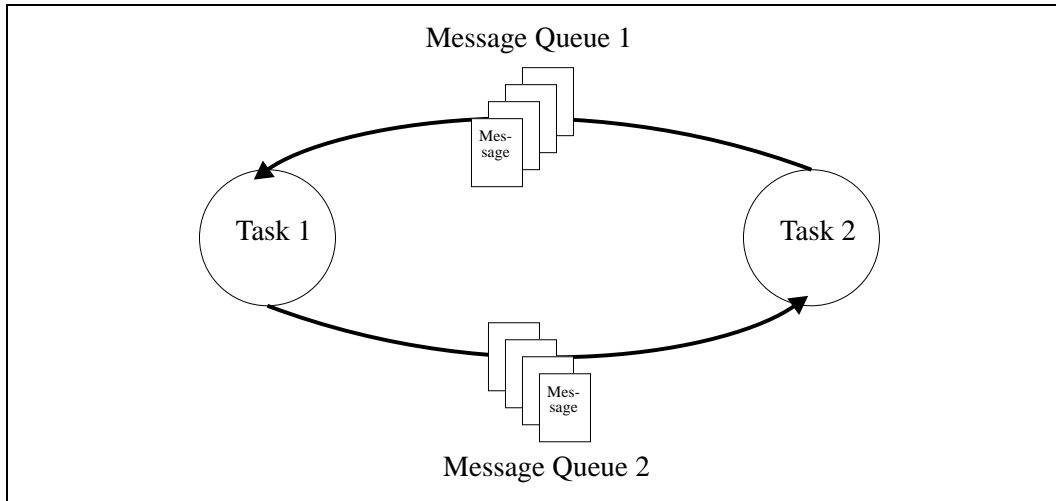
2.5.3.1 Gemeinsam genutzte Speicherbereiche

Es ist leicht einsichtig, dass über Datenstrukturen, die im Speicherbereich abgelegt werden, Tasks Informationen austauschen können. Globale Variablen, Ringspeicher, verkettete Listen, Zeiger oder andere Datenstrukturen können dazu verwendet werden. Damit es zu keiner Kollision kommt, muss ein wechselseitiger Ausschluss der Tasks beim Zugriff auf die gemeinsam genutzten Datenobjekte durchgeführt werden. Dies kann durch Deaktivieren anderer Tasks und Interrupts oder durch die Verwendung von Semaphoren realisiert werden.

³⁵ISR = Interrupt Service Routine

³⁶Semaphor = Zeichenträger, Signalmast

Abbildung 2.25: Kommunikation mittels Message Queues



Quelle: [26], Seite 71

2.5.3.2 Binäre Semaphoren

Die Verwendung von binären Semaphoren ist die schnellste Methode der Kommunikation zwischen Tasks. Ein binäres Semaphor ist ein Datenelement, das die Zustände „verfügbar“ oder „nicht verfügbar“ einnehmen kann.

Binäre Semaphoren können zur Synchronisation von Tasks und zum wechselseitigem Ausschluss von Tasks bei gemeinsam genutztem Speicherbereich verwendet werden. Wird das Semaphor von einem Task angefordert und ist das Semaphor verfügbar, besitzt dieser Task das Semaphor. Bis dieser Task das Semaphor wieder freigibt, ist es nicht verfügbar.

Wird ein Semaphor, das nicht verfügbar ist, von einem Task angefordert, kann der Task folgendermaßen reagieren:

- auf die Freigabe des Semaphors warten
- eine bestimmte Zeit auf die Freigabe des Semaphors warten (Timeout)
- nicht warten

2.5.3.3 Message Queues

Message Queues werden verwendet, um Nachrichten variabler Größe und Anzahl zwischen den Tasks auszutauschen. Die Nachricht des sendenden Tasks werden zwischengespeichert bis sie von dem empfangendem Task abgearbeitet werden können. Durch das Verwenden von zwei Message Queues kann eine Kommunikation in beide Richtungen realisiert werden (siehe Abbildung 2.25).

Erwartet ein Task die Nachricht eines anderen Tasks, die noch nicht gesendet wurde, kann sie folgendermaßen reagieren:

- auf das Senden der Nachricht warten
- eine bestimmte Zeit auf das Senden der Nachricht warten (Timeout)

- nicht warten

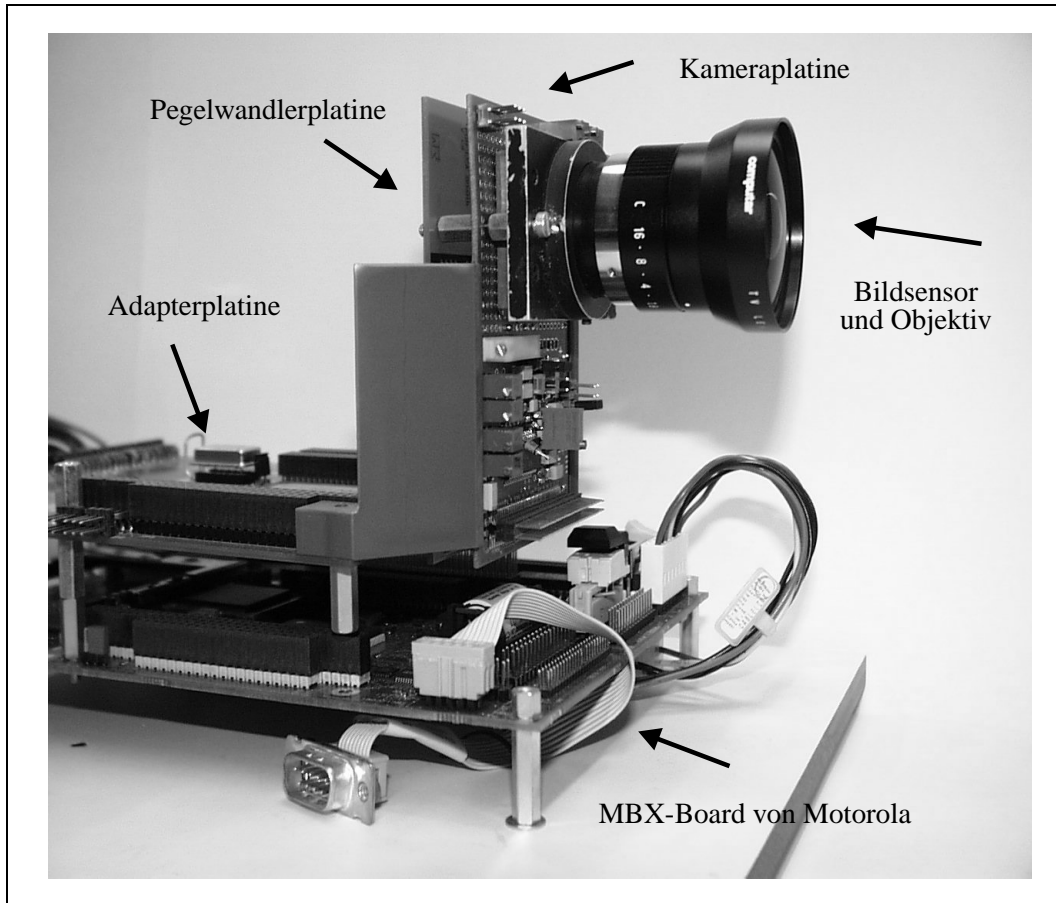
Mit Hilfe von Message Queues können Tasks Informationen austauschen, auf verschiedene Ereignisse reagieren oder synchronisiert werden.

Message Queues, binäre Semaphoren und gemeinsam genutzte Speicherbereiche können auch zur Kommunikation zwischen einer ISR und einem Task verwendet werden.

Kapitel 3

Konzeption und Realisierung der Internetkamera

Abbildung 3.1: Die verwendete Hardware der Internetkamera



3.1 Einführung

Die Internetkamera, die im Rahmen dieser Diplomarbeit entwickelt wird, basiert auf der Hardware, die in der vorangegangenen Studienarbeit mit dem Thema „Konzeption und Realisierung einer Entwicklungsplattform für eine ISDN-Kamera“ [40] entstanden ist. Abbildung 3.1 zeigt ein Foto der Hardware.

Grundlage des Aufbaus ist das MBX860-Mikroprozessorboard für eingebettete Systeme, das den MPC860¹-PowerQUICC²-Prozessor unterstützt. Dieses Prozessorboard kann mit dem Echtzeitbetriebssystem „VxWorks“ der Firma „Wind River Systems“ betrieben werden. Wind River stellt für das MBX860 ein fertiges Board Support Package (BSP³) zur Verfügung. Durch das BSP kann die Hardware auf dem Mikroprozessorboard, wie Mikrocontroller, externer Speicher, Ethernet oder serielle Schnittstelle, durch VxWorks genutzt werden.

Mit Hilfe eines Stecksystems kann das MBX-Board erweitert werden. Als Kamera wird eine Kameraplatine mit einem CIF⁴-CMOS⁵-Bildsensor verwendet, der im Fraunhofer Institut für Mikroelektronische Schaltungen

¹MPC860 = Motorola PowerPC 860

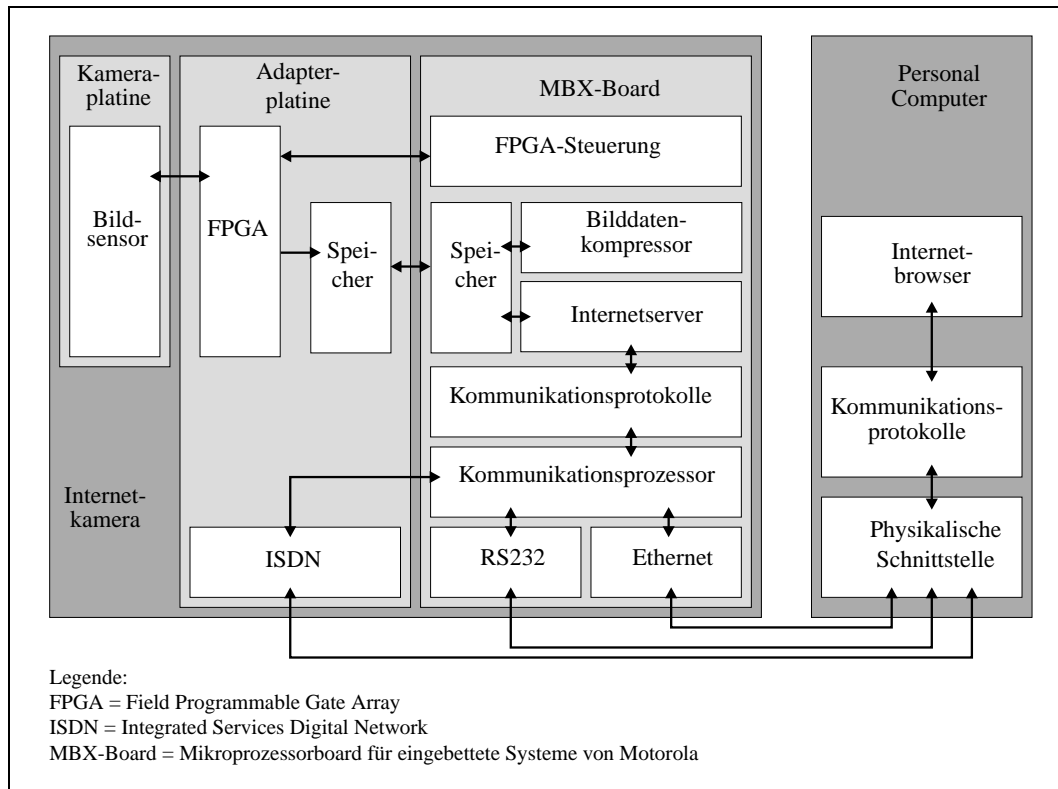
²QUICC = Quad Integrated Communication Controller

³BSP = Board Support Package

⁴CIF = CALTECH Intermediate Format

⁵CMOS = Complementary Metal-Oxide Semiconductor

Abbildung 3.2: Konzept des Systems „Internetkamera“



und Systeme (im Folgenden IMS genannt) entwickelt wurde. Um die Kamera mit dem Prozessorboard zu verbinden, wird eine Adapterplatine in Form eines Steckmoduls für das MBX-Board genutzt. Auf dieser Adapterplatine befinden sich ein FPGA⁶, SRAM⁷ und eine ISDN-Schnittstelle. Die Bilddaten des Sensors können mit Hilfe des FPGAs ausgelesen und in dem SRAM gespeichert werden.

Ziel dieser Diplomarbeit ist, Bilder der Kamera mit einem Internetbrowser zu visualisieren. Neben der bestehenden Ethernet- und der seriellen RS232-Schnittstelle, für die die notwendige Teilersoftware bereits durch das BSP bereitgestellt wird, soll die ISDN-Schnittstelle, die sich auf der Adapterplatine befindet, zur Bildübertragung verwendet werden.

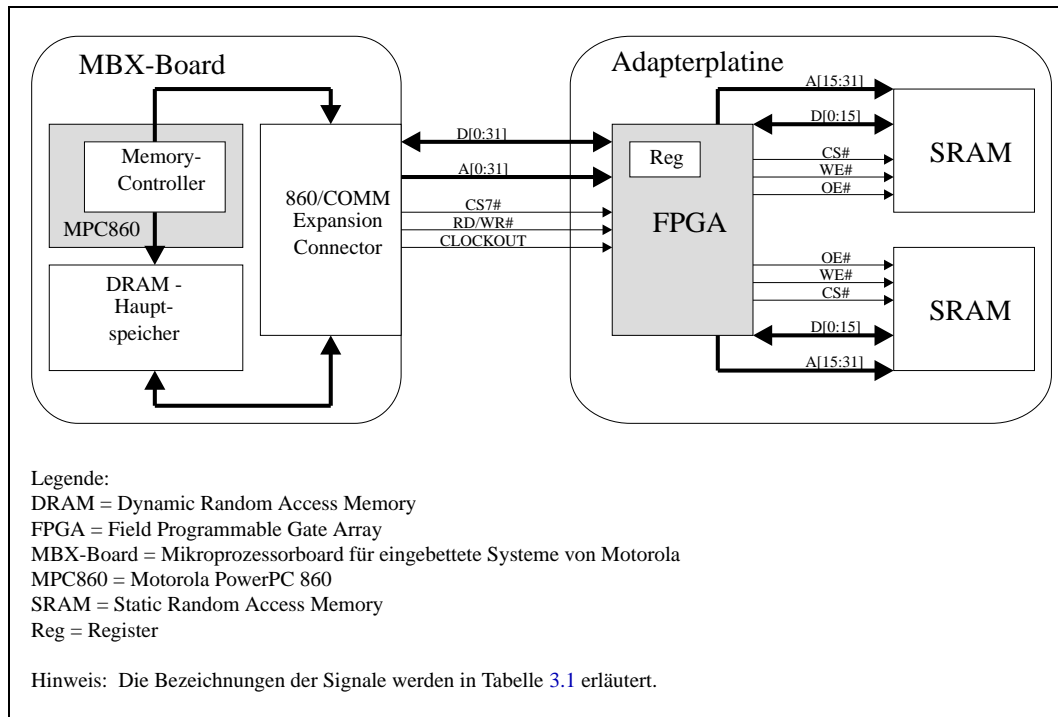
Um dieses Ziel zu erreichen, werden folgende Arbeitspakete im Rahmen der Diplomarbeit bearbeitet:

- Die Daten aus dem SRAM-Speicher auf der Adapterplatine werden in den Hauptspeicher des Prozessorboards übertragen.
- Bei einer Bildübertragung über ISDN, RS232 und Ethernet ist eine Kompression der Bilddaten sinnvoll, um die zu übertragende Datenmenge zu reduzieren. Daher wird ein Bildkompressionsalgorithmus auf dem MPC860 implementiert.
- Ein Internetserver wird dem System hinzugefügt.

⁶FPGA= Field Programmable Gate Array

⁷SRAM = Static Random Access Memory

Abbildung 3.3: Datentransport vom SRAM in den Hauptspeicher des MBX- Boards



- Kommunikationsprotokolle wie TCP, IP und PPP werden eingebunden.
- Die ISDN-Hardware wird mit Hilfe des im MPC860 integrierten Kommunikationsprozessormoduls angesprochen.
- ISDN-D-Kanal-Protokolle werden implementiert und mit den bestehenden Kommunikationsprotokollen verknüpft.
- Das Kamerabild muss in einem Internetbrowser visualisiert und periodisch aktualisiert werden.

Mit Hilfe der bereits vorhandenen und dieser neuen Komponenten kann ein funktionierendes Gesamtsystem „Internetkamera“ erstellt werden. Abbildung 3.2 zeigt das Konzept des Gesamtsystems.

In den folgenden Abschnitten wird die Konzeption und Realisierung der einzelnen Arbeitspakete dokumentiert.

3.2 Auslesen der Bilddaten aus dem SRAM

Um die Bilddaten mit dem MPC860 weiterzuverarbeiten, müssen sie vom SRAM in den Hauptspeicher des MBX-Board übertragen werden. Das SRAM befindet sich auf der Adapterplatine. Es besteht aus zwei Speicherbausteinen von je 512 KByte mit einer Wortbreite von 16 Bit. Alle Steuer-, Adress- und Datenleitungen der SRAMs sind mit I/O-Pins des FPGAs verbunden.

Der Datentransport zwischen Adapterplatine und MBX-Board geschieht über den „860/COMM Expansion Connector“ des MBX-Boards. Abbildung 3.3 zeigt alle verwendeten Signale. Tabelle 3.1 erläutert deren

Tabelle 3.1: Verwendete Signale beim Auslesen des SRAMs

Abk.	x-aktiv	engl. Bezeichnung	Beschreibung
CS#	low	chip-select	Die entsprechende SRAM-Speicherbank wird selektiert.
OE#	low	output enable	Die Ausgänge der SRAMs treiben die Datenleitungen.
WE#	low	write enable	Das SRAM kann beschrieben werden.
CS7#	low	chip-select Nr. 7	CS-Leitung, die auf dem 860/COMM Expansion Connector zur Verfügung steht.
RD/WR#	-	read/ not write	high = MPC will Daten lesen; low = MPC will Daten schreiben;
CLOCKOUT	-	clock output	Bus-Takt des 860/COMM Expansion Connector
D[0:31]	-	data	Datenleitungen des MPC860. Achtung: LSB = D31 und MSB = D0 !
A[0:31]	-	address	Adressleitungen des MPC860.

jeweilige Funktion. Alle zusätzlichen Signale dieses Steckers, die auf I/O-Pins des FPGAs gelegt wurden, können der Hardwarebeschreibung der Adapterplatine in der Studienarbeit ([40], Konzeption Seite 16 ff u. Schaltplan Seite 36) entnommen werden.

Wie in Abbildung 3.3 gezeigt, besteht keine direkte Verbindung zwischen dem MPC860 und den SRAMs. Der MPC860 kann nur indirekt über das FPGA auf den Speicher zugreifen. Der MPC860 besitzt einen integrierten Memory-Controller, der das Timing für einen Speicherzugriff generieren kann.

3.2.1 Erweiterung des Verilog-Programms

Um die Signale des Memory-Controllers für den Datentransport vom SRAM in den DRAM-Hauptspeicher des MBX-Boards zu verwenden, wird das FPGA folgendermaßen programmiert:

- Bei einem Lesezugriff des Memory-Controllers auf eine Adresse im Adressbereich des SRAM-Speichers wird das FPGA „unsichtbar“. Steuer, Adress- und Datenleitungen des SRAMs werden auf die entsprechenden, vom Memory-Controller generierten, Signale durchgeschaltet.
- Bei einem Schreibzugriff des Memory-Controllers wird unabhängig von der anliegenden Adresse ein 32-Bit Register im FPGA beschrieben.
- Bei einem Lesezugriff des Memory-Controllers auf eine Adresse außerhalb des Adressbereichs des SRAM-Speichers wird auf den Datenleitungen der Inhalt des 32-Bit-Registers zurückgegeben.

Die Aktionen des FPGAs werden durch den Wert des 32-Bit-Registers gesteuert. Durch den Inhalt dieses Registers wird dem FPGA mitgeteilt, ob es auf den SRAM-Speicher zugreifen und welche Speicherbank es zur Speicherung der Bilddaten verwenden darf.

Der Aufbau des 32-Bit-Registers ist in Tabelle 3.2 zu sehen. Die verwendeten Kürzel werden im Folgenden beschrieben:

- PREROW (Previously Read Rows)
8-Bit Integer = Die Zeit (in Anzahl der zuvor ausgelesenen Zeilen), die zur Integration jedes Pixels verwendet wird (siehe Studienarbeit [40], S. 53, Abschnitt 3.4.0.1). PREROW wird verwendet, um die Helligkeit des Kamerabildes zu steuern.

Tabelle 3.2: Der Aufbau des 32-Bit-Registers

Bits	31 bis 16				15 bis 8			
Funktion	Reserviert				PREROW			
Reset	0				0			
R/W	R				W			

Bits	7 bis 6	5	4	3	2	1	0
Funktion	Reserviert	PICRDY	RWPIC	Reserviert	WPIC	CHOBNK	CMPCBNK
Reset	0	0	0	0	0	0	0
R/W	R	R	R	R	R/W	R/W	R/W

- **PICRDY (Picture Ready)**
1 = Zeigt an, dass ein fertiges Bild der Kamera im SRAM-Speicher liegt.
0 = Kein Bild im Speicher.
- **RWPIC (Received Write Picture)**
1 = Das FPGA hat die Aufforderung erkannt, ein Bild in das SRAM zu schreiben.
0 = Anforderung wurde noch nicht erkannt.
- **WPIC (Write Picture)**
1 = Anforderung des MPC860 an das FPGA, ein Bild in den SRAM-Speicher zu schreiben.
0 = PICRDY und RWPIC werden auf 0 gesetzt. Die Adress- und Steuerleitungen des MPC860 sind mit beiden SRAM-Bausteinen verbunden.
(MPC → SRAM 1 und SRAM 2)
- **CHOBNK (Choose Bank)**
1 = Die Adress-, Daten und Steuerleitungen von Speicherbank 1 sind mit den Adress-, Daten- und Steuerleitungen des FPGA verbunden. Die Adress- und Steuerleitung des MPC860 sind mit Speicherbank 2 verbunden.
(FPGA ↔ SRAM 1, MPC → SRAM 2)
0 = Die Adress-, Daten- und Steuerleitungen von Speicherbank 2 sind mit den Adress-, Daten- und Steuerleitungen des FPGA verbunden. Die Adress- und Steuerleitung des MPC860 sind mit Speicherbank 1 verbunden.
(FPGA ↔ SRAM 2, MPC → SRAM 1)
- **CMPCBNK (Change MPC860 Bank)**
1 = Die Datenleitungen des MPC860 sind beim Lesezugriff mit Speicherbank 1 verbunden.
(MPC ← SRAM 1)
0 = Die Datenleitungen des MPC860 sind beim Lesezugriff mit Speicherbank 2 verbunden.
(MPC ← SRAM 2)

Das Verilog-Modul „cam4mbx.v“ ist auf der CD im Anhang A im Verzeichnis „Verilog“ zu finden. Mit Hilfe dieses Moduls werden die Logikbausteine im FPGA so programmiert, dass sie obige Funktionalität erfüllen. Es ist eine erweiterte Version des Moduls „cam4mbx.v“, welches in der Studienarbeit erstellt wurde.

3.2.2 FPGA-Steuerung durch den MPC860

Das Flussdiagramm 3.4 zeigt die Steuerung des FPGAs mit Hilfe des 32-Bit-Registers durch den MPC860. Die zwei SRAM-Speicherbänke werden im Doppelpufferbetrieb verwendet, d. h. während der MPC860 einen

Speicher ausliest, wird der andere Speicher durch das FPGA beschrieben. Der Doppelpufferbetrieb dient einem schnellen und kontinuierlichen Auslesen der Bilddaten.

Die Bilddaten der Kamera können vom MPC860 nicht so schnell verarbeitet werden, wie sie mit Hilfe des Doppelpufferbetriebs theoretisch ausgelesen werden können. Es kann daher aus Kostengründen sinnvoll sein, nur einen SRAM-Speicher zu verwenden. Das Flussdiagramm 3.5 zeigt für diesen Fall die Ansteuerung des FPGAs durch den MPC860.

Die Bilddaten müssen periodisch ausgelesen werden. Für diese Aufgabe wird ein eigenständiger Task gestartet, der im Folgenden FPGA-Task genannt wird.

Abbildung 3.4: Auslesen des Speichers im Doppelpufferbetrieb

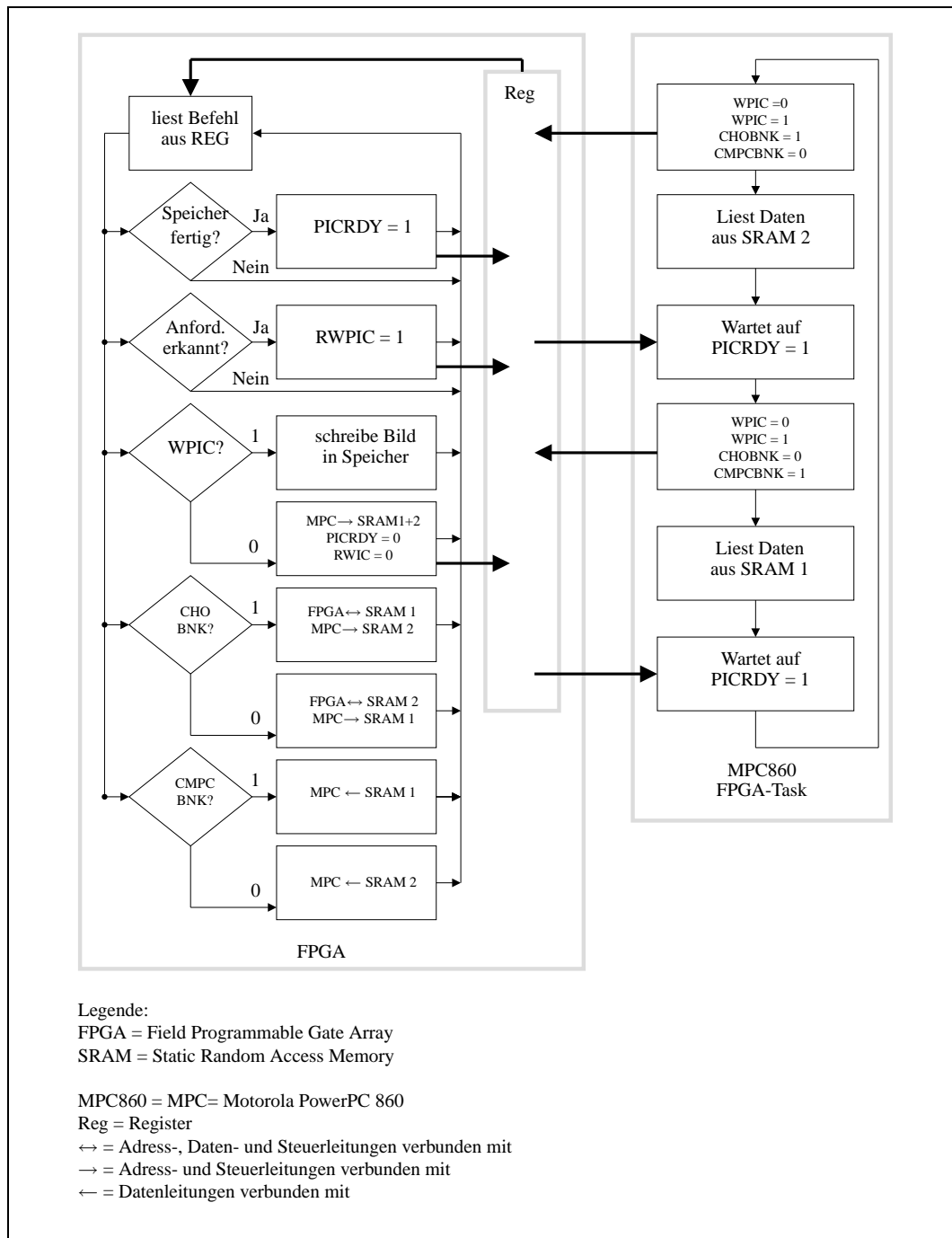
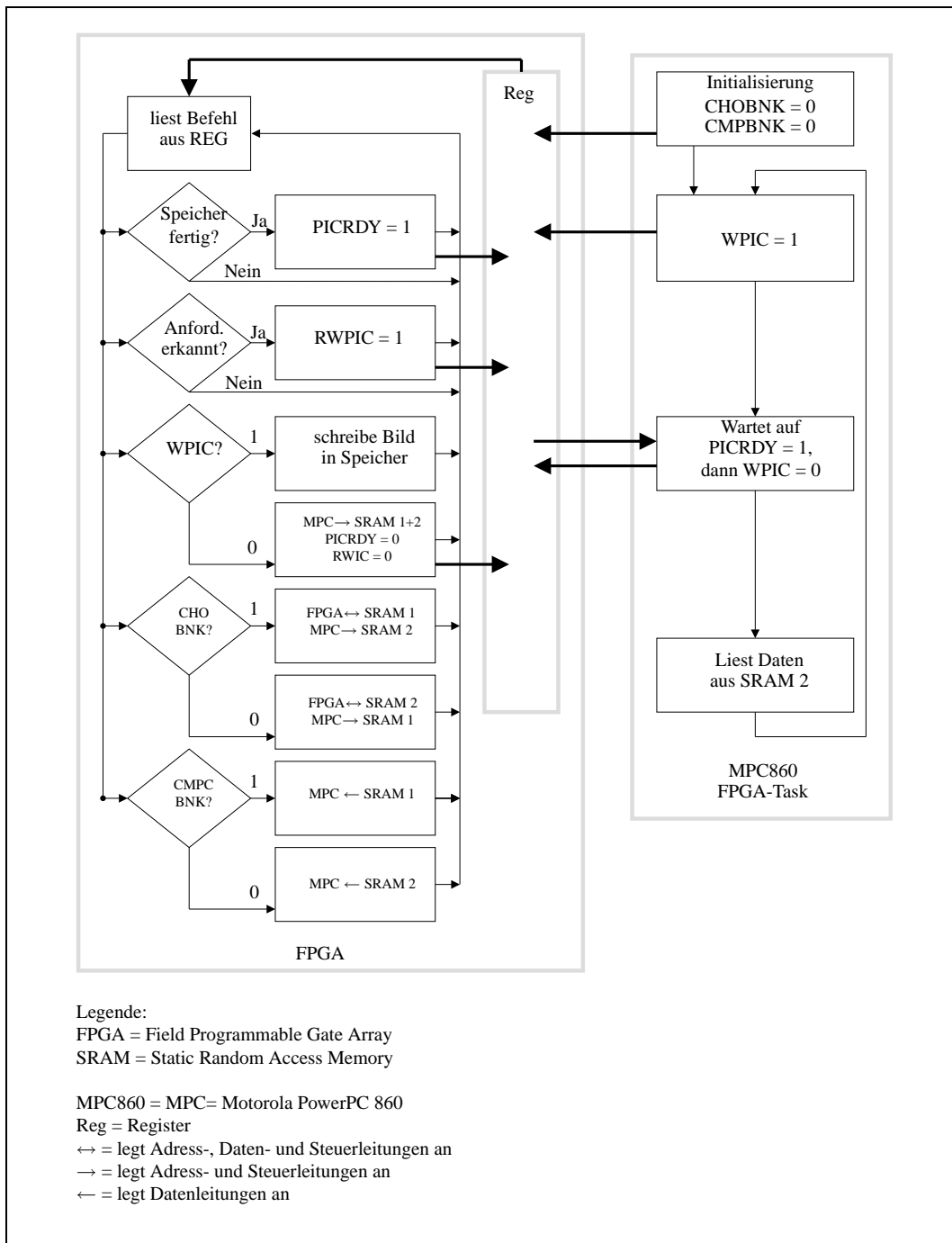


Abbildung 3.5: Auslesen des Speichers bei Verwendung einer Speicherbank



3.2.3 Der Memory-Controller

Der Memory-Controller umfasst drei Basis-Maschinen:

- General-purpose chip-select machine (GPCM)
- User-programmable machine A (UPM A)
- User-programmable machine B (UPM B)

Der Memory-Controller des MPC860 kann bis zu 8 Speicherbänke verwalten. Für die Ansteuerung kann eine dieser drei Maschine individuell für jede Speicherbank gewählt werden. Die GPCM dient zur Ansteuerung von Speichern mit einfachem Ansteuerungstiming, wie z. B. SRAMs, EPROMs⁸, oder ROMs⁹. Die zwei UPMs dienen zur Ansteuerung von Speichern mit komplizierterem Timing wie DRAM oder SDRAM¹⁰. Da es sich bei dem Speicher auf der Adapterplatine um SRAM handelt, wird im Folgenden nur der GPCM berücksichtigt.

Um die GPCM zu programmieren, werden folgende Register des MPC860 verwendet:

- BR0 bis BR7 (Base Register 0 bis 7) werden zur Einstellung der Basisfunktionen für jede Speicherbank verwendet.
- OR0 bis OR7 (Option Register) werden zur Einstellung von Optionen verwendet. Hauptsächlich dienen sie zum Einstellen des richtigen Timings für den Speicherbaustein.

Die Bedeutung der einzelnen Bits im BR und OR können dem MPC860-Handbuch [24] entnommen werden. Das BR-Register wird in Tabelle 15-14 auf Seite 15-71 und das OR-Register in Tabelle 15-15 auf Seite 15-73 beschrieben.

Für jede Speicherbank existiert eine Chip-Select-Leitung (CS0 bis CS7). Jeder Leitung CS_x ist ein Registerpaar bestehend aus BR_x und OR_x zugeordnet (mit x = 0 bis 7).

3.2.4 Die Register des MPC860

Die Register des MPC860 werden auf seinen internen Speicher (IMM = Internal Memory Map) abgebildet und sind dort für den Programmierer zugänglich. Die Offsets der einzelnen Register zur Startadresse des internen Speichers sind im MBX-Handbuch [24] in Tabelle 3-1 auf Seite 3-1 aufgelistet. Die Startadresse des internen Speichers ist durch das BSP von Wind River für das MBX-Board mit IMM_R = 0xfa200000 angegeben (Quelle: [22], Tabelle 2-1 auf Seite 2-2).

3.2.5 Speicherorganisation des MBX-Boards

Vom MBX-Board werden durch das BSP von Wind River alle acht Chip-Select-Leitungen verwendet. In Tabelle 3.3 wird aufgelistet, welche CS-Leitung für welchen Speicher genutzt wird. Die Anfangswerte, die das BSP in die entsprechenden BR- und OR-Register schreibt, können Tabelle 2-3 aus [22], Seite 2-5, entnommen werden. Damit ergibt sich die Speicherbelegung des MBX-Boards nach Tabelle 2-4 aus [22], Seite 2-6.

Der MPC860 lädt sein minimales Betriebssystem normalerweise immer von dem Speicher, der mit CS0 verknüpft ist. Durch den Jumper J4 auf dem MBX-Board kann zwischen dem On-Board-Flash und dem

⁸EPROM = Erasable Programmable Read-Only Memory

⁹ROM = Read-Only Memory

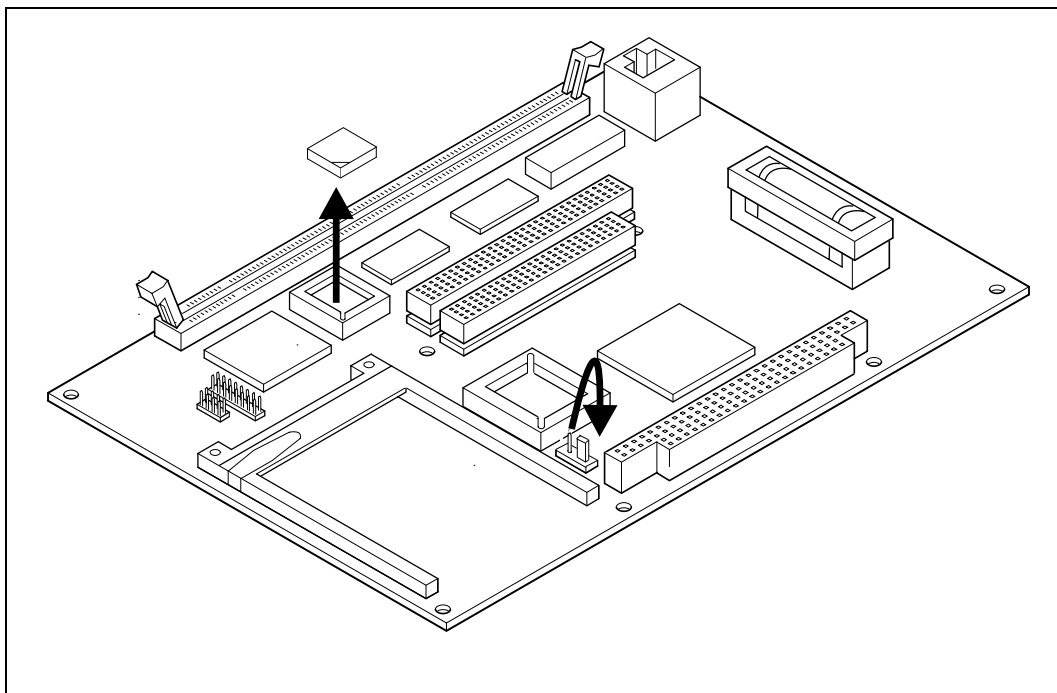
¹⁰SDRAM = Synchronous Dynamic Random-Access Memory

Tabelle 3.3: Verwendete Chip-Select-Leitungen

Chip-Select	Speicher
0	On-Board-Flash oder gesockeltes Flash
1	On-Board DRAM
2	DIMM RAM Bank 0
3	DIMM RAM Bank 1
4	NVRAM
5	PCI I/O und PCI Memory
6	PCI Bus Bridge Registers
7	On-Board-Flash oder gesockeltes Flash

Hinweis: vgl. Abkürzungsverzeichnis

Abbildung 3.6: Entfernen des Flash und Umstecken des Jumpers J4



gesockelten Flash gewählt werden. Der Speicherbaustein, von dem nicht gebootet wird, bekommt CS7 zugewiesen. In der Originaleinstellung bootet der MPC860 vom gesockelten Flash.

Das Signal CS7 kann alternativ vom 860/COMM Expansion Connector verwendet werden. Dazu muss das gesockelte Flash XU1 entfernt und der Jumper J4 umgesteckt werden, so dass vom On-Board-Flash gebootet wird (siehe Abbildung 3.6). Um das SRAM auf der Adapterplatine auszulesen, wird CS7 verwendet (siehe Abbildung 3.3). Daher müssen diese Veränderungen der Originaleinstellungen des MBX-Boards vorgenommen werden.

3.2.6 Programmieren des Memory-Controllers

Um das SRAM auf der Adapterplatine auszulesen, muss der Memory-Controller entsprechend programmiert werden. Das SRAM hat die gleiche Größe wie das entfernte Flash (512 KByte). Deshalb muss an der Speicheradressenbelegung des MBX-Boards nichts geändert werden. Der Memory-Controller wird im Folgenden so programmiert, dass bei einem Lese- oder Schreibzugriff auf den Speicherbereich des entfernten Flash nun das SRAM bzw. das FPGA auf der Adapterplatine angesprochen wird.

Dazu müssen die BR7- und OR7-Register programmiert werden, die der CS7-Leitung zugewiesen sind. Dies geschieht durch folgende Zeilen C-Kode:

Definitionen aus BSP (ppc860Siu.h):

```
# define BR_R 0x00000001 /* Bank Valid */
# define BR_PS_16 0x00000800 /* 16 bit port size */
/* CS is output 1/2 a clock later */
# define OR_ACS_DIV2 0x00000600
# define OR_BI 0x00000100 /* Burst inhibit */
# define OR_SCY_0_CLK 0x00000000 /* 0 clock cycles wait states */
/* Option Reg bank 7*/
# define OR7(base) (CAST(VUINT32 *) (base + 0x013C))
/* Base Reg bank 7 */
# define BR7(base) (CAST(VUINT32 *) (base + 0x0138))
```

C-Programm:

```
# define SRAMBASEADDR 0xFE000000

UINT32 immrVal = vxImmrGet();
*BR7(immrVal) = SRAMBASEADDR | BR_PS_16 | BR_V;
*OR7(immrVal) = OR_ACS_DIV2 | OR_BI | OR_SCY_0_CLK;
```

`vxImmrGet()` liefert die Startadresse der Internal Memory Map (IMM) zurück. `BR7()` und `OR7()` sind Definitionen des BSP, die den Offset der Adresse der Register zur Startadresse der IMM hinzu addieren und somit die Adresse der Register liefern.

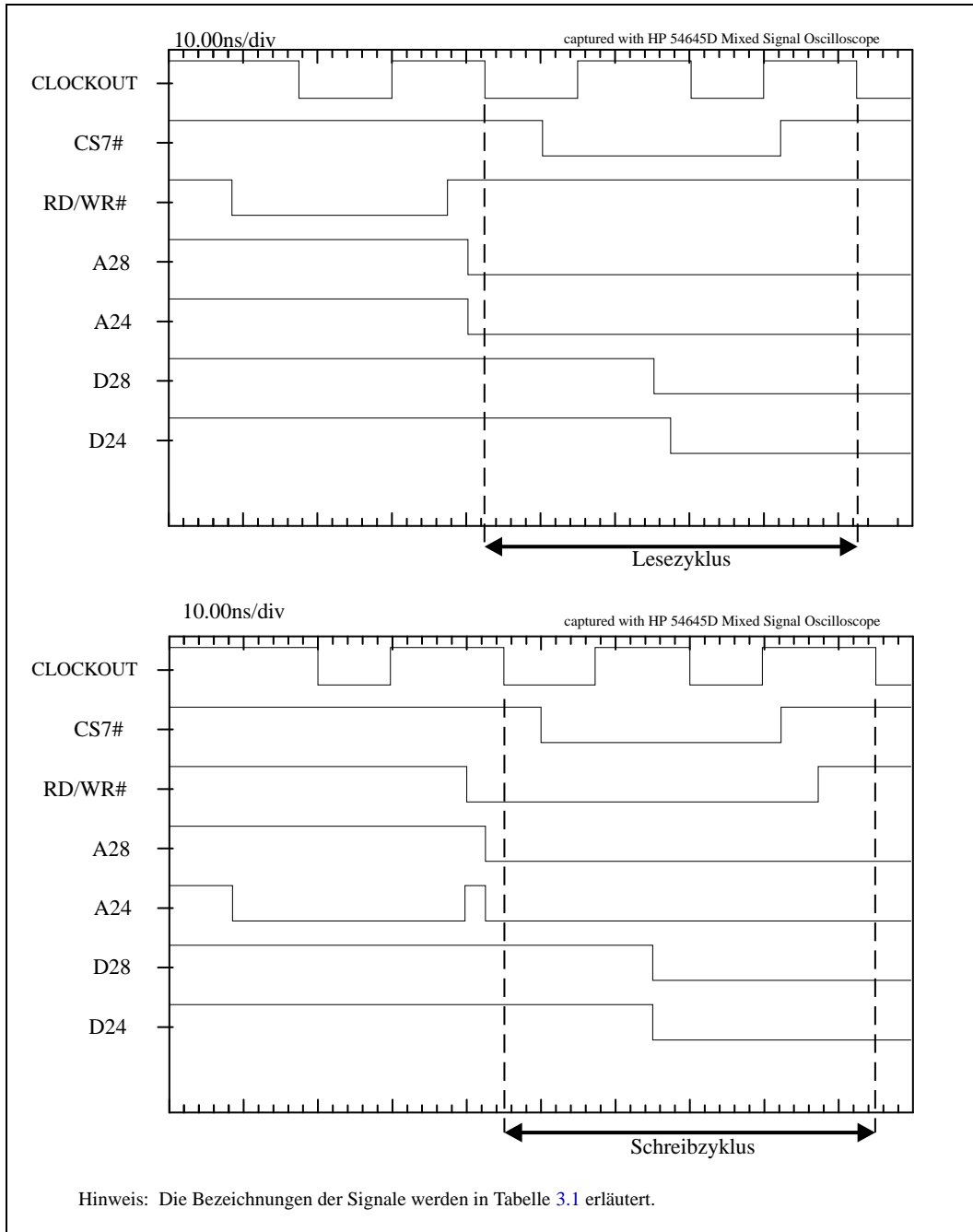
Abbildung 3.7 zeigt die resultierenden Signale auf dem 860/COMM Expansion Connector für einen Lese- und einen Schreibzugriff. Es sind exemplarisch die Daten- und Adressleitungen 24 und 28 herausgegriffen worden. Die Signale auf den Adressleitungen werden bei der fallenden Flanke von CS vom SRAM angenommen. Die Daten sind bei der steigenden Flanke von CS gültig. Es ist deutlich die Verzögerung der fallenden Flanke des CS-Signals um eine halbe Taktlänge zu erkennen, was den obigen Einstellungen im OR7 entspricht.

Die Bilddaten der Kamera können nun in den Hauptspeicher der MBX-Boards übertragen werden.

3.3 Bilddatenkompressor

Die sichtbare Bildmatrix des CIF-CMOS-Sensors hat $360 * 288$ Pixel. Die Grauwerte jedes Pixels werden mit einer Genauigkeit von 12 Bit von der Kamera ausgegeben. Da es sich bei dem SRAM um einen 16 Bit

Abbildung 3.7: Timing Diagramm für Lese- und Schreibzyklus des MPC860



organisierten Speicherbaustein handelt, wird für jedes Pixel 16 Bit Speicherplatz verwendet. Damit ergibt sich die Größe des nicht komprimierten Bildes zu:

$$(360 * 288) * 16 \text{ Bit} = 1658880 \text{ Bit pro Bild}$$

$$\frac{1658880 \text{ Bit}}{8 \frac{\text{Bit}}{\text{Byte}} * 1024} = 202.5 \text{ KByte pro Bild}$$

Eine erste Datenreduktion kann durch die Verwendung einer geringeren Genauigkeit der Grauwerte erreicht werden. Bei einer Genauigkeit von 8 Bit pro Pixel beträgt der Speicherbedarf eines Bildes nur noch 101.25 KByte.

Bei einer idealen Datenübertragung dieser 101.25 KByte Bilddaten über einen ISDN B-Kanal mit einer Bandbreite von 64 Kbit/s, ergibt sich eine minimale Übertragungszeit von:

$$\frac{101.25 \text{ KBytes pro Bild} * 8 \frac{\text{Bit}}{\text{Byte}}}{64 \frac{\text{KBit}}{\text{s}}} \approx 12.66 \text{ s pro Bild}$$

Eine größere Bildwiederholfrequenz ist für viele Anwendungen wünschenswert. Dies kann durch einen Kompressionsalgorithmus erreicht werden, der die zu übertragene Datenmenge reduziert.

Es kann zwischen Algorithmen zur Kompression von Einzel- und Bewegtbildern unterschieden werden. Bekannte standardisierte Bildformate für komprimierte Einzelbilder sind:

- JPEG (Joint Photographic Experts Group) Der Algorithmus von JPEG wurde bereits ausführlich im Abschnitt 2.4 vorgestellt. Es können Kompressionsraten von 25:1 erreicht werden.
- GIF (Graphics Interchange Format) Dieses Bildformat ist verlustfrei, kann aber nur 8 Bit (256 Farben) pro Pixel speichern. Es wird ein LZW¹¹- Kodierer der Firma Unisys verwendet. Dieser Algorithmus ist patentgeschützt. Die Kompressionsrate erreicht maximal 5:1.
- PNG (Portable Network Graphics Format) Dieser aktuelle, lizenzfreie GIF- Nachfolger sieht neben dem 256-farbigen GIF-Paletten-Modell mehrere Graustufen und Echtfarbformate vor.

Für die Übertragung von bewegten Bildern werden häufig folgende Standards verwendet:

- H.261 (ITU-T H.261 Video Codec for Audiovisual Services at p x 64 Kbits) Der Standard für den Video-Kodierer eines Bildtelefons bzw. einer Videokonferenzübertragung nach H.320¹². Als Bildformat wird das CIF-Format verwendet. Neben dem Ausnutzen der Redundanz der Einzelbilder kann durch Bewegungskompensation der Unterschied zwischen den einzelnen Bildern weiter verringert werden. Die Abweichungen zwischen den Bildern werden ähnlich dem JPEG-Algorithmus komprimiert und übertragen. Die Kombination der beiden Verfahren führt zu großen Kompressionsraten des Videostrusses.
- MPEG (Moving Pictures Experts Group) Dieser Standard umfasst Video- und Audiosignale. Der Kompression von Bewegtbildern im MPEG-1-Standard (ISO/IEC 11172) basiert auf dem Algorithmus von H.261. Es wird allerdings eine höhere Qualitätsanforderung gestellt. Es können Datenraten von bis zu 1,5 MBit/s erreicht werden. Bei MPEG-2 (ISO/IEC 13818) wird diese Datenrate auf 40 MBit/s erhöht und größere Bildformate sind zugelassen.

Leider werden nicht alle Formate von den weit verbreiteten Internetbrowsern „Internet Explorer“ der Firma Microsoft und „Netscape Navigator“ der Firma Netscape unterstützt. Tabelle 3.4 bietet eine Übersicht über die unterstützten Grafikformate der Versionen 3.x bis zur aktuellen Version 5.0 des „Netscape Navigator“ und des „Internet Explorer“.

¹¹LZW = Lemple, Welch, Zic (Namen der Entwickler des Komprimierungsalgorithmus)

¹²H.320 = ITU-T H.320

Tabelle 3.4: Unterstützte Grafikformate der Generationen von Internetbrowsern

Internet Explorer und Netscape Navigator	JPEG	GIF	PNG	MPEG	H.261
Version 3.x	✓	✓			
Version 4.x	✓	✓	✓		
Version 5.0	✓	✓	✓	✓	

Da der GIF-Kompressor patentgeschützt ist, kommen als Einzelbildformate JPEG und PNG in Frage. Das JPEG-Format wird von allen Browsern und im Gegensatz zu PNG von Java¹³(Version 1.1.x und höherer) unterstützt. Daher ist dieses Format die erste Wahl für die Internetkamera.

3.3.1 JPEG-Komprimierer

Im IMS existierte bereits ein JPEG-Kompressor für das Betriebssystem VxWorks. Diese JPEG-Bibliothek wurde in der Diplomarbeit von T. Stevens mit dem Thema „Nutzung von Intra- und Internetmechanismen für Eingebettete Systeme“ [36] verwendet. Er basiert auf dem frei verfügbaren JPEG-Kompressor (Version 6a) der Independent JPEG Group, der im Internet unter „ftp.uu.net:/graphics/jpeg/jpegsrc.v6a.tar.gz“ zu finden ist.

Die JPEG-Bibliothek wird über einige Funktionsaufrufe in eigene Anwendungen eingebunden. Dabei können dem JPEG-Kompressor mehrere Optionen mitgeteilt werden. Für die Anwendung als Bildkompressor für die Internetkamera müssen die Optionen der JPEG-Bibliothek so gewählt werden, dass ein Optimum zwischen Bildqualität, Kompressionsrate und benötigter Zeit gefunden wird.

Die Bildqualität kann mit einem Zahlenwert zwischen 0 (schlechteste) und 100 (beste) gewählt werden. Der Standardwert ist 75. Der Zahlenwert beeinflusst die verwendeten Quantisierungs-Koeffizienten des JPEG-Kompressors. Mit Hilfe dieser Koeffizienten wird (wie in Abschnitt 2.4 beschrieben) die Bildqualität bzw. die Größe der JPEG-Datei verändert. Bei der Internetkamera wird der Koeffizientenwert 65 verwendet.

Um die Geschwindigkeit des JPEG-Kompressors zu erhöhen, können vorberechnete Werte des Kosinus für die DCT verwendet werden (vergleiche Abschnitt 2.4). Durch Optionen kann der JPEG-Bibliothek mitgeteilt werden, ob diese Werte ganzzahlig oder als Gleitkommazahl für die Berechnung eingesetzt oder jedesmal neu berechnet werden. Im Allgemeinen ist die Berechnung der DCT mit ganzzahligen, vorberechneten Werten die schnellste Methode. Daher wurde sie bei dem JPEG-Komprimierer der Internetkamera verwendet.

Der JPEG-Kompressor muss ständig neue Bilder komprimieren. Daher läuft er in einem eigenen Task, der im folgenden JPEG-Task genannt wird .

Der C-Quellcode zum Einbinden der JPEG-Bibliothek kann in „webcam.c“ im Verzeichnis „webcam“ auf der CD im Anhang A gefunden werden. Der Quellcode der JPEG-Bibliothek und deren Dokumentation sind in den Archiven „JPEG.zip“ und „JPEGDOC.zip“ im Verzeichnis „JPEG Bibliothek“ auf der CD vorhanden.

¹³Java = Programmiersprache, die von der Firma „Sun“ entwickelt wurde.

3.4 Internetserver

Der Internetserver der Internetkamera ist ein minimaler HTTP-Server, der nur die Methode GET unterstützt (siehe Tabelle 2.3 in Kapitel 2.2.5).

Beim Starten der Internetkamera wird der HTTP-Server initiiert. Dateien, die der Client später vom HTTP-Server abrufen können soll, werden in den Arbeitsspeicher des MBX-Boards geladen. Dateiname, Speicheradresse und die Länge der Dateien werden dem Server mitgeteilt. Der HTTP-Server kommuniziert mit dem Client über eine Socketverbindung. Dazu werden die Funktionen aus der Socket-Bibliothek von VxWorks verwendet. Nach dem Erstellen und Konfigurieren der Socket-Verbindung durch die Funktionsaufrufe `socket()`, `setsockopt()` und `bind()`, wird dem Client vom Server durch den Funktionsaufruf `connect()` erlaubt, einen Verbindungswunsch mitzuteilen. Ein eigenständiger Task wird gestartet, der auf die Anfrage des Clients durch Aufrufen der Funktion `accept()` wartet und diese gegebenenfalls abarbeitet.

Im Folgenden wird beispielhaft die HTTP-Anfrage des Netscape Browsers Version 4.51 gezeigt. Der theoretische Aufbau einer Client-Anfrage wurde bereits in Kapitel 2.2.5 beschrieben.

```
message GET /aus.jpg?prerow=243 HTTP/1.0
Connection: Keep-Alive
User-Agent: Mozilla/4.51 [de] (WinNT; I)
Pragma: no-cache
Host: 192.44.3.200
Accept: image/gif, image/x-xbitmap, image/jpeg, \
image/pjpeg, image/png, */*
Accept-Encoding: gzip
Accept-Language: de
Accept-Charset: iso-8859-1,*,utf-8
```

Hat der HTTP-Server diese Datei im Speicher, beantwortet er die Anfrage mit:

```
HTTP/1.0 200 OK
Content-type: image/jpeg
Content-Length: 9564
<JPEG-kodierte Bilddaten>
```

Eine Besonderheit ist die Request-URL des Clients. Normalerweise ignoriert ein HTTP-Server die Zeichen nach einem Fragezeichen. Der HTTP-Server der Internetkamera verwendet die zusätzliche Information nach dem Fragezeichen zur Steuerung der Helligkeit des CMOS-Sensors, indem der Wert der Variablen PREROW übergeben wird. Dieser Wert wird der Steuerung des CMOS-Sensors durch das in Abschnitt 3.2.1 dargestellte Verfahren übermittelt. Auf diese Weise kann der Benutzer des Browsers die Internetkamera fernsteuern. Folgende URL setzt PREROW auf den Wert 200, wobei 192.44.3.200 als IP-Adresse der Internetkamera angenommen wird:

`http://192.44.3.200/aus.jpg?prerow=200`

Dieses Verfahren der Kommunikation zwischen HTTP-Server und Client ist nicht Bestandteil des HTTP-Protokolls, sondern eine zusätzliche proprietäre Vereinbarung zwischen Client und Server, die an die GET-Methode des CGI-Interface erinnert. Internetserver, die CGI¹⁴ unterstützen, könne Informationen des Clients nach den CGI-Spezifikationen an ein CGI-Skript weiterleiten und Informationen vom Skript entgegennehmen. CGI wird in der Praxis bei fast allen Suchmaschinen im Internet eingesetzt. Nach der Eingabe des Suchbegriffs

¹⁴CGI = Common Gateway Interface

durch den Anwender erfolgt eine Anfrage nach der HTML-Seite mit den Treffern der Suche. Dem Server wird durch die Information nach dem Fragezeichen der Suchbegriff des Anwenders mitgeteilt. Der Server stellt mit Hilfe des CGI-Skripts die Seite mit den Treffern der Suche zusammen und sendet seine Antwort an den Client. Folgende URL wird z.B. von der bekannten Suchmaschine „YaHoo“ bei Eingabe der Suchbegriffe Internet und ISDN aufgerufen.

```
http://search.yahoo.com/bin/search?p=internet+ISDN
```

3.4.1 Synchronisation und Geschwindigkeitsoptimierung

Die Bildwiederholfrequenz der Internetkamera soll möglichst groß sein. Um bei jeder Anfrage des Clients ein neues Bild zu übertragen, ist es sinnvoll, den Task des HTTP-Servers, den JPEG-Task und den FPGA-Task zu synchronisieren. Sonst kann es besonders bei schnellen Übertragungsmedien, wie z.B. Ethernet, passieren, dass ein Bild mehrfach zum Client übertragen wird.

Eine Mehrfachübertragung tritt auf, wenn ein Client schneller eine neue Anfrage startet als der JPEG-Task ein neues Bild komprimiert hat. Wenn die Tasks nicht synchronisiert sind, wird in diesem Fall der HTTP-Task das alte Bild noch einmal senden.

Die Kommunikation zwischen den Tasks geschieht mit Hilfe von binären Semaphoren (siehe Abschnitt 2.5). Abbildung 3.8 zeigt den zeitlichen Verlauf der Taskabarbeitung des MPC860 für den Fall der Synchronisation. Der FPGA-Task besitzt die höchste Priorität, gefolgt von dem JPEG-Task und dem HTTP-Task mit der niedrigsten Priorität der drei Tasks.

Zum Zeitpunkt 6,73 Sekunden befinden sich der JPEG-Task und der FPGA-Task im blockierten Zustand (pending). Sie warten jeweils auf ein Semaphor. Hat der HTTP-Server ein Bild gesendet, gibt er das Semaphor 1 frei, die den JPEG-Task blockiert. Der JPEG-Task gibt sofort das Semaphor 2 für den FPGA-Task frei. Da er die größte Priorität hat bekommt der nun unblockierte FPGA-Task die Rechenzeit des Prozessor zugeteilt (executing). Hat der FPGA-Task nach 70,412ms ein Bild aus dem SRAM-Speicher der Adapterplatine ausgelesen, blockiert er sich selbst durch das Anfordern des Semaphors 2. Automatisch bekommt der Task mit der nächst niedrigeren Priorität, die Rechenzeit, in diesem Fall der JPEG-Task. Er komprimiert das Bild in 230,342ms, und setzt sich danach in den Zustand blockiert, durch Anfordern des Semaphors 1. Jetzt bekommt der HTTP-Task die Rechenzeit. Erfolgt oder erfolgte in der Zwischenzeit eine Anfrage durch den Client, kann der HTTP-Server das neue Bild übertragen und der Zyklus beginnt von vorne.

Für einen Auslesezyklus benötigt die Internetkamera in diesem Beispiel 333,919ms, so dass eine maximale Bildwiederholfrequenz von ca. 3 Bildern pro Sekunde erreicht wird.

Das Semaphor 2 besitzt einen Timeout von ca. 1,65s, das heißt, sie wechselt automatisch nach 1,65s in den unblockierten Zustand. Dieser Mechanismus wird dazu verwendet, das Bild im Speicher auch ohne Anfrage des Clients periodisch zu aktualisieren. Ohne diesen Timeout würde der HTTP-Server nach langen Ruhepausen ein veraltetes Bild zum Client senden. Abbildung 3.9 zeigt den zeitlichen Verlauf der Taskabarbeitung für diesen Fall.

3.4.2 Speichermanagement

Die drei Tasks müssen die Bilddaten untereinander weiterreichen. Dazu werden gemeinsame genutzte Speicherbereiche verwendet (siehe Abbildung 3.10). Der Zugriff auf jeden dieser Speicherbereiche ist durch

Abbildung 3.8: Taskarbeitung ohne Timeout

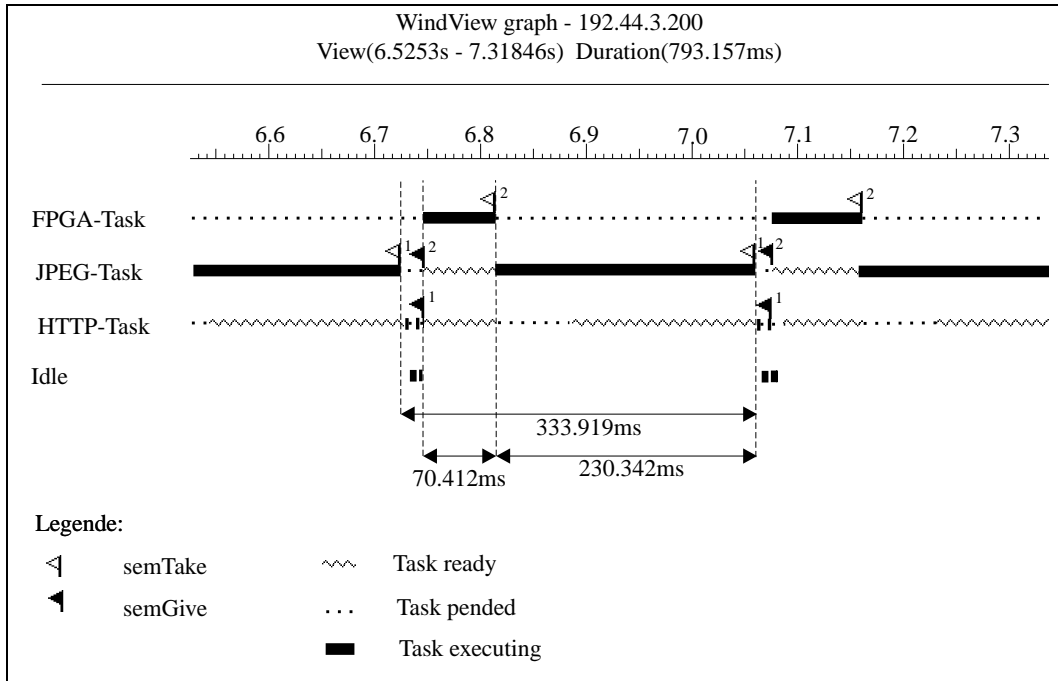
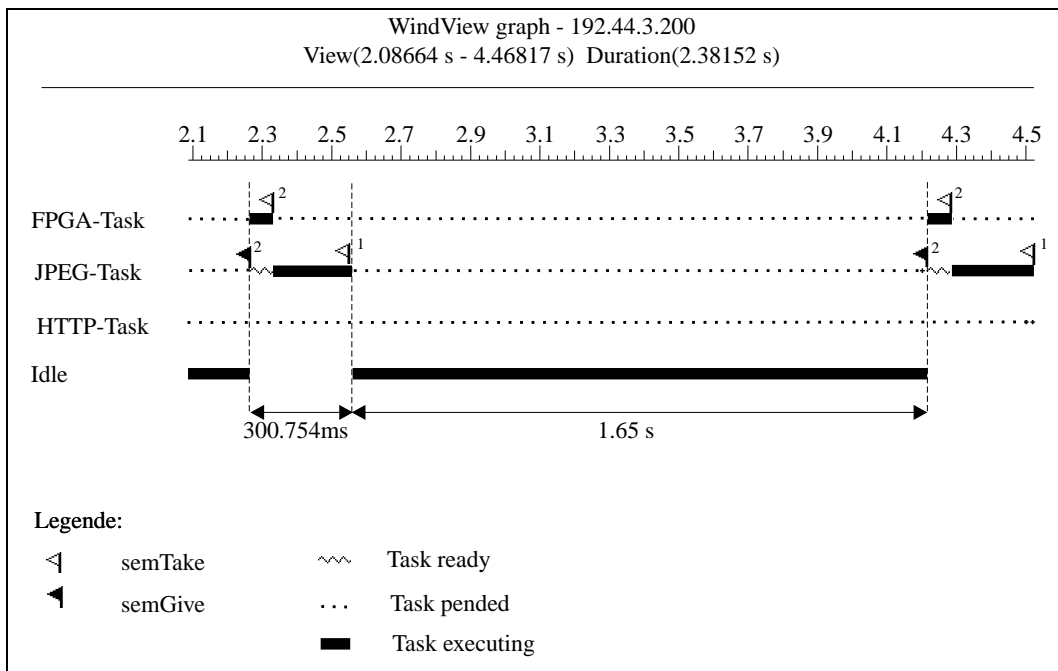
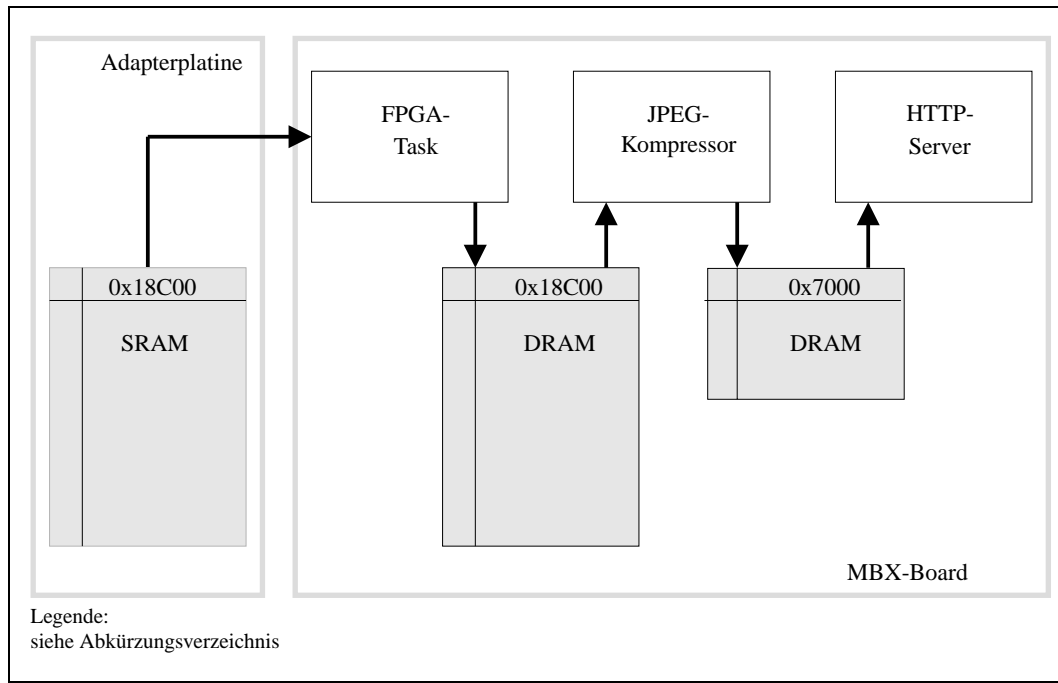


Abbildung 3.9: Taskarbeitung mit Timeout



ein binäres Semaphor geschützt. Vor einem Schreib- oder Lesezugriff muss jede Task das Semaphor anfordern und nach dem Zugriff wieder freigeben. Mit diesem Mechanismus wird die Kollision der Tasks verhindert.

Abbildung 3.10: Speicherorganisation der einzelnen Module



Durch die Synchronisation der Tasks (siehe Abschnitt 3.4.1) ist ein gemeinsamer Zugriff normalerweise ausgeschlossen. Die binären Semaphoren sind somit nur ein zusätzlicher Sicherheitsmechanismus, der besonders wichtig wird, wenn die Tasks nicht mehr im Priority-Scheduling, sondern im Round-Robin-Scheduling abgearbeitet werden.

Ein Task muss bei der Anordnung wie in Abbildung 3.10 möglicherweise auf den anderen warten. Eine Verbesserung ist der Doppelpufferbetrieb wie Abbildung 3.11 gezeigt. Nachdem ein Bild geschrieben worden ist, wird der Speicherbereich umgeschaltet und ein anderer Speicherbereich verwendet. Wartezeiten, die durch unterschiedliche Ausführungszeiten der Tasks bei der Bearbeitung eines Bildes entstehen, können jedoch auch durch diese Anordnung nicht verhindert werden.

Abbildung 3.11: Speicherorganisation der einzelnen Module mit Doppelpufferbetrieb

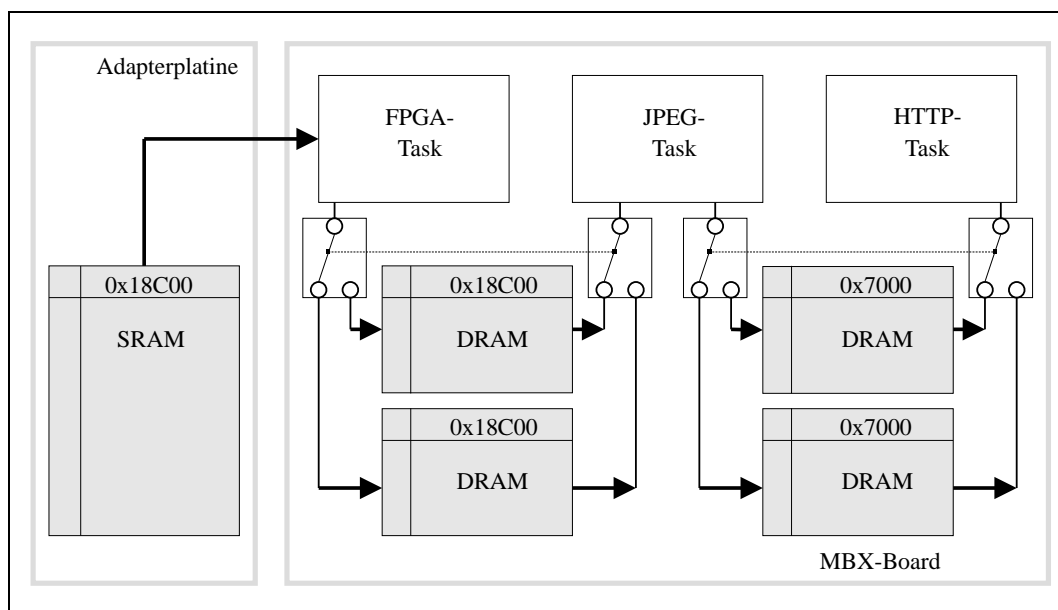
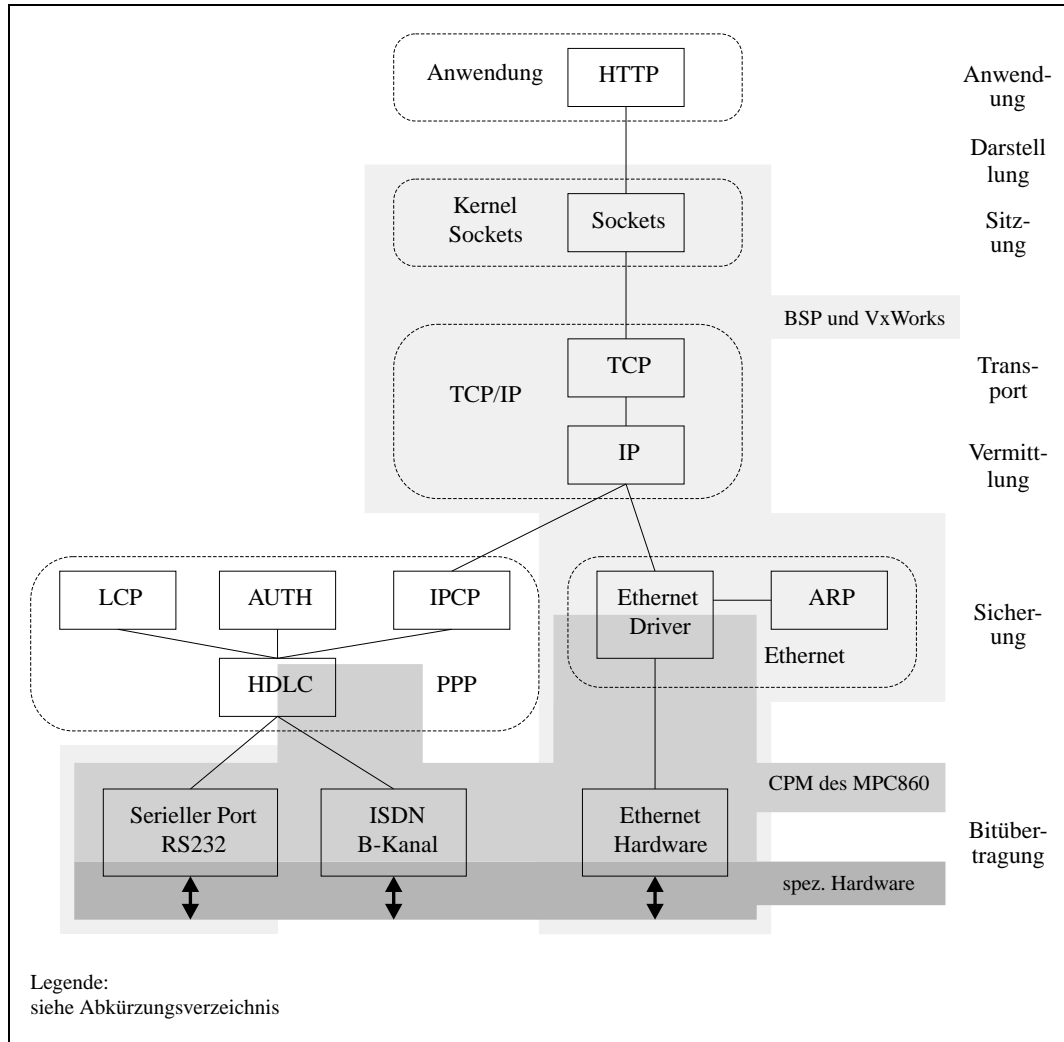


Abbildung 3.12: Aufgabenverteilung bei der Implementierung der Kommunikationsprotokolle



3.5 Kommunikationsprotokolle

Abbildung 3.12 ist eine Erweiterung der Abbildung 2.5 aus Kapitel 2.2. Es werden die verwendeten Protokoll und Schnittstellen zum Aufbau eines Kommunikationssystems für die Internetkamera dargestellt. Durch den Einsatz von VxWorks und das fertig BSP für das MBX-Boards sind die hellgrau unterlegten Protokolle und Schnittstellen bereits in der Standardimplementierung des MBX-Boards verfügbar.

Die mittelgrau unterlegten Bereiche dienen dazu, die Funktionalität des CPM (Communication Processor Module) vom MPC860 in das OSI-Schichtenmodell einzuordnen. Das CPM wird verwendet, um die spezifische Hardware (dunkelgrau gekennzeichnet) der Schnittstellen anzusteuern. Das CPM verfügt über Controller, die verschiedene Protokolle unterstützen. So kann zum Beispiel das bit-synchrone HDLC-Protokoll für den B-Kanal der ISDN-Schnittstelle durch einen Controller des CPM realisiert werden (siehe Abschnitt 3.6).

3.5.1 PPP

VxWorks verfügt über ein PPP-Protokoll, das durch Eintragen in eine Konfigurationsdatei dem Betriebssystem hinzugefügt werden kann.

Dazu wird in die Header-Datei „tornado/target/conf/g/all/conf/gall.h“ die Zeile

```
# define INCLUDE_PPP /* include Point-to-Point Protocol */
```

eingefügt und das Betriebssystem neu erstellt.

Dieses PPP-Protokoll liefert AHDLC-kodierte Daten (vergleiche Abschnitt 2.2.2.5). Daher ist der Einsatz dieses PPP-Protokolls nur dann sinnvoll, wenn die serielle Schnittstelle des MBX-Boards genutzt werden soll. Für die ISDN-Schnittstelle wird ein PPP-Protokoll benötigt, welches das pure PPP-Rahmenformat liefert, damit die Daten durch das CPM bit-synchron HDLC-kodiert werden können.

In der Diplomarbeit von J. Wittmann mit dem Titel „Konzeption und Realisierung einer ISDN-Verbindungssteuerung in Anlehnung an den CAPI 2.0 Standard“ [43], Abschnitt 4.2.5, Seite 80, wurde bereits ein PPP-Protokoll eingesetzt, das sowohl für die serielle Schnittstelle, als auch für die ISDN-Schnittstelle verwendet werden kann. Daher ist es sinnvoll, für die Internetkamera dieses PPP-Protokoll zu verwenden.

Das neue PPP ist auf der CD im Anhang A im Verzeichnis „pppsdl“ zu finden. Das Objektfile „pppd.o“ kann einfach der Anwendung hinzugefügt werden. Das Original-PPP von VxWorks darf in diesem Fall nicht eingebunden sein. Bei der Verwendung der seriellen Schnittstelle verhält sich das neue PPP abwärtskompatibel.

3.5.2 TCP/IP über die Socket-Schnittstelle

Die Socket-Schnittstelle wurde bereits in Abschnitt 2.2.4 vorgestellt. Das folgende C-Programm enthält die Initialisierungssequenz für eine TCP-Socketverbindung auf der Serverseite:

```
#include <vxWorks.h>
#include <sockLib.h>

struct sockaddr_in serverAddr; /* server's socket address */
int sFd; /* socket file descriptor */
int sockAddrSize; /* size of socket address structure */
const int one = 1;
const int keepalive_value = 1;

/* choose local address*/
sockAddrSize = sizeof (struct sockaddr_in);
bzero ((char *) &serverAddr, sockAddrSize);
serverAddr.sin_family = AF_INET;
serverAddr.sin_port = htons (SERVER_PORT_NUM);
serverAddr.sin_addr.s_addr = htonl (INADDR_ANY);

/* create the TCP-socket */
sFd = socket (AF_INET, SOCK_STREAM, IPPROTO_TCP)

/* socket options*/
setsockopt(sFd, SOL_SOCKET, SO_REUSEADDR, (char *)&one, sizeof(one));
setsockopt(sFd, SOL_SOCKET, SO_KEEPALIVE, (char *)&keepalive_value,
```

```

        sizeof(keepalive_value));

/* bind socket to local address*/
bind(sFd, (struct sockaddr *) &serverAddr, sockAddrSize);
listen(sFd, SERVER_MAX_CONNECTIONS);
.
.

```

Nach der Initialisierung kann der Server ein `accept()` ausführen. Erfolgt eine Anfrage durch den Client, kehrt `accept()` zurück und die Verbindung ist aufgebaut. Der Server kann mit `read()`- und `write()`-Operationen Daten mit dem Client austauschen:

```

struct sockaddr_in clientAddr; /* client's socket address */
int newFd; /* socket descriptor from accept */

newFd = accept (sFd, (struct sockaddr *)&clientAddr,
                &sockAddrSize);
nRead = read(newFd, &request_message[0], REQUEST_MSG_SIZE);

```

3.6 ISDN

Um die ISDN-Schnittstelle der Internetkamera zu realisieren, müssen folgende Arbeitspakete bearbeitet werden.

- Die Bitübertragungsschicht der ISDN-Schnittstelle wird mit Hilfe des im MPC860 integrierten CPM und einem Schicht 1 Baustein von Motorola realisiert.
- ISDN-D-Kanal-Protokolle werden implementiert und mit den bestehenden Kommunikationsprotokollen verknüpft.

Abbildung 3.13 zeigt das Konzept für die ISDN-Schnittstelle und die Anbindung der D-Kanal-Protokolle an die Kommunikationsprotokolle.

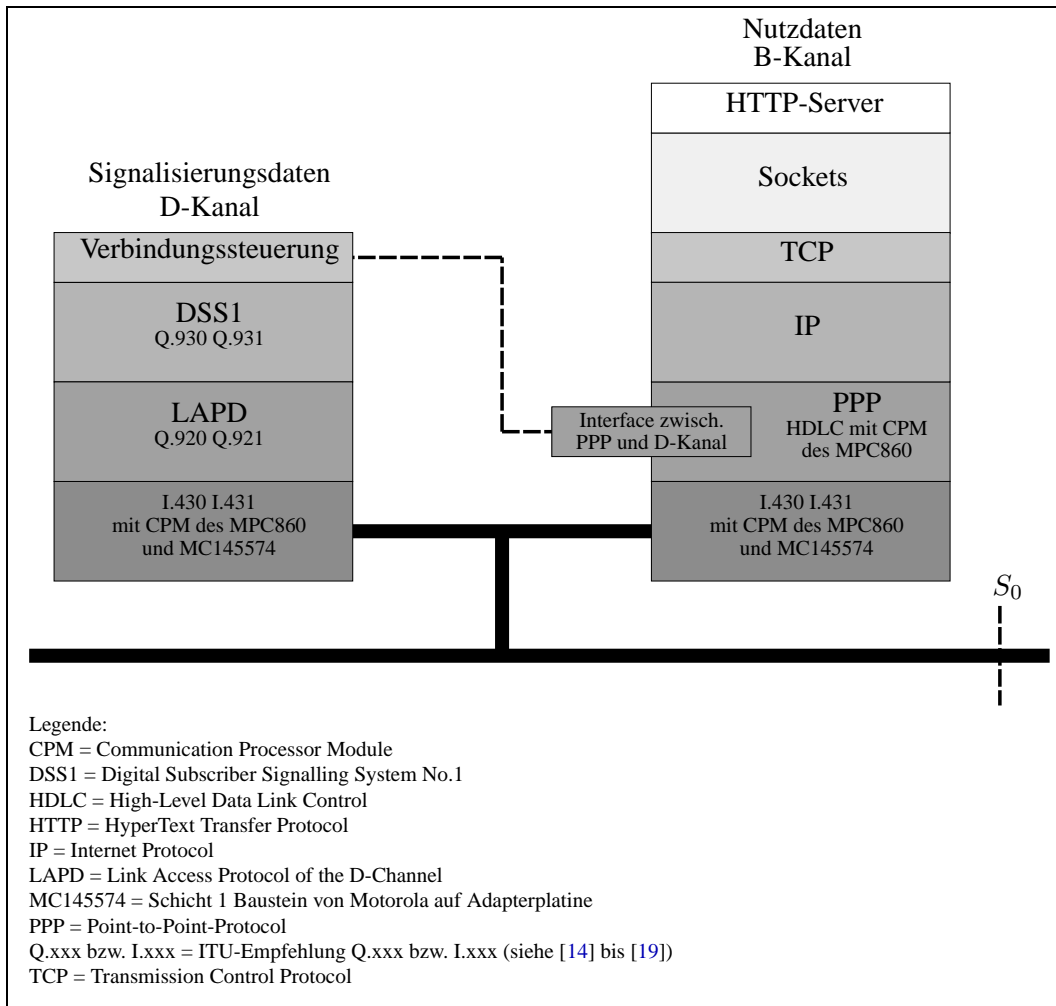
3.6.1 Bitübertragungsschicht

Die Bitübertragungsschicht wird durch das CPM des MPC860 und den MC145574 Motorola Schicht 1 Baustein, der sich auf der Adapterplatine befindet, realisiert. In Abbildung 3.14 werden die beiden Komponenten gezeigt.

Das CPM und der MC145574 kommunizieren über des Serial Peripheral Interface (SPI) und Interchip Digital Link (IDL) miteinander. Das SPI wird verwendet, um den MC14557 durch den MPC860 zu initialisieren und zu steuern. In der Gegenrichtung werden vom MC145574 Nachrichten an den MPC860 gesendet, die ihn z.B. über einen Zustandswechsel auf dem S_0 -Bus informieren. IDL wird verwendet, um die eigentlichen Daten des D-Kanals und der B-Kanäle im Vollduplexmodus zu übertragen. Es wird ebenfalls der Request/Grant-Mechanismus unterstützt, mit dessen Hilfe festgestellt wird, ob D-Kanal-Daten gesendet werden können.

In der Diplomarbeit von J. Pluschke mit dem Titel „Konzeption und Realisierung der hardwarenahen Protokollschichten des Euro-ISDN für ein Hochleistungskommunikationssystem“ bei Prof. Zimmer [30] wurde sich bereits mit diesem Arbeitspaket beschäftigt. Das SPI wurde im Rahmen dieser Arbeit erfolgreich

Abbildung 3.13: D-Kanal- und Kommunikationsprotokolle bei der ISDN-Schnittstelle



implementiert. Mit Hilfe des SPI wurde der MC145574 so initialisiert, dass er eine Schicht 1-Verbindung auf dem S_0 -Bus aufbauen kann.

Im Folgenden wird die Kommunikation über IDL beschrieben:

3.6.1.1 Interchip Digital Link

Die Grundlage zur Verwendung von IDL mit dem MPC860 ist sein Time Slot Assigner (TSA). Durch ihn werden die verschiedenen Kommunikationskanäle (SCCs oder SMCs¹⁵) im Zeitmultiplex auf einen TDM¹⁶-Kanal geführt. Dabei werden IDL und GCI¹⁷, auch bekannt als IOM-2¹⁸, vom MPC860 unterstützt.

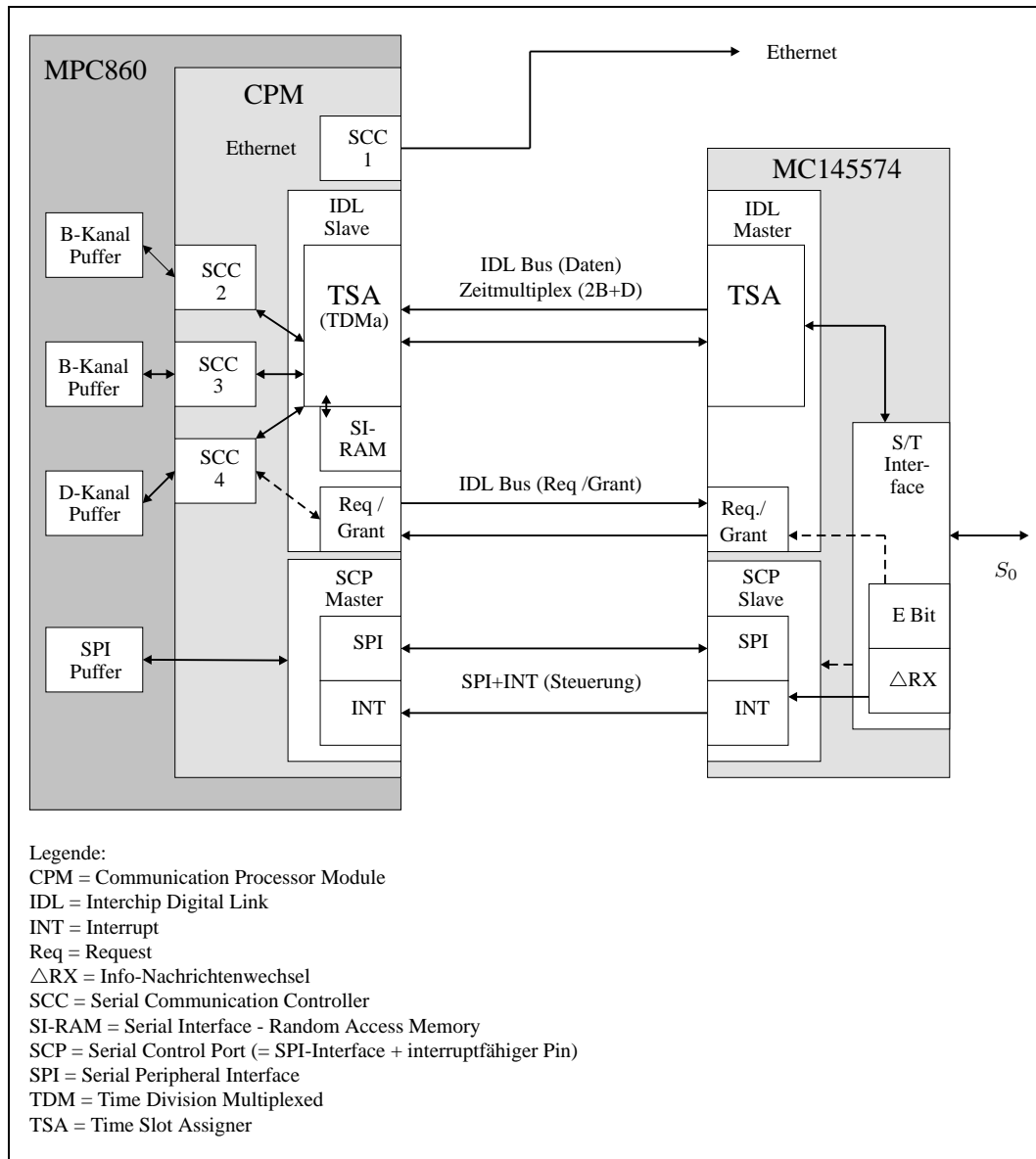
¹⁵SMC = Serial Management Controller

¹⁶TDM = Time-Division Multiplexed

¹⁷GCI = General Circuit Interface

¹⁸IOM-2 = ISDN-Oriented Modular rev 2.2

Abbildung 3.14: Der MC145574 Schicht-1 Baustein und das Communication Processor Module des MPC860

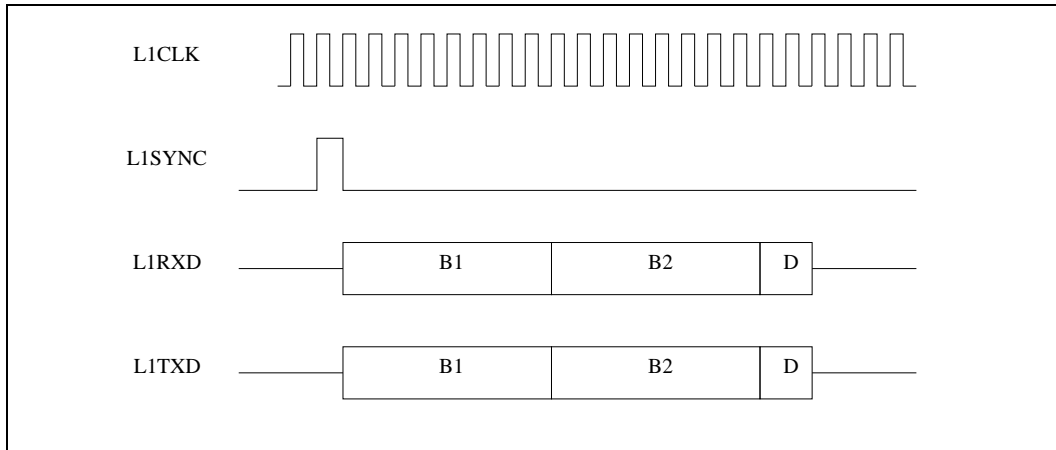


Quelle: [30], Seite 81

Um die Daten per IDL im Zeitmultiplex zum MC14455 zu übertragen, werden folgenden Signale verwendet (Quelle: [24], Abschnitt 16.12.6, Seite 127):

- L1RCLK x - clock signal; input to the MPC860
- L1RSYNCK x - sync signal; input to the MPC860
- L1RXDK x - receive data; input to the MPC860
- L1TXDK x - transmit data; open-drain output
- L1RQ x - request permission to transmit on the D channel; output

Abbildung 3.15: Timing des 8-Bit-IDL



Quelle: [24], Seite 16-129

Tabelle 3.5: SI-RAM Einträge für 8-Bit-IDL

Eintrag Nr.	RAM Word						Beschreibung
	SWTR	SSEL	CSEL	CNT	BYT	LST	
1	0	0000	010	0000	1	0	8 Bit SCC2 für B-Kanal 1
2	0	0000	011	0000	1	0	8 Bit SCC3 für B-Kanal 2
3	0	0000	100	0001	0	1	2 Bit SCC4 für D-Kanal

Anmerkung: Die Tabelle ist nach einem Beispiel des MPC860-Handbuchs [24] angefertigt worden.
Die Abkürzungen werden dort in Abschnitt 16.12.4.5 erklärt

- L1GRx - grant permission; input to the MPC860

In Abbildung 3.15 ist das Timing des IDL-Buses gezeigt.

3.6.1.2 Time Slot Assigner

Das Timing des IDL-Buses wird mit Hilfe des TSA erzeugt. Dem TSA kann über SI-RAM Einträge mitgeteilt werden, welcher SCC zu welchem Zeitpunkt auf den zeitmultiplexten IDL-Kanal geschaltet wird. Die Programmieranleitung für die SI-RAM-Einträge ist im Handbuch des MPC860 [24], im Abschnitt 16.12.4.5 auf Seite 16-106 zu finden. Beispiele sind auf den Seiten 16-109, 16-131 (10 Bit-IDL) und 16-135 (GCI) vorhanden.

Für 8-Bit IDL werden die verwendeten SI-RAM Einträge in Tabelle 3.5 gezeigt. Dabei wird davon ausgegangen, dass SCC2 für den ersten B-Kanal, SCC3 für den zweiten B-Kanal und SCC4 für den D-Kanal verwendet wird (wie in Abbildung 3.14 gezeigt). Da IDL die gleichen SI-RAM Einträge fürs Senden und Empfangen benötigt, wird das SI-RAM wie folgt programmiert:

Definitionen aus BSP (ppc860Siu.h):

```
/* SI Routing RAM */
# define SIRAM(base) (CAST(VUINT32 *) (base + 0x0C00))
```

C-Programm:

```

UINT32 immrVal = vxImmrGet();

*(SIRAM(immrVal)+0) = (0x00820000);
*(SIRAM(immrVal)+1) = (0x00C20000);
*(SIRAM(immrVal)+2) = (0x01050000);
*(SIRAM(immrVal)+64) = (0x00820000);
*(SIRAM(immrVal)+65) = (0x00C20000);
*(SIRAM(immrVal)+66) = (0x01050000);

```

Beim verwendeten „One Multiplexed Channel with Static Frames“-Modus (siehe MPC860-Handbuch, Abschnitt 16.12.4.1) liegen die Einträge für die Empfangsrichtung bei der SI-RAM-Adresse 0 und für die Sende-richtung bei Adresse 64 (Achtung: Dies entspricht einem Offset von 256 Bytes zwischen SI-RAM-Adresse 0 und 64, da ein Eintrag 4 Byte lang ist). SIRAM() ist eine Definition des BSP, die den Offset des SI-RAMs der Adresse des IMM hinzu addiert und somit die Startadresse des SI-RAMs liefert.

3.6.1.3 Serial Communication Controller

Das CPM verfügt über vier SCCs¹⁹. Diese vier SCCs können unabhängig konfiguriert werden, um verschiedene serielle Protokolle zu verwenden. Die SCCs holen sich Daten aus dem Speicher des MPC860, wenden das gewählte Protokoll auf die Daten an und senden den seriellen Bitstrom an externe Pins des MPC860 oder an den TSA²⁰.

Die Sende- und Empfangsdaten werden durch BD²¹-Tabellen organisiert. Ein BD besteht aus Status, Datenlänge und einen Zeiger auf Daten im externen Speicher. BD-Tabellen sind im Dual-Port-RAM abgelegt. Die Startadresse des Dual-Port-RAMs hat einen Offset von 0x2000 zur Startadresse der IMM. In Abbildung 16-5 auf Seite 16-15 des MBX860-Handbuch ist der Speicheraufbau des Dual-Port-RAM gezeigt.

Innerhalb des Dual-Port-RAMs befindet sich bei Startadresse IMM+0x3C00 das Parameter-RAM. Jedem SCC ist ein Speicherbereich im Parameter-RAM zugeordnet. Die Zuordnungstabelle 3.6 zeigt die Aufteilung des Parameter-RAMs. Durch Einträge in das Parameter RAM kann der SCC konfiguriert werden. Die ersten Einträge für jeden SCC sind protokollunspezifisch, so z.B. die Zeiger auf die Rx und Tx-BD-Tabellen. Ab einem Offset von 30 Byte beginnen die protokollspezifischen Einträge (siehe MPC860-Handbuch [24] Seite-164 für protokollunspezifische Einträge, Seite 16-292 für Transparent-Modus des SCC und Seite 16-216 für HDLC-Modus des SCC).

In Abbildung 3.16 ist die Speicherstruktur der SCCs noch einmal anhand einer Grafik beschrieben.

3.6.2 HDLC im D-Kanal mit SCC und TSA

In den Abschnitten 3.6.1.2 und 3.6.1.3 wurden die Grundlagen für TSA und SCC vorgestellt. Nun soll mit diesen beiden Komponenten ein HDLC-Protokoll im D-Kanal der IDL-Schnittstelle implementiert werden.

Das Programmierbeispiel des MPC-860-Handbuch zum HDLC-Modus des SCC ist auf Seite 16-233 zu finden. Leider verwendet dieses Beispiel (wie alle anderen Beispiele) nicht den TSA, sondern leitet den Ausgangsstrom des SCC auf externe Pins. Die folgende Darstellung orientiert sich an der Beschreibungsweise

¹⁹SCC = Serial Communication Controller

²⁰TSA = Time Slot Assigner

²¹BD = Buffer Descriptor

Tabelle 3.6: Zuordnungstabelle des Parameter-RAMs

	Seite	Adresse	Peripherie
IMMR + 0x3C00	1	DPRAM_Base + 0x1C00	SCC1
		DPRAM_Base + 0x1C7F	
		DPRAM_Base + 0x1C80	I2C
		DPRAM_Base + 0x1CAF	
		DPRAM_Base + 0x1CB0	MISC
		DPRAM_Base + 0x1CBF	
		DPRAM_Base + 0x1CC0	IDMA1
IMMR + 0x3D00	2	DPRAM_Base + 0x1D00	SCC2
		DPRAM_Base + 0x1D7F	
		DPRAM_Base + 0x1D80	SPI
		DPRAM_Base + 0x1DAF	
		DPRAM_Base + 0x1DB0	Timers
		DPRAM_Base + 0x1DBF	
		DPRAM_Base + 0x1DC0	IDMA2
IMMR + 0x3E00	3	DPRAM_Base + 0x1E00	SCC3
		DPRAM_Base + 0x1E7F	
		DPRAM_Base + 0x1E80	SMC1
		DPRAM_Base + 0x1EBF	
		DPRAM_Base + 0x1EC0	DSP1
IMMR + 0x3F00	4	DPRAM_Base + 0x1F00	SCC4
		DPRAM_Base + 0x1F7F	
		DPRAM_Base + 0x1F80	SMC2
		DPRAM_Base + 0x1FBF	
		DPRAM_Base + 0x1FC0	DSP2
		DPRAM_Base + 0x1FFF	
Legende: DSP = Digital Signal Processor IDMA = Independent Direct Memory Access I2C = Synchroner Multimaster Bus SCC = Serial Communication Controller SMC = Serial Management Controller SPI = Serial Peripheral Interface Hinweis: IMMR = 0xfa200000 und DPRAM_Base = 0x2000			

Quelle: [24], Seite 16-17

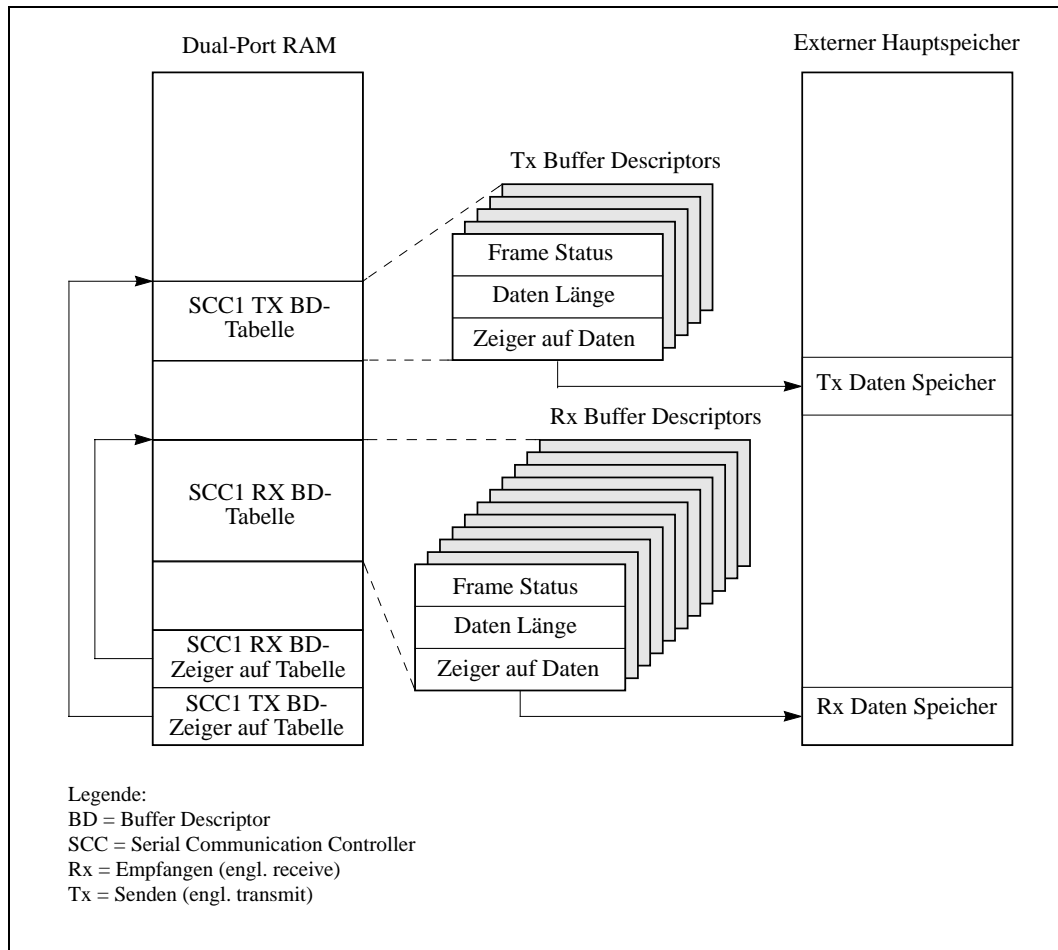
im MPC-860-Handbuch. Abkürzungen und Funktionen der einzelnen Bits und der Register müssen dem MPC860-Handbuch [24] entnommen werden. Die entsprechenden Seitenzahlen sind angegeben.

SCC HDLC Beispiel mit TSA:

Die folgende Initialisierungssequenz verwendet den SCC4. Externes Takt- und Synchronisationssignal werden über die Pins L1RCLKa und L1RSYNCa vom IDL-Master zur Verfügung gestellt. Tabelle 3.7 ordnet den verwendeten Signalnamen Pins auf dem 860/COMM Expansion Connector zu.

1. Programmieren des SI-RAMs. Alle nicht verwendeten Einträge werden mit 0x0001 beschrieben. Die anderen Einträge erfolgen laut Tabelle 3.5.
2. SIMODE = 0x00000141. TDMA wird verwendet. 1-Bit-Delay bei Rx und Tx, TDMA verwendet externe Clock und Sync-Signale. (vgl. MPC860-Handbuch S. 16-114).

Abbildung 3.16: Speicherstruktur der SCCs



Quelle: [24], Seite 16-162

Tabelle 3.7: IDL-Signale auf dem 860/COMM Expansion Connector

Signalname	860/COMM Pin
L1TCLKa	Row B Pin 11
L1RXDa	Row B Pin 13
L1TXDa	Row B Pin 14
L1RSYNCa	Row B Pin 21 oder Row A Pin 17

- SICR Bit1 = 1. SCC4 wird mit dem TSA verbunden. (vgl. MPC860-Handbuch S. 16-121)
- PAPAR Bits 9, 8 und 7 = 1. PADIR Bits 9 und 8 = 1, Bit 7 = 0. Konfiguriert Port A Pins L1TXDa, L1RXDa und L1RCLKa. PAODR Bit 9 = 0. Konfiguriert L1TXDa als Open-Drain-Ausgang. (vgl. MPC860-Handbuch S.16-455, MBX Series Programmer's Guide [22] Seite 3-6 für die Funktionen der Port A Pins, MBX Series Installation and Use [21] Seite 6-19 für Zuordnung der Signal auf dem 860/COMM Expansion Connector).

5. PCPAR Bits 12, 5 und 4 = 1. PADIR Bits 12, 5, 4 = 0. Konfiguriert Port C Pins L1RQa, L1TSYNCa und L1RSYNCa. (vgl. MPC860-Handbuch S.16-465, MBX Series Programmer's Guide [22] Seite 3-9 für die Funktionen der Port C Pins, MBX Series Installation and Use [21] Seite 6-19 für Zuordnung der Signal auf dem 860/COMM Expansion Connector).
6. SIGMR = 0x04. Ermöglichen des TDMA im „One Multiplexed Channel with Static Frames“-Modus (vgl. MPC860-Handbuch S. 16-113).
7. Programmieren des SCC4 Parameter-RAMs (liegt bei IMM + 0x3F00 = 0xfa203F00):
 - Schreibe RXBASE und TXBASE, die auf die BD-Tabellen zeigen. Angenommen die Tx-BD-Tabellen liegen bei 0xfa202A40 und die Rx-BD-Tabellen bei 0xfa202A80, schreibe TXBASE = 0x0A40 und RXBASE = 0x0A80 (vgl. MPC860-Handbuch Seite 16-164).
 - RFCR und TFCR = 0x18 für Big-Endian Bytereihenfolge. (vgl. MPC860-Handbuch Seite 16-165)
 - Schreibe MRBLR mit der maximalen Anzahl von Bytes pro Empfangspuffer.
 - C_MASK = 0x0000F0B8 und C_PRES = 0x0000FFFF, um beim HDLC-Protokoll 16-Bit-CRC zu verwenden (vgl. MPC860-Handbuch Seite 16-216).
 - Lösche DISFC, CRCEC, ABTSC, NMARC und RETRC.
 - Beschreibe MFLR mit der maximalen HDLC-Rahmenlänge.
 - HMASK = 0x0000, um alle Adressen zu erkennen.
 - Lösche HADDR1, HADDR2, HADDR3, HADDR4.
8. Ausführen eines INIT RX PARAMS und INIT TX PARAMS der CPCR für den SCC4. Durch diese Befehle aktualisiert der SCC4 die Werte RBPTR und TBPTR mit den Werten von RXBASE und TXBASE (vgl. MPC860-Handbuch 16-11).
9. Initialisieren eines Rx-BDs. Rx_BD_Status = 0xB000 (E-Bit = 1 für leerer Speicher, W-Bit = 1 für letzter BD und I-Bit = 1 für Interrupt aktiviert (vgl. MPC860-Handbuch 16-226). In der Interruptroutine wird das E-Bit kontrolliert. Wenn E-Bit = 0 wurde ein HDLC-Rahmen empfangen. E-Bit = 1 setzen, um neuen HDLC-Rahmen zu empfangen.) Rx_BD_Length = 0x0000. Rx_BD_Pointer wird mit der Startadresse des Empfangspuffers im externen Speicher beschrieben.
10. Initialisieren eines Tx-BDs. Tx_BD_Status = 0xBC00 (R-Bit = 1 für fertiger Speicher, W-Bit = 1 für letzter BD, und I-Bit = 1 für Interrupt aktiviert, TC-Bit für 16-Bit-CRC (vgl. MPC860-Handbuch 16-228). In der Implementierungsphase ist es sehr nützlich, ebenfalls das CM-Bit zu setzen, um eine kontinuierlichen Sendestrom zu erzeugen.) Tx_BD_Length = Länge des Sendepuffers. Tx_BD_Pointer wird mit der Startadresse des Sendepuffers im externen Speicher beschrieben.
11. SCCE4 = 0xFFFF, um vorherige Ereignisse zu löschen.
12. SCCM4 = 0x009F, um bei diesen Ereignissen Interrupts auszulösen. (vgl. MPC860-Handbuch S. 16-231)
13. CIMR Bit 4 = 1, damit SCC4 einen System-Interrupt auslösen kann (vgl. MPC860-Handbuch S. 16-483).
14. GSMR_H4 = GSMR_L4 = 0x00000000 (vgl. MPC860-Handbuch S. 148). Achtung: Beim Transparent-Modus des SCC GSMR_H4 = 0x00003D80.
15. PSMR = 0x0000 für 16-Bit-CRC und normale HDLC-Operation (vgl. MPC860-Handbuch 16-222).

16. `GSMR_L4 = 0x00000030`, um das Senden und Empfangen mit dem SCC4 zu starten.

Das entsprechende C-Programm ist in den Dateien „MpcTsaHdlc.c“ und „MpcTsaHdlc.h“ zu finden, die auf der CD im Anhang A im Verzeichnis „mbx860isdn“ abgelegt worden sind.

3.6.3 D-Kanal Protokolle

Im Folgenden werden die verwendeten ISDN-D-Kanal-Protokolle aufgeführt, die bereits aus Abbildung 3.13 bekannt sind.

- LAPD und DSS1.
Die Protokolle LAPD²² und DSS1²³ sind im FhG IMS als C-Bibliotheken vorhanden. Für die Internetkamera wird die Version 5.04 des DSS1 Protokolls und Version 4.7 des LAPD Protokolls der Firma GiK (Gesellschaft für innovative Kommunikationssysteme mbH) verwendet.
- Verbindungssteuerung.
Die Verbindungssteuerung liegt in Schicht 4 des Schichtenmodells oberhalb des DSS1-Protokolls. Die eingesetzte Verbindungssteuerung besitzt derzeit nur eine minimale Zustandsmaschine, die nicht alle Nachrichten des DSS1-Protokolls auswertet. Die Nachrichten zum Auf- und Abbau einer ISDN-Verbindung werden jedoch komplett abgearbeitet, so dass mit Hilfe der implementierten Verbindungssteuerung eine ISDN-Verbindung hergestellt werden kann.
- Schnittstelle zwischen PPP und D-Kanal.
Das Interface zwischen PPP und D-Kanal ist im IMS FhG bereits öfter verwendet worden und ist als C-Bibliothek vorhanden. Nach dem Aufbau einer ISDN-Verbindung wird in der Verbindungssteuerung durch das DSS1-Protokoll der Auf- oder Abbau eines B-Kanals signalisiert. Mit Hilfe des Interface zwischen PPP und D-Kanal wird in diesem Fall das PPP-Protokoll gestartet bzw. beendet.

3.6.4 Anbindung der D-Kanal Protokolle und PPP an die Bitübertragungsschicht

Die ISDN-Bitübertragungsschicht der Internetkamera wird an die bestehenden Protokolle der höheren Schichten (LAPD und PPP) angebunden. In Abbildung 3.13 wird die Schnittstelle zwischen LAPD bzw. PPP und der Bitübertragungsschicht dargestellt.

Die horizontale Kommunikation zwischen den Schichten wird beim LAPD-Protokoll von der Firma GiK durch Funktionsaufrufe in der dienstbringenden Schicht und Callback-Funktionen in der eigenen, dienstnutzenden Schicht realisiert. Dabei werden die SDUs der dienstbringenden Schicht bzw. PDUs der dienstnutzenden Schicht durch sogenannte IDUs²⁴ (dynamisch angelegte gemeinsam genutzte Speicherbereiche) beim Funktionsaufruf als Parameter übergeben.

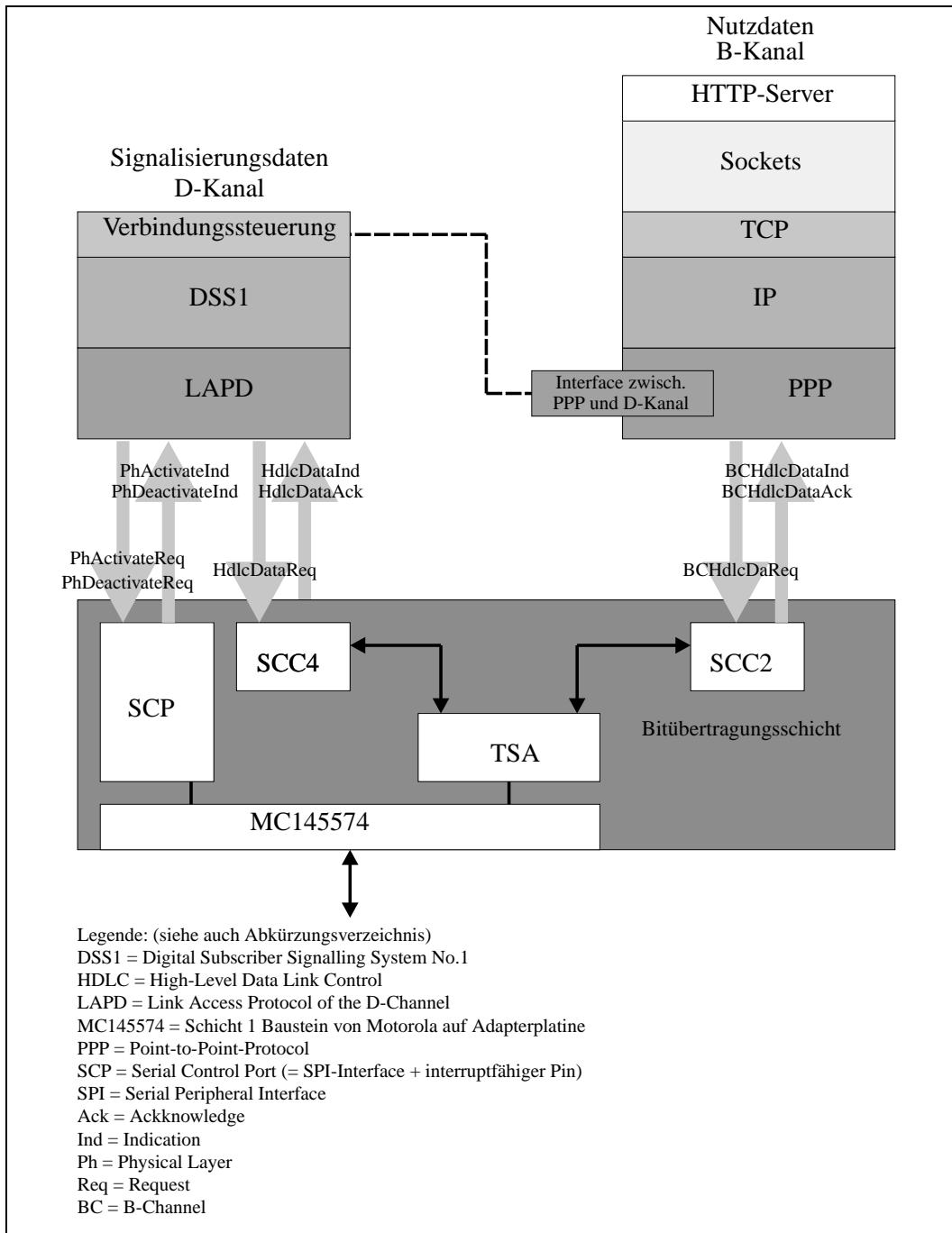
In Abbildung 3.18 ist die Schnittstelle zwischen LAPD und der Bitübertragungsschicht noch einmal verfeinert dargestellt. In den Kreisen sind die Tasks der beiden Schichten und die Interruptroutine zu sehen, die mit SCC4 verknüpft ist. Wenn das LAPD-Protokoll Daten zu versenden hat, ruft der LAPD Scheduler Task die Funktion `HdlcDataReq()` auf und übergibt als Parameter die zu versendende IDU. Die Funktion packt die Daten der IDU in eine Message Queue und sendet sie an den Transmit Task. Dieser schreibt die Daten in

²²LAPD = Link Access Protocol of the D-Channel

²³DSS1 = Digital Subscriber Signalling System No.1

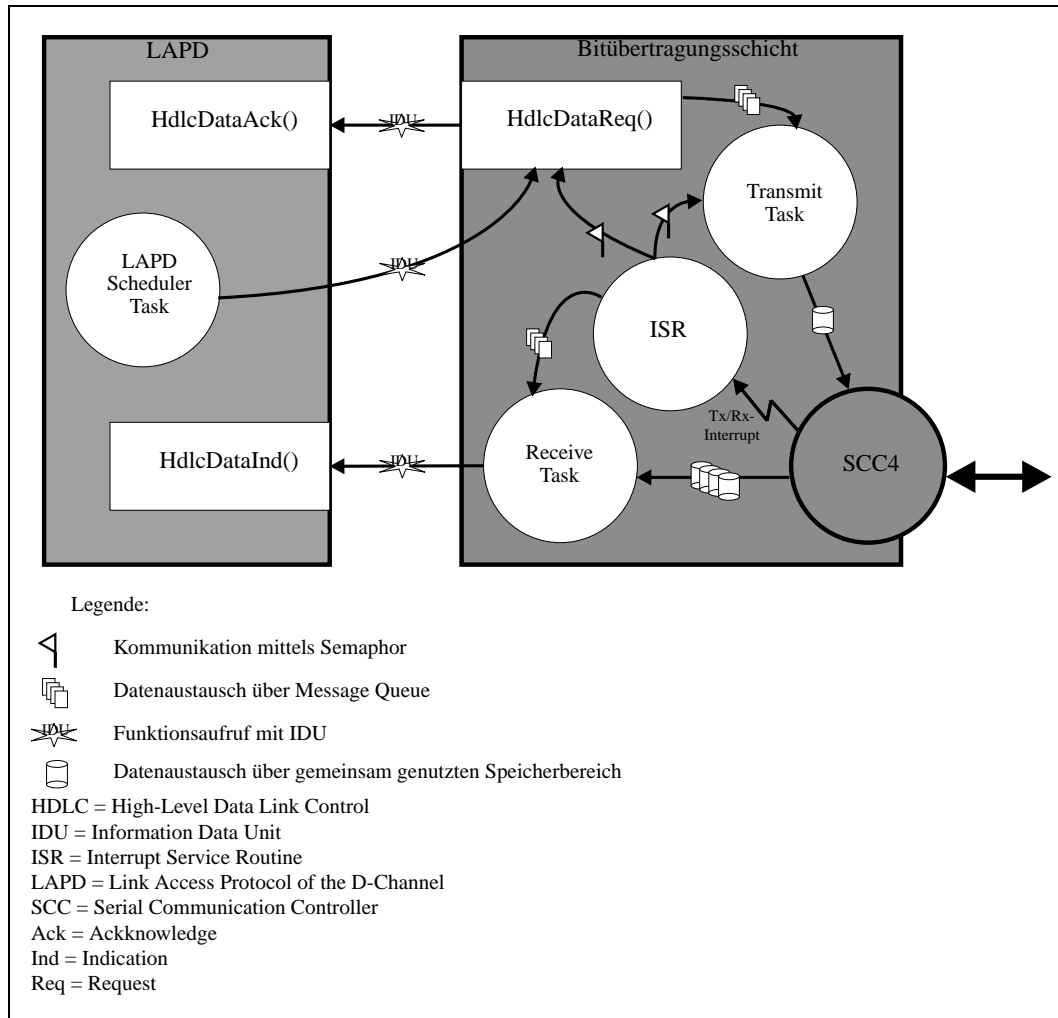
²⁴IDU = Information Data Unit

Abbildung 3.17: Schnittstellenfunktionen zwischen LAPD bzw. PPP und Bitübertragungsschicht



einen lokalen Speicher und gibt dem SCC4 die Adresse und die Länge bekannt. Hat der SCC4 die Daten erfolgreich HDLC-kodiert und versendet (siehe Abschnitt 3.6.2) oder ist eine Fehler aufgetreten, wird ein Interrupt ausgelöst und die ISR ausgeführt. Ein erfolgreiches Versenden der Daten teilt die ISR dem Transmit Task und der Funktion HdlcDataReq() durch freigeben einer Semaphor mit. In diesem Fall ruft die Funktion HdlcDataReq() die LAPD-Callback-Funktion HdlcDataAck() auf und übergibt als Parameter die versendete

Abbildung 3.18: Aufbau der Schnittstelle zur ISDN-Datenübertragung zwischen LAPD und Bitübertragungsschicht



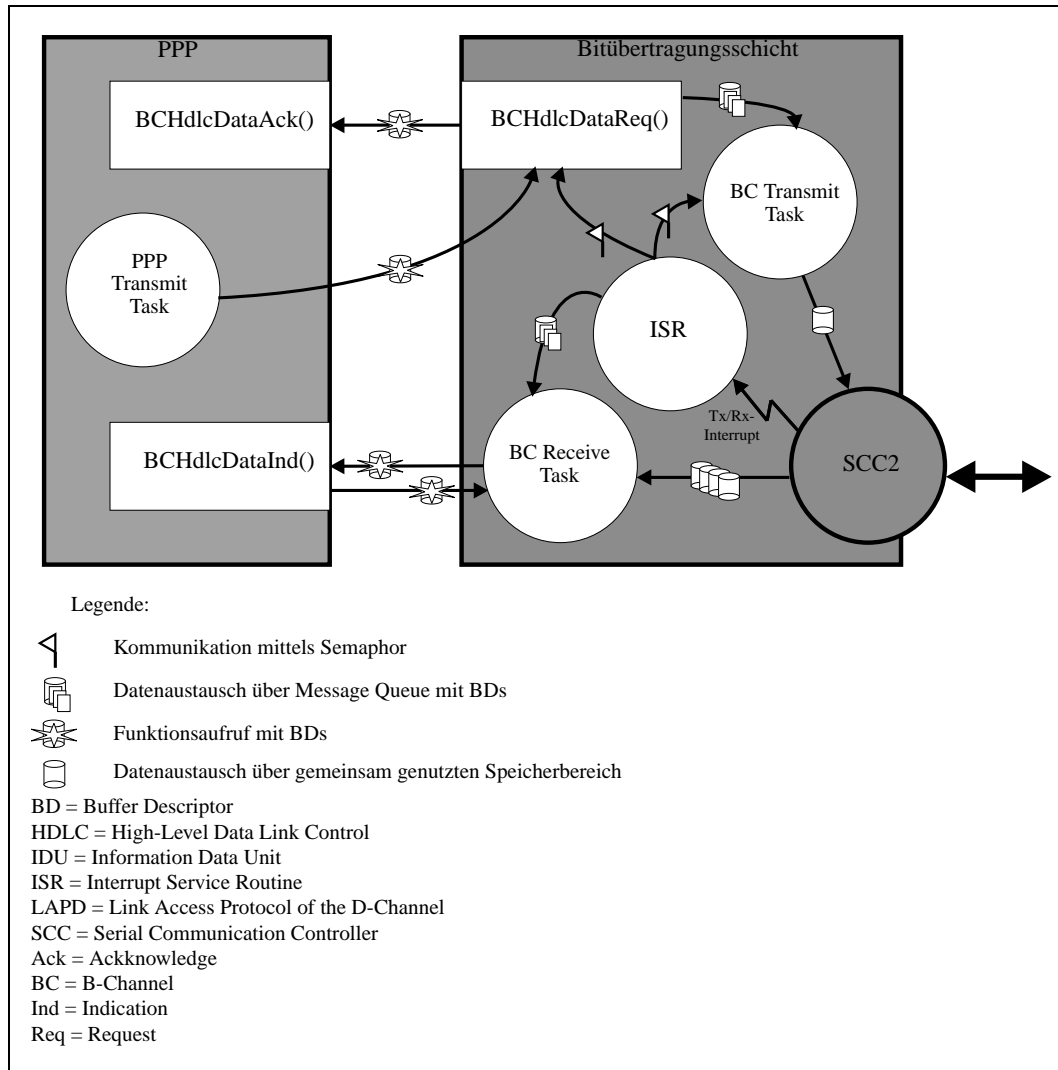
IDU.

Empfängt der SCC4 Daten, signalisiert er dies durch einen Rx-Interrupt. Die ISR wird ausgeführt und versendet die empfangenen Daten über eine Message Queue an den Receive Task. Der Receive Task fordert eine IDU an und beschreibt sie mit den Daten aus der Message Queue. Danach ruft der Receive Task die Callback-Funktion HdlcDataInd() auf und übergibt die IDU als Parameter an die LAPD.

Durch die Verwendung von Message Queues in Send- und Empfangsrichtung werden die Daten notfalls durch die Bitübertragungsschicht zwischengespeichert. Dies kann besonderes beim Empfangen von schnell hintereinander eintreffenden Datenpaketen sinnvoll sein, die durch das System nicht so schnell weiterverarbeitet werden können.

Die Schnittstellenfunktionen zwischen PPP und der Bitübertragungsschicht, werden genauso verwendet wie die Schnittstellenfunktionen zwischen der LAPD und der Bitübertragungsschicht (siehe Abbildung 3.19).

Abbildung 3.19: Aufbau der Schnittstelle zur ISDN-Datenübertragung zwischen PPP und Bitübertragungsschicht



Es werden jedoch keine IDUs sondern Buffer Descriptoren (BD) als Funktionsparameter übergeben, die aus Speicheradresse und Datenlänge bestehen. Die Funktion `BCHdlcDataInd()` besitzt im Unterschied zur Funktion `HdlcDataInd()` der LAPD einen Rückgabewert. Wird von der Bitübertragungsschicht durch die Funktion `BCHdlcDataInd()` der Empfang einer neuen Datenpaketes an das PPP-Protokoll gemeldet, übergibt die Funktion als Rückgabewerte einen neuen BD als Empfangspuffer für den Receive Task.

Das C-Programm der Bitübertragungsschicht ist in den Dateien „MpcTsaHdlc.c“ und „MpcTsaHdlc.h“ zu finden, die auf der CD im Anhang A im Verzeichnis „mbx860isdn“ abgelegt worden sind. Das C-Programm der SPI-Schnittstelle besteht aus den Dateien „mc145574v2.c“ und „mc145574v2.h“.

3.6.5 Bewertung der ISDN-Schnittstelle der Internetkamera

Die ISDN-Schnittstelle der Internetkamera kann in ihrem derzeitigen Implementierungsgrad zur Datenübertragung in einem internen ISDN-Netz verwendet werden. Soll die ISDN-Schnittstelle in einem öffentlichen ISDN-Netz betrieben werden, verbleiben folgende Entwicklungsarbeiten:

- Unterstützen des Request/Grant-Mechanismus.
Der Request/Grant-Mechanismus des D-Kanals wurde im Rahmen dieser Diplomarbeit nicht bearbeitet. Der MC145574-Baustein wurde bei der Initialisierung über SCP²⁵ durch Setzen des Bit5 im Byte Register BR7 (siehe [23], Seite 6-11) konfiguriert, die entsprechenden D-Kanal Prozeduren zu ignorieren.
- Wechseln der B-Kanäle.
Derzeit kann die Kamera nur im ersten B-Kanal Daten versenden. Wenn dieser durch ein anderes Teilnehmerendgerät belegt ist, werden die Daten normalerweise über den zweiten B-Kanal gesendet. Die entwickelte Software der Kamera unterstützt den Wechsel des B-Kanals jedoch noch nicht.
- Lösen der Cache-Problematik.
Beim Arbeiten mit dem MPC860 unter VxWorks traten häufig Inkonsistenzen bei der Verwendung von dynamisch angelegtem Speicher auf. Besonders bei I/O-Operationen machte sich dieses Problem stark bemerkbar. Die beobachteten Inkonsistenzen verschwanden beim Abschalten des Caches vollständig. Die Systemleistung des MBX-Boards wurde durch das Abschalten des Caches jedoch stark beeinträchtigt. Deshalb wurde versucht, bei eingeschaltetem Cache Betriebssystemfunktionen zu verwenden, die den Inhalt des Caches und des Hauptspeichers abgleichen sollten. Dies führte jedoch zu keinem befriedigendem Ergebnis. Die ISDN-Schnittstelle der Internetkamera wird daher derzeit mit abgeschaltetem Cache betrieben.

Um das Cache auszuschalten wird in die Header-Datei „tornado/target/confg/all/confgall.h“ die Zeile

```
#undef USER_D_CACHE_ENABLE    /* undef to leave disabled*/
```

eingefügt und das Betriebssystem neu erstellt.

- Erweiterung der Verbindungssteuerung
Die Zustandsmaschine der Verbindungssteuerung arbeitet derzeit nicht alle Nachrichten der DSS1-Schicht ab. Besonders wichtig wäre die Auswertung der Rufnummer des gerufenen Teilnehmers. Derzeit beantwortet die Internetkamera jeden Anruf auf dem S_0 -Bus.
- Zulassung der Hard- und Software zum Betrieb im öffentlichen Netz.
Vor dem Betrieb in einem öffentlichen Netz müssen die Schichten 1 bis 3 der ISDN-Schnittstelle vom Zentralamt für Zulassungen im Fernmeldewesen (ZZF) auf Einhaltung der Normen geprüft werden.

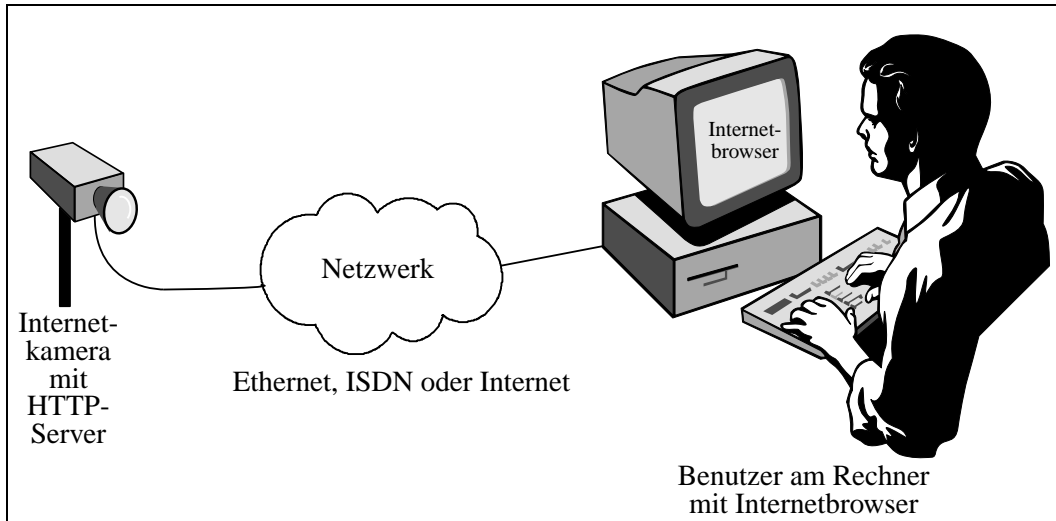
Bei den anstehenden Entwicklungsarbeiten sind aus systemtechnischer Sicht keine Schwierigkeiten zu erwarten. Die Internetkamera kann daher in Zukunft als Teilnehmerendgerät im ISDN-Netz betrieben werden.

²⁵Serial Control Port

Kapitel 4

Verifikation des Systems

Abbildung 4.1: Visualisierung des Kamerabildes in einem Internetbrowser



4.1 Visualisierung in einem Internetbrowser

In diesem Kapitel wird die Funktionalität der Internetkamera überprüft. Das Kamerabild wird dazu in einem Internetbrowser visualisiert und periodisch aktualisiert. Dazu wird zwischen Internetkamera und Internetbrowser eine TCP/IP-Verbindung hergestellt. Die TCP/IP-Verbindung kann über Ethernet, Internet, ISDN oder serielle Schnittstelle aufgebaut werden (siehe Abbildung 4.1).

4.1.1 HTML

HTML¹-Dateien können mit Hilfe des HTTP-Servers der Internetkamera über TCP/IP in den Internetbrowser des PC-Benutzers geladen werden. Die Sprache HTML beschreibt, wie eine Seite in einem Internetbrowser dargestellt werden soll. Dazu werden in HTML-Dokumenten Formatierbefehle verwendet, die der Browser interpretiert. Die einfachste HTML-Seite, besteht aus den drei Formatierbefehlen HTML, HEAD und BODY. Das folgende HTML-Dokument erzeugt beim Aufruf „MBX-Board WebCam (1.1) - example“ in der Kopfzeile des Browser und die große Überschrift „MBX-Board WebCam“ auf der Internetseite selbst:

```
<HTML>
  <HEAD>
    <TITLE>MBX-Board WebCam (1.1) - example</TITLE>
  </HEAD>
  <BODY>
    <H1> MBX-Board WebCam </H1>
  </BODY>
</HTML>
```

Um das Bild der Internetkamera auf der Seite darzustellen, kann der Formatierbefehle IMG verwendet werden. Dem Parameter SRC wird die URL des Bildes mit dem Namen „aus.jpg“ übergeben:

```
<HTML>
  <HEAD>
```

¹HTML = HyperText Markup Language

```
<TITLE>MBX-Board WebCam (1.2) - example</TITLE>
</HEAD>
<BODY>
  <H1> MBX-Board WebCam </H1>
  <IMG SRC="http://192.44.3.200/aus.jpg">
</BODY>
</HTML>
```

Dieses HTML-Dokument erzeugt jedoch nur ein einzelnes stehendes Bild. Um das Kamerabild periodisch zu aktualisieren, können drei verschiedenen Methoden verwendet werden.

- META-Tag
- JavaScript
- Java-Applet

Die drei Möglichkeiten werden im Folgenden vorgestellt:

4.1.2 META-Tag

Der META-Tag wird in HTML-Dokumenten zur Angabe von Zusatzinformationen verwendet. Durch den Parameter HTTP-EQUIV wird die Art der Information beschrieben. Durch den Parameter CONTENT wird die Information übergeben. Diese Informationen können durch jeden Server oder Client interpretiert werden, dem die Semantik der Information bekannt ist. Häufig wird der META-Tag im Internet verwendet, um Inhaltsangaben oder Stichwörter der HTML-Seite für Server mit Suchmaschinen bereitzustellen. Die Interpretation eines META-Tags ist nicht standardisiert und kann von Fall zu Fall variieren. Daher kann die Verwendung zu unbestimmten Ergebnissen führen.

Das folgende HTML-Dokument verwendet einen META-Tag mit dem Parameter HTTP-EQUIV="refresh" und CONTENT=1.0. Durch dieses META-Tag wird die Internetseite alle 1,5 Sekunden durch den Internetbrowser aktualisiert. Dazu wird eine neue Anfrage an den HTTP-Server gerichtet und die neu übertragene Seite dargestellt.

```
<HTML>
  <HEAD>
    <TITLE>MBX-Board WebCam (1.3) - example</TITLE>
    <META HTTP-EQUIV="refresh" CONTENT=1.5>
  </HEAD>
  <BODY>
    <H1> MBX-Board WebCam </H1>
    <IMG SRC="http://192.44.3.200/aus.jpg">
  </BODY>
</HTML>
```

Beim Aufruf dieser Seite mit dem Microsoft Internet Explorer wird alle 1,5 Sekunden ein neues Bild der Internetkamera dargestellt. Wird dieselbe Seite mit dem Netscape Navigator geöffnet, wird die Seite zwar alle 1,5 Sekunden neu aufgebaut, es erscheint jedoch kein aktualisiertes Bild der Kamera. Der Netscape Navigator stellt keine neue Anfrage an den HTTP-Server, sondern holt sich HTML-Seite und Bild aus dem internen Speicher, so dass jedesmal das gleiche Bild dargestellt wird.

Soll der Netscape Navigator bei Verwendung des META-Tags periodisch aktualisierte Bilder der Internetkamera liefern, kann dies durch einen kleinen Zusatzaufwand im HTML-Dokument erreicht werden. Durch die

Verwendung von mehreren Frames auf der Internetseite werden die Bilder auch durch den Netscape Navigator aktualisiert.

```
<HTML>
  <HEAD>
    <TITLE>MBX-Board WebCam (1.4) - example</TITLE>
    <META HTTP-EQUIV="refresh" CONTENT=0.01>
  </HEAD>
  <BODY>
    <H1> MBX-Board WebCam </H1>
    <FRAMESET ROWS='67%,33%' BORDER=1>
      <FRAME SRC="http://192.44.3.200/aus.jpg">
      <FRAME SRC="http://192.44.3.200/example4.html">
    <NOFRAMES>
  </NOFRAMES>
  </FRAMESET>
</BODY>
</HTML>
```

Die Verwendung des META-Tags ist zwar die einfachste Methode, aber aufgrund der unterschiedlichen Ergebnisse bei den verschiedenen Browsern ist der Einsatz bei der Internetkamera nicht sinnvoll. Ein weiterer Nachteil ist die zitternde und unruhige Darstellung. Die gesamte Seite wird jedes Mal im Browser gelöscht und dann relativ langsam neu aufgebaut.

4.1.3 JavaScript

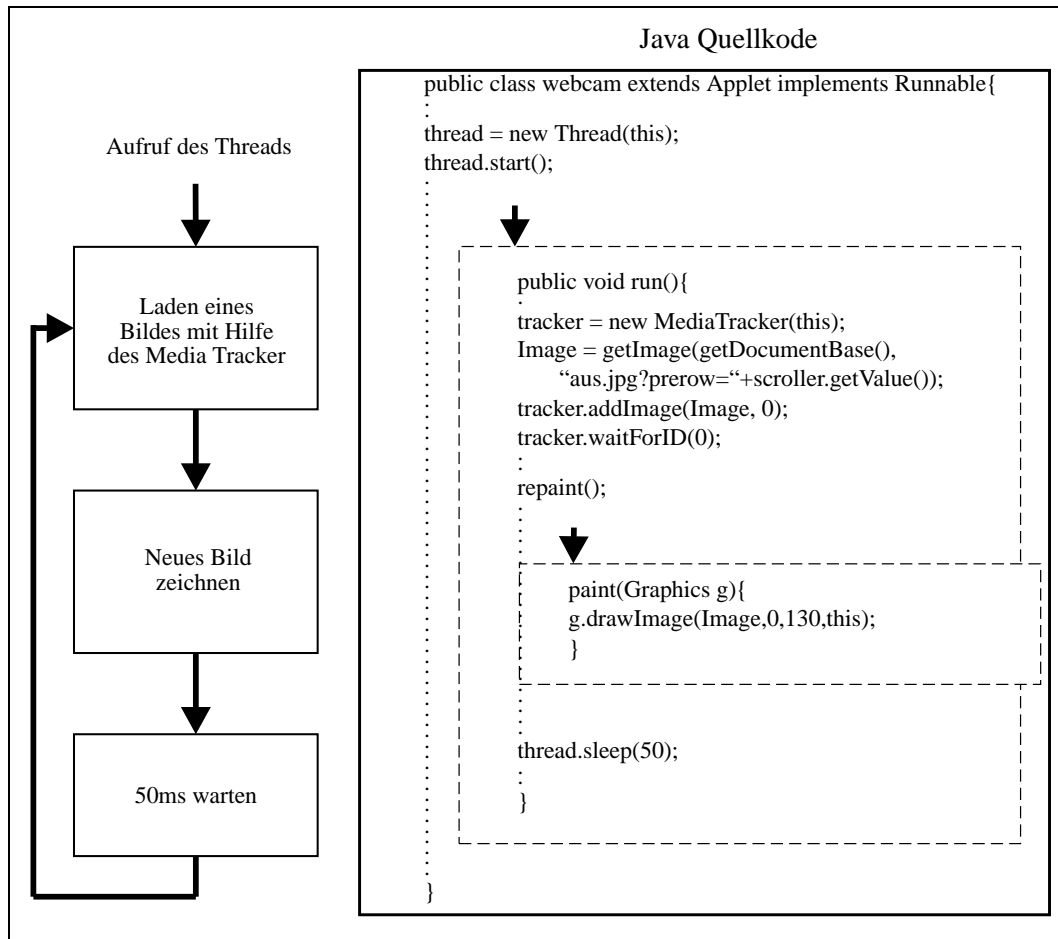
JavaScript erweitert HTML um Objekte und Funktionen mit dem Ziel, HTML-Seiten flexibler zu gestalten. Um JavaScript zu verwenden, muss ein JavaScript-fähiger Browser verwendet werden. Durch den Formatierbefehl `<SCRIPT language="JavaScript"> ... </SCRIPT>` wird der JavaScript-Kode direkt und unkompiliert dem HTML-Dokument hinzugefügt.

Das folgende Beispiel zeigt das erweiterten HTML-Dokument zum periodischen Anzeigen des Kamerabildes mit Hilfe von JavaScript:

```
<HTML>
  <HEAD>
    <TITLE>MBX-Board WebCam (1.5) - example</title>
    <SCRIPT language="JavaScript">
      i = 1;
      function next()
      {document.animation.src = "aus.jpg?Zaehler="+i++;}
    </SCRIPT>
  </HEAD>
  <BODY>
    <H1> MBX-Board WebCam </H1>
    <IMG SRC="aus.jpg" NAME= "animation"
      ONLOAD = "setTimeout('next()',100)">
  </BODY>
</HTML>
```

Der Formatierbefehl IMG ist um die Parameter NAME und ONLOAD erweitert worden. Durch ONLOAD wird der Funktionsaufruf der JavaScript-Funktion setTimeout() übergeben. Diese Funktion wird ausgeführt, sobald das Bild vom Browser geladen worden ist. Die Funktion setTimeout() wartet 100 ms und ruft die JavaScript-Funktion next() auf. next() ändert den Parameter SRC des Formatierbefehls IMG. Daraufhin liest

Abbildung 4.2: Periodisches Aktualisieren des Kamerabildes mit Hilfe eines Java-Applets



der Browser ein neues Bild der Internetkamera.

Bei jedem Lesezyklus wird die Datei „aus.jpg“ vom Server der Internetkamera geladen. Würde next() den Parameter SRC bei jedem Zyklus gleich „aus.jpg“ setzen, würde der Browser das Bild nicht aktualisieren, da der Dateiname sich nicht geändert hat. Um diese Problematik zu umgehen, wird der Dateiname um die Zeichenkette „?Zaehler=“ und einen variablen Wert erweitert, der sich bei jedem Zyklus ändert. Da dem Server die Variable Zaehler nicht bekannt ist, ignoriert er die Zeichen nach dem Fragezeichen (vgl. GET-Methode beim CGI, siehe Abschnitt 3.4). Er sendet daher bei jedem Zyklus die aktualisierte Datei „aus.jpg“ an den Browser.

Die Verwendung von JavaScript zur Darstellung des Kamerabildes ist eine einfache Methode, die sehr gute Ergebnisse liefert. Die Darstellung der Seite ist im Gegensatz zur Verwendung des META-Tags ruhig und stummerfrei. Der Browser aktualisiert nur das Bild und nicht die gesamte Seite. Zudem wird das alte Bild nicht vom Bildschirm gelöscht, sondern lediglich vom neuen überschrieben.

4.1.4 Java-Applet

Java-Applets sind in der Programmiersprache Java geschrieben. Der kompilierte Code in Form von Klassen-Dateien (*.class) kann in HTML-Seiten eingebunden werden. Die Klassen-Dateien werden mit der HTML-Datei auf den Rechner des Benutzers geladen und ausgeführt. Damit läuft ein Programm auf dem Rechner des Benutzer ab, das unter Berücksichtigung bestimmter Sicherheitsmechanismen den vollen Funktionsumfang der Programmiersprache Java nutzen kann.

Durch den Formatierbefehl APPLET wird das Java-Applet in die HTML-Seite eingefügt.

```
<HTML>
  <HEAD>
    <TITLE>MBX-Board WebCam (1.6) - example</TITLE>
  </HEAD>
  <BODY>
    <H1> MBX-Board WebCam </H1>
    <APPLET code=webcam.class name=webcam width=650 height=730>
    </APPLET>
  </BODY>
</HTML>
```

Um das Kamerabild im Java-Applet darzustellen und periodisch zu aktualisieren, wird in der Applet-Klasse ein Thread (Task) gestartet, der diese Aufgabe übernimmt. In Abbildung 4.2 ist ein Ablaufdiagramm des Threads gezeigt. Neben den einzelnen Funktionsblöcken steht der entsprechende Java-Quellcode.

Der komplette Quellcode des Java-Applets ist in der Datei „webcam.java“ im Verzeichnis „JavaApplet“ auf der CD im Anhang A zu finden.

Bei der Darstellung des Kamerabildes liefert der Einsatz eines Java-Applet genauso gute Ergebnisse wie die Verwendung von JavaScript. Der Programmieraufwand beim Java-Applet war jedoch um ein Vielfaches größer. Die Bildwiederholfrequenz konnte bei der Verwendung des Applets gegenüber JavaScript gesteigert werden.

In Abbildung 4.3 wird das Java-Applet gezeigt. Durch die beiden Druckknöpfe START und STOP kann das Ausführen des Threads gestartet oder gestoppt werden. Unter den Druckknöpfen wird eine Information des Applet ausgegeben. Sie besteht aus der Anzahl der bisher empfangenen Bilder, der aktuellen und der durchschnittlichen Frame-Rate. Im Feld SIZE kann die Größe des Kamerabildes gewählt werden (quarter, half, original, double). Mit dem Schieberegler PREROW kann die Helligkeit des Kamerabildes eingestellt werden. Dieser Wert wird der Steuerung des CMOS-Sensors durch das in Abschnitt 3.2.1 und 3.4 dargestellte Verfahren übermittelt. Auf diese Weise kann der Benutzer des Applets die Internetkamera fernsteuern.

4.2 Inbetriebnahme der Internetkamera und Aufbau der Verbindung

4.2.1 Starten der Kameraanwendung

Die Internetkamera befindet sich noch in der Entwicklungsphase. Daher ist sie nicht direkt beim Einschalten betriebsbereit. Um die Internetkamera über Ethernet oder Internet zu verwenden, muss die Kameraanwendung bestehend aus FPGA-, JPEG- und FPGA-Task auf dem System vorhanden und gestartet sein. Dazu wird die Objektdatei „webcam.o“, die ich auf der CD im Anhang A im Verzeichnis „webcam“ befindet, auf die Internetkamera heruntergeladen. Dies geschieht mit Hilfe der Entwicklungsumgebung Tornado der Firma

Abbildung 4.3: Java-Applet der Internetkamera



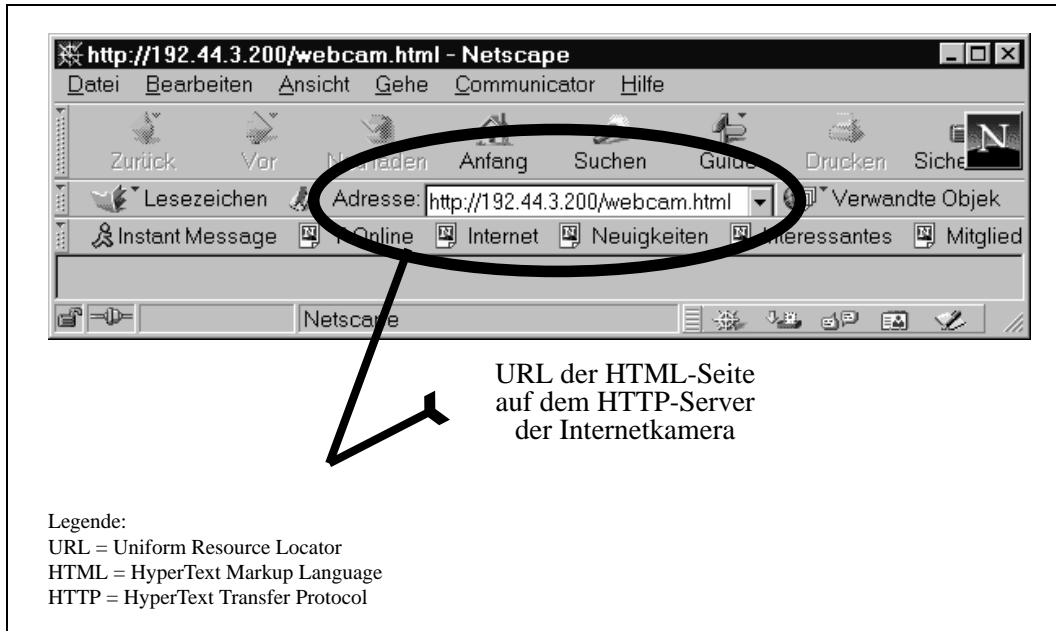
Wind River. Durch den Funktionsaufruf `start_webcam()` wird die Anwendung gestartet.

Nach Abschluss der Entwicklungsarbeit kann ein selbststartendes Archiv, das alle Module der Internetkamera enthält, erstellt werden. Dies Archiv wird im Konstantenspeicher des MBX-Boards abgelegt, um ein echtes autonomes System zu erhalten.

4.2.2 Internet oder Ethernet

Bei der TCP/IP-Verbindung ist die Internetkamera der Server und der Internetbrowser auf dem PC der Client. In Abbildung 4.4 wird gezeigt, wie der Benutzer durch Eingabe der entsprechenden URL im Browserfenster die

Abbildung 4.4: Aufbau einer TCP/IP Verbindung mit Internetbrowser als Client



HTML-Seite über Internet oder Ethernet vom HTTP-Server der Internetkamera anfordern kann. Ist die Datei dem HTTP-Server bekannt, sendet er sie über TCP/IP an den Browser.

4.2.3 Serielle Schnittstelle

Die Bilder der Internetkamera können über die seriellen RS232-Schnittstelle abgefragt werden. Dazu muss eine PPP-Verbindung über TCP/IP zwischen Internetkamera und einem PC aufgebaut werden.

Um PPP auf der Internetkamera zu verwenden, muss die Objektdatei „pppd.o“ aus dem Verzeichnis „pppsdl“ auf das System heruntergeladen werden. Das PPP wird durch Aufruf der Funktion serial_up() auf der Internetkamera gestartet.

In Abbildung 4.5 wird dargestellt, wie die serielle Verbindung über PPP auf PC-Seite unter dem Betriebssystem Window NT 4.0 hergestellt wird. Dazu muss ein „DFÜ²-Netzwerk mit seriellen Kabel zwischen 2 PCs“, wie dargestellt, eingerichtet werden.

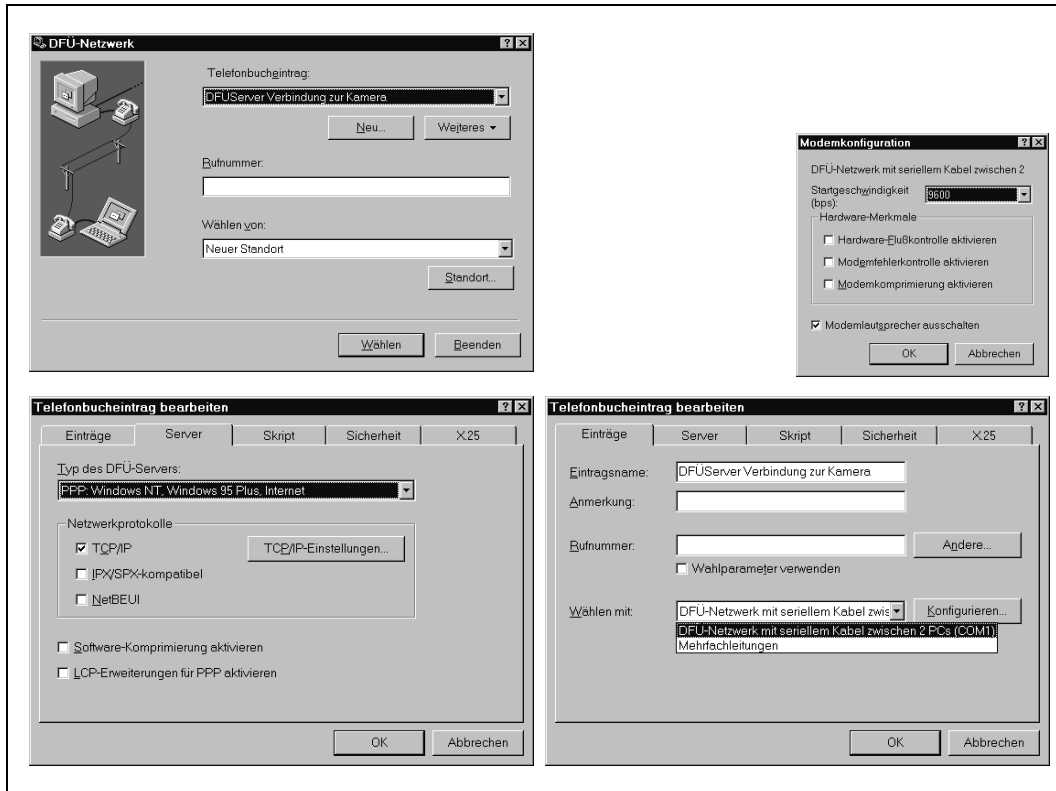
Abbildung 4.6 zeigt die Statusmeldung des DFÜ-Monitors nach erfolgreichem Verbindungsaufbau. Nach dem Aufbau der Verbindung, können die Bilder der Kamera, wie bei einer Internetverbindung (siehe Abbildung 4.4), mit Hilfe eines Browser angeschaut werden.

4.2.4 ISDN

Der Aufbau einer ISDN-Verbindung zur Internetkamera wird genau wie bei der seriellen Verbindung vorgenommen. Bei einem PC mit installierter ISDN-Karte unter dem Betriebssystem Windows NT 4.0 wird ein

²DFÜ = Daten Fern Übertragung

Abbildung 4.5: Aufbau einer seriellen Verbindung über PPP und TCP/IP unter Windows NT 4.0



DFÜ-Netzwerk, wie in Abbildung 4.7 gezeigt, eingerichtet.

Um die ISDN-Schnittstelle und PPP auf der Internetkamera zu starten, muss die Objektdatei „mpc860isdn.o“ aus dem Verzeichnis „mpc860isdndundpp“ auf das System heruntergeladen werden. Das PPP-Protokoll und die ISDN-Schnittstelle werden durch Aufruf der Funktion `isdn_up()` auf der Internetkamera gestartet.

In der Datei „isdn_up.txt“ im Verzeichnis „ISDN Aurora logfiles“ auf der CD im Anhang A ist des Protokoll des Auf- und Abbau einer ISDN-Verbindung vom PC zur Internetkamera zu finden. Die Nachrichten der Schicht 1 bis Schicht 3 des D-Kanal-Protokolls werden gezeigt. Durch diese Protokolldatei wird belegt, dass der Auf- und Abbau der ISDN-Verbindung zur Internetkamera einwandfrei funktioniert.

Abbildung 4.6: Statusreport der seriellen Verbindung über PPP und TCP/IP unter Windows NT 4.0

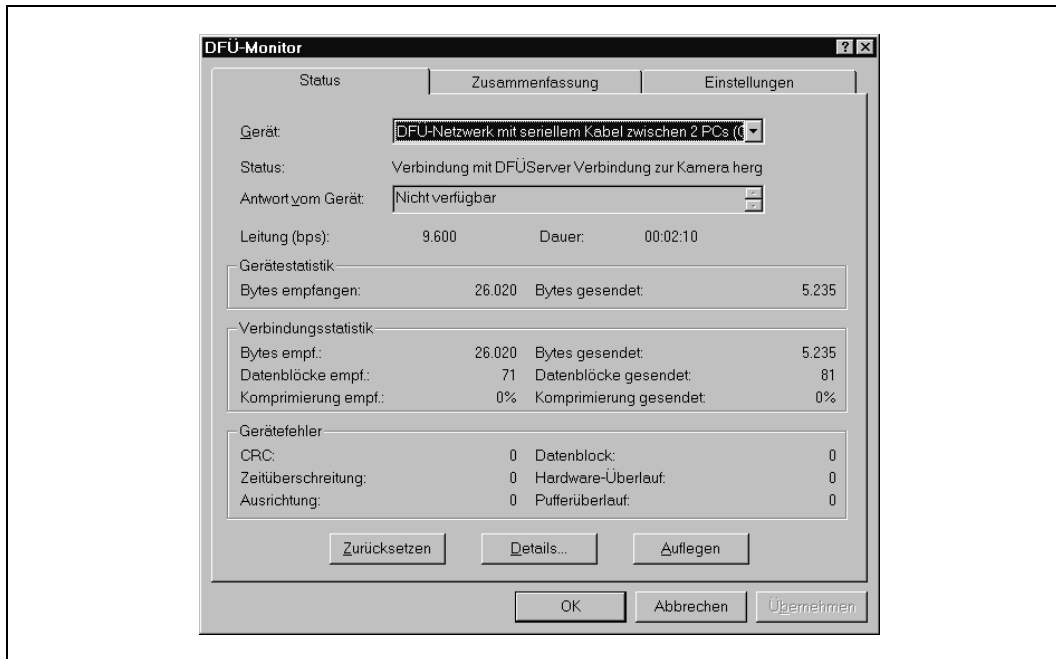
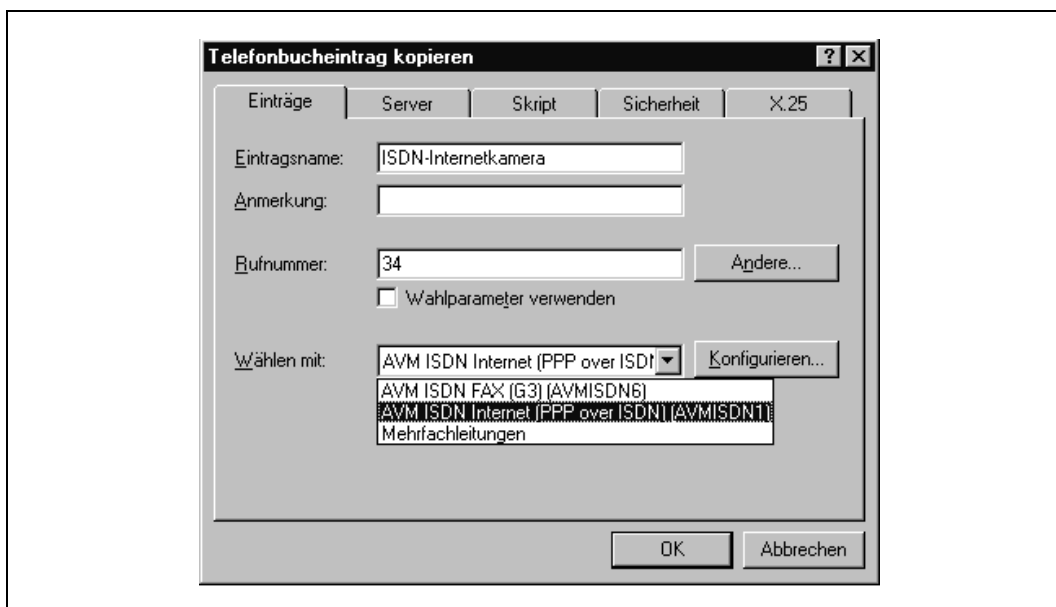


Abbildung 4.7: Aufbau einer ISDN-Verbindung zur Internetkamera über PPP und TCP/IP unter Windows NT 4.0 mit Hilfe einer ISDN-Karte



Kapitel 5

Zusammenfassung und Ausblick

Im Rahmen dieser Diplomarbeit wurden zuerst die theoretischen Grundlagen zur Echtzeitprogrammierung, JPEG, ISDN, Kommunikationsprotokollen und offenen Systemen beschrieben, um sich auf diese Weise in das Thema einzuarbeiten und zum Kern der Arbeit hinzuführen. Es wurde ein Konzept zur Realisierung eines Gesamtsystems „ISDN-Internetkamera“ aufgestellt und die Arbeiten in Arbeitspakete aufgeteilt.

Im ersten Arbeitspaket wurden die Bilddaten aus dem SRAM-Speicher auf der Adapterplatine in den Hauptspeicher des Prozessorboards übertragen. Im nächsten Schritt wurden sie durch einen JPEG-Kompressor komprimiert. Anschließend wurde dem System ein Internetserver hinzugefügt. Danach war es möglich, die Bilder der Kamera mit Hilfe der Ethernet-Schnittstelle über einen Browser abzurufen. Um die Bilder über die ISDN-Schnittstelle zu übertragen, wurde das Kommunikationsprotokoll PPP auf der Internetkamera implementiert. Da die ISDN-Schnittstelle noch nicht zur Verfügung stand, wurde das PPP-Protokoll zunächst dazu verwendet, Bilder über die serielle RS232-Schnittstelle zu versenden. Im nächsten Arbeitspaket entstand die ISDN-Schnittstelle. Die ISDN-Hardware wurde mit Hilfe des im MPC860 integrierten Kommunikationsprozessormoduls angesprochen und durch ein ISDN-Testgerät überprüft. Im letzten Arbeitspaket wurden die ISDN-D-Kanal-Protokoll implementiert und eine TCP/IP-Verbindung über PPP zu einem PC über ISDN aufgebaut.

Zum Betreiben der Internetkamera per ISDN-Schnittstelle in einem öffentlichen Netz, bedarf es noch weiterer Entwicklungsarbeiten. Da die ISDN-Schnittstelle erst kürzlich fertig gestellt wurde, war es bisher nicht möglich, sie ausgiebig zu testen. Bei der Verwendung der Ethernet-Schnittstelle der Internetkamera ist eine stabile und zuverlässige Funktion der Hard- und Software der Internetkamera festzustellen.

Die Internetkamera auf Basis des MPC860 ist ein eingebettetes System mit enormem Entwicklungspotential. Diverse Anwendungen aus dem Bereich der Automatisierungs- und Kommunikationstechnik können auf dem entwickelten System aufsetzen.

Mögliche Anwendungsbereiche sind die Überwachung von bestimmten Ereignissen oder Zuständen, wie z.B. die Anwesenheit von Personen, und die Überwachung von Prozessen und Abläufen, wie z.B. bei einer Verkehrszählung oder in einem Produktionsprozess. Durch die ISDN-Schnittstelle ist es möglich, auch weit entfernt aufgestellte Internetkameras per Wählverbindung in ein TCP/IP-Netzwerk zu integrieren. Diese Kameras könnten bei bestimmten Ereignissen selbständig eine Wählverbindung aufbauen, um Nachrichten oder Alarmmeldungen weiterzuleiten.

Im IMS werden ständig neue CMOS-Bildsensoren entwickelt. Durch die Verwendung eines programmierbaren FPGAs auf der Adapterplatine kann die Internetkamera mit beliebigen Sensoren verwendet werden. Zum Test der Sensoren und zur Weiterverarbeitung der Bilddaten mit Bildverarbeitungsalgorithmen ist das vorgestellte leistungsstarke autonome System ideal geeignet.

Der JPEG-Kompressor kann in zukünftigen Anwendungen durch Video-Kodierer nach dem MPEG- oder H.261-Standard ersetzt werden, um eine bessere Ausnutzung der Bandbreite des Übertragungskanal zu erhalten. Mit Hilfe von H.261 ist es möglich, eine ISDN-Schnittstelle nach dem H.320-Standard aufzubauen und auf diese Weise mit ISDN-Bildtelefonen oder Videokonferenzanlagen zu kommunizieren.

Literaturverzeichnis

- [1] A. Badach, K. Merz, S. Müller, *ISDN und CAPI. Grundlagen der Programmierung von ISDN-Anwendungen auf dem PC*. VDE-Verlag
- [2] M. F. Barnsley, L.H. Hurd. *Bildkompression mit Fraktalen*. 1993. Vieweg Verlag GmbH.
- [3] W. Brecht, *Verteilte Systeme unter UNIX* 1992. Vieweg Verlag GmbH.
- [4] J. Carlson, *PPP Design and Debugging*. 1998. Addison Wesley Longman.
- [5] CCITT/ITU T.81, *Digital Compression and Coding of Continuous-Tone Still Images - Requirements and Guidelines*. 1992. -.
- [6] CCITT/ITU T.83, *Digital Compression and Coding of Continuous-Tone Still Images - Compliance Testing*. 1994. -.
- [7] C. Facchi, *Methodik zur formalen Spezifikation des ISO/OSI Schichtenmodells*. 1995. Herbert Utz Verlag.
- [8] P. Gerdson, P. Kröger. *Kommunikationssystem 1*. 1994. Springer-Verlag.
- [9] J. Hildebrandt, *Konzeption und Realisierung einer ISDN-Datenübertragungsschnittstelle gemäß dem E-DSS1-Standard*. Jan 1998. Diplomarbeit bei Prof. Zimmer. IMS FhG.
- [10] Andy C. Hung, *PVRG-JPEG CODEC 1.1*. Internet: [ftp:// havefun.stanford.edu:pub/jpeg/JPEGv1.1.tar.Z](ftp://havefun.stanford.edu:pub/jpeg/JPEGv1.1.tar.Z). am 1.3.99
- [11] R. Klute, *Das World Wide Web. Web-Server und -Clients, HTML 2.0/3.0, HTTP*. 1996. Addison-Wesley GmbH.
- [12] G. McGregor, *rfc1332 The PPP Internet Protocol Control Protocol (IPCP)*. 1992. Internet: <http://www.cis.ohio-state.edu/htbin/rfc/rfc1332.html> am 30.8.99.
- [13] R. Niles, J.Dwight, *CGI by Example*. 1996. Que Coporation
- [14] o.V. *Digital Subscriber Signalling System No.1 (DSS1) User-Network Interface Data Link Layer - General Aspects*. 1993. ITU-T Recommendation Q.920
- [15] o.V. *Digital Subscriber Signalling System No.1 (DSS1) Abstract Test Suite for LAPD Conformance Testing*. 1993. ITU-T Recommendation Q.921
- [16] o.V. *Digital Subscriber Signalling System No.1 (DSS1) User-Network Interface Layer 3 - General Aspects*. 1993. ITU-T Recommendation Q.930

- [17] o.V. *Digital Subscriber Signalling System No.1 (DSS1) User-Network Interface Layer 3 - Specification for Basic Call Control*. 1993. ITU-T Recommendation Q.931
- [18] o.V., *ISDN User-Network Interfaces, Basic User-Network Interface - Layer 1 Specification*. 1995. ITU-T Recommendation I.430
- [19] o.V., *ISDN User-Network Interfaces, Primary Rate User-Network Interface - Layer 1 Specification*. 1993. ITU-T Recommendation I.431
- [20] o.V., *MBX Probe Card Installation Instructions*. 1998. Motorola Inc.
- [21] o.V., *MBX Series Embedded Controller Installation and Use*. 1997. Motorola Inc.
- [22] o.V., *MBX Series Embedded Controller Programmer's Reference Guide*. 1998. Motorola Inc.
- [23] o.V., *MC145574 ISDN S/T-Interface Transceiver*. 1997. Motorola Inc.
- [24] o.V., *MPC860 PowerQUICC User's Manual*. 1996. Motorola Inc.
- [25] o.V., *Protokolle und Dienste der Informationstechnologie*. 1998. Interest Verlag.
- [26] o.V., *VxWorks Programmer's Guide 5.3*. 1995. Wind River Systems Inc.
- [27] o.V., *VxWorks Network Programmer's Guide 5.4 Edition 1*. 1999. Wind River Systems Inc.
- [28] o.V., *X.200 - Data Networks and Open Systems Communication*. 1994. ITU-T-Standard.
- [29] J.-R. Ohm, *Digitale Bildcodierung*. 1995. Springer Verlag.
- [30] J. Pluschke, *Konzeption und Realisierung der hardwarenahen Protokollschichten des Euro-ISDN für ein Hochleistungskommunikationssystem*. Oktober 1999. Diplomarbeit bei Prof. Zimmer. IMS FhG.
- [31] J. K. Reynolds, *Assigned Numbers*. 1992. Internet: <http://www.cis.ohio-state.edu/htbin/rfc/rfc1662.html> am 30.8.99
- [32] M. Santifaller, *TCP/IP und ONC/NFS in Theorie und Praxis*. 1993. Addison-Wesley GmbH.
- [33] R. Schoblick, *Euro-ISDN im praktischen Einsatz*. 1996. Franzis-Verlag GmbH.
- [34] W. Simpson, *rfc1661 The Point-to-Point Protocol (PPP)*. 1994. Internet: <http://www.cis.ohio-state.edu/htbin/rfc/rfc1661.html> am 30.8.99.
- [35] W. Simpson, *rfc1662 PPP in HDLC-like Framing*. 1994. Internet: <http://www.cis.ohio-state.edu/htbin/rfc/rfc1662.html> am 30.8.99
- [36] T. Stevens, *Nutzung von Intra- und Internetmechanismen für Eingebettete Systeme*. Januar 1998
Diplomarbeit bei Prof. Zimmer. IMS FhG.
- [37] W. Stevens, *TCP/IP Illustrated Volume 1*. 1993. Addison-Wesley GmbH.
- [38] W. Stevens, *TCP/IP Illustrated Volume 3*. 1996. Addison-Wesley GmbH.
- [39] A. Tanenbaum, *Computernetzwerke*. 1996. Prentice Hall Inc.
- [40] T. Thormählen, *Konzeption und Realisierung einer Entwicklungsplattform für eine ISDN-Kamera*. März 1999. Studienarbeit bei Prof. Zimmer. IMS FhG.

-
- [41] G. K. Wallace, *The JPEG Still Picture Compression Standard*. 1991. IEEE Transactions on Consumer Electronics.
- [42] T. Wheeler, *Offene Systeme*. 1993. Vieweg + Sohn GmbH.
- [43] J. Wittmann, *Konzeption und Realisierung einer ISDN-Verbindungssteuerung in Anlehnung an den CAPI 2.0 Standard*. März 1999. Diplomarbeit bei Prof. Zimmer. IMS FhG.

Anhang A

Compact Disk zur Diplomarbeit

Abbildung A.1: Verzeichnisstruktur der CD

