Entwicklung eines „Transputerlink over Ethernet“ Datenbussystems

BACHELORARBEIT

für die Prüfung zum

Bachelor of Engineering

des Studienganges Informationstechnik

an der Dualen Hochschule Baden-Württemberg Mannheim

von

Christian König

19.09.2011

Bearbeitungszeitraum	12 Wochen
Matrikelnummer, Kurs	4207313, TIT08AIN
Ausbildungsfirma	Deutsches Zentrum für Luft- und Raumfahrt, Braunschweig
Betreuer der Ausbildungsfirma	Dipl.-Ing. (FH) Michael Przybilla
Gutachter der Dualen Hochschule	Dr. Mirko Wachs

Ehrenwörtliche Erklärung

Hiermit versichere ich, dass ich
die vorliegende Arbeit selbstständig
und nur unter Verwendung der als Quellen
angegebenen Hilfsmittel angefertigt habe.

Braunschweig, den 19.09.2011 _____
Christian König

Kurzfassung

Titel:	**Entwicklung eines „Transputerlink over Ethernet“ Datenbussystems**
Studierender:	**Christian König**
Matrikelnummer:	**4207313**
Kurs:	**TIT08AIN**
Hochschule:	Duale Hochschule Baden Württemberg Mannheim Studiengang Informationstechnik
Ausbildungsbetrieb:	Deutsches Zentrum für Luft- und Raumfahrt e.V. Standort Braunschweig
Betreuer:	Dipl.-Ing (FH) Michael Przybilla
Jahr:	2011

In der Abteilung Hubschrauber des Instituts für Flugsystemtechnik des DLR in Braunschweig werden Drehflüglersysteme entwickelt und getestet. Der Entwicklungsprozess beinhaltet Windkanaltests, bei denen unter anderem die selbstentwickelte Vielkanalmessanlage TEDAS II zum Einsatz kommt, die auf der Transputer-Technologie basiert.
Die zur Kommunikation mit dieser Messanlage verwendeten Hardwareschnittstellen haben ihre theoretische Einsatzdauer erreicht. Ein Ersatz ist am Markt nach bisherigem Wissensstand nicht verfügbar.
Um die langfristige Einsatzbereitschaft von TEDAS II zu gewährleisten, ist die Entwicklung einer transparenten, bidirektionalen Datenverbindung sowie eines User Interfaces erforderlich. Verwendet werden hierfür der Mikrocontroller BECK IPC SC13 sowie der für Transputer entwickelte Link-Adapter IMS C012.
Das entwickelte Datenbussystem EthLink ermöglicht die Konfiguration der Vielkanalmessanlage über ein Web-Interface. Weiterhin stellt es eine transparente Datenverbindung zwischen Ethernet und Transputerlink bereit.
Das EthLink Datenbussystem stellt einen vollwertigen Ersatz der verwendeten Hardwareschnittstellen dar und wird diese ersetzen. Die langfristige Einsatzbereitschaft von TEDAS II ist gewährleistet und die Möglichkeit zur Anpassung der Vielkanalmessanlage an neue Anforderungen ist durch die transparente Datenverbindung ebenfalls gegeben.

Abstract

Title:	**Development of a "Transputerlink over Ethernet" Databussystem**
Student:	**Christian König**
Matriculation no.:	**4207313**
Course:	**TIT08AIN**
University:	Baden-Wuerttemberg Cooperative State University Mannheim
Company:	German Aerospace Center Site Braunschweig
Tutor:	Dipl.-Ing (FH) Michael Przybilla
Year:	2011

The department rotorcraft of the institute of flight systems at the German Aerospace Center developes and tests rotorcraft-systems. During the developement process wind tunnel tests are performed using the in-house developed, transputer-based measuring system TEDAS II.
The hardware-interfaces used to control the measuring system have reached their theoretical period of use. A replacement is not commercially available.
To provide a long term usability of TEDAS II a transparent, bidirectional data link has to be developed as well as an user interface. The BECK IPC SC13 microcontroller and the IMS C012 link-adapter are utilized.
Using the developed databussystem EthLink, TEDAS II can be configured via web-interface. Furthermore, a transparent data link between ethernet and transputerlink is provided.
The EthLink databussystem is an equivalent replacement for the used hardware-interfaces and will replace them. The long term applicability of TEDAS II is ensured and it can be adapted to new requirements via the transparent data link.

Inhaltsverzeichnis

Abkürzungsverzeichnis

CGI	**C**ommon **G**ateway **I**nterface
CISC	**C**omplex **I**nstruction **S**et **C**omputer
DMA	**D**irect **M**emory **A**ccess
EPROM	**E**rasable **P**rogrammable **R**ead-**O**nly **M**emory
EthLink	Transputer**link** over **Eth**ernet
FFT	**F**ast **F**ourier **T**ransformation
FPU	**F**loating **P**oint **U**nit
FTP	**F**ile **T**ransfer **P**rotocol
HTML	**H**yper**t**ext **M**arkup **L**anguage
ISO	**I**nternationale **O**rganization for **S**tandardization
LSB	**L**east **S**ignificant **B**it
MAC	**M**edia **A**ccess **C**ontrol
MTM	**M**ulti **T**ransputer **M**odul
OSI	**O**pen **S**ystem **I**nterconnection
PIO	**P**rogrammable **I**nput/**O**utput
RAM	**R**andom-**A**ccess **M**emory
RISC	**R**educed **I**nstruction **S**et **C**omputer
RTOS	**R**eal-**T**ime **O**perating **S**ystem
T-Link	Transputer**link**
TCP	**T**ransmission **C**ontrol **P**rotocol
TEDAS	**T**hroughput **E**nhanced **D**ata **A**cquisition **S**ystem

Abbildungsverzeichnis

Tabellenverzeichnis

Quellcodeverzeichnis

1 Einleitung

1.1 DLR

Das Deutsche Zentrum für Luft- und Raumfahrt (DLR) ist das Forschungszentrum der Bundesrepublik Deutschland für die Bereiche Luftfahrt, Raumfahrt, Verkehr und Energie. Es fungiert laut [6] als Raumfahrtagentur der Bundesregierung und kooperiert in diesem Bereich, genau wie in der Forschung, national und international. So betreibt das DLR beispielsweise das Kontrollzentrum des Columbus-Labors der ISS. Die Ziele des DLR sind Erforschung von Erde und Sonnensystem, die Forschung für den Erhalt der Umwelt sowie die Entwicklung umweltverträglicher Technologien im Bereich der Mobilität, Kommunikation und Sicherheit.

Das DLR betreibt in Deutschland 31 Institute bzw. Test- und Betriebseinrichtungen, die sich auf 15 Standorte verteilen. International werden Büros in Brüssel, Paris und Washington D.C. unterhalten. Insgesamt beschäftigt das DLR etwa 6900 Mitarbeiter.

Die vorliegende Arbeit wurde im Bereich der Luftfahrt und zwar am Institut für Flugsystemtechnik in der Abteilung Hubschrauber erstellt.

Abbildung 1: Forschungsflotte des DLR [7]

1.2 Abteilung Hubschrauber - Gruppe Rotorversuchsstand

Die Abteilung Hubschrauber des Instituts für Flugsystemtechnik befasst sich mit Drehflüglern im Allgemeinen und Hubschraubern im Besonderen. Das Aufgabengebiet umfasst die Forschung und Entwicklung im Bereich der Rotorsystem- und Flugsystemtechnik. Die Ziele der Abteilung sind wie folgt gewählt:

> „Zielrichtung der Aktivitäten ist die Unterstützung der Entwicklung neuer Technologien und deren Bewertung im Hinblick auf die Verbesserung der Flugeigenschaften, der Sicherheit, der Wirtschaftlichkeit, der Umweltverträglichkeit und der Missionseffektivität für die Fluggeräte.“ [7]

Die während der Forschungs- und Entwicklungsarbeit durchgeführten Aktivitäten reichen von der Systemanalyse, der Modellierung und Simulation bis hin zu Windkanalversuchen sowie der Erprobung im realen Flug.

Die Windkanaltests werden im Rotorversuchsstand vorbereitet. In Abbildung 2 ist dieser während einer Messung zu sehen.

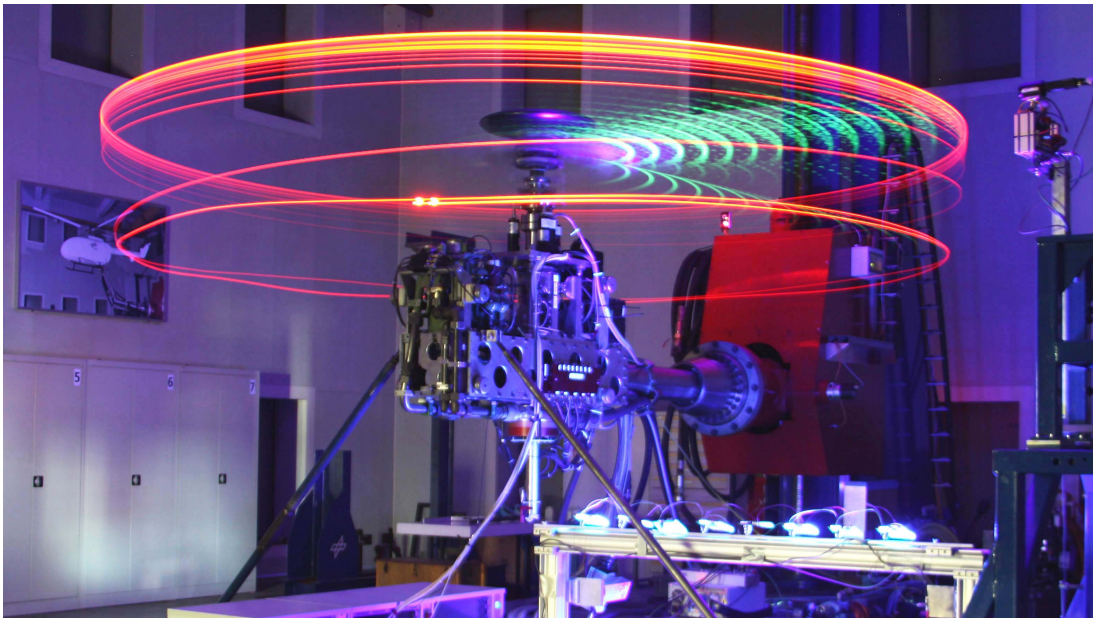

Abbildung 2: Rotorversuchsstand

Im Rotorversuchsstand sind mehrere große Messanlagen installiert, unter anderem auch Throughput Enhanced Data Acquisition System (Abk.: TEDAS) I und TEDAS II, bei denen es sich um Vielkanalmessanlagen handelt. Mit diesen können nahezu in Echtzeit die Daten von

Druckmessungen ausgewertet werden. Eine detaillierte Beschreibung dieser Vielkanalmesssysteme befindet sich in Abschnitt 1.3.

1.3 TEDAS II

TEDAS II ist eine extern getaktete Vielkanalmessanlage im Rotorversuchsstand. Eine eingehende Beschreibung kann [8] entnommen werden.

TEDAS II verarbeitet Messungen am drehenden Rotorblatt. Auf die gemessenen Werte wird anschließend eine Fast Fourier Transformation (Abk.: FFT) angewendet. Im Gegensatz zu gewöhnlichen Messanlagen wird nicht mit einer konstanten Taktrate gearbeitet. Stattdessen wird die Taktrate aus einer externen Quelle zugeführt und ist abhängig von der Drehgeschwindigkeit des Rotors. Abbildung 3 stellt TEDAS II und eine konventionelle Messanlage gegenüber.

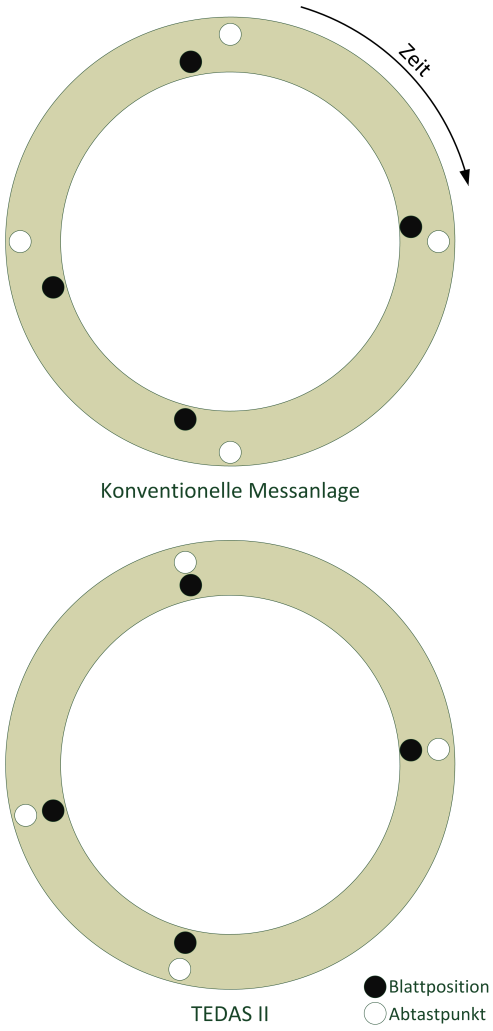

Abbildung 3: TEDAS II und konventionelle Messanlage im Vergleich

Ein Rotor dreht nicht mit einer exakt konstanten Geschwindigkeit. Aerodynamische und mechanische Kräfte sind mit der Drehung korreliert, sodass der exakte Zeitpunkt der Messung von Bedeutung ist.
Wie in Abbildung 3 erkennbar, bewirkt die veränderliche Drehgeschwindigkeit, dass nicht exakt beim Durchgang des Rotorblattes gemessen wird. Konventionelle Messanlagen mit konstanten Taktraten messen somit unter Umständen neben der Blattposition, was zu Phasensprüngen zwischen Umdrehungen und einer Verfälschung der Ergebnisse führt. Bei der folgenden FFT ist eine Fensterfunktion zur Wichtung der Abtastpunkte notwendig.
Bei Verwendung der an die Drehzahl gekoppelten Taktrate in TEDAS II werden die Messpunkte zum exakt richtigen Zeitpunkt abgetastet, beim Durchgang des Roborblatts. Somit ist keine Fensterfunktion nötig.

Abbildung 4 zeigt die Messanlage TEDAS II.

Abbildung 4: TEDAS II

1.4 Aufgabenstellung

Die am Rotorstand eingesetzten Messanlagen sind in ihrer Form sehr speziell und wurden daher auch schon vor längerer Zeit im Institut für Flugsystemtechnik entwickelt. Beispiele sind hier die Vielkanalmessanlagen TEDAS I und TEDAS II. Für die Messungen wird beispielsweise ein extern getaktetes Vielkanalmesssystem benötigt. Ein entsprechendes serienfertiges System, welches am Markt gekauft werden könnte, war damals und ist auch heute nach bisherigem Wissenstand nicht verfügbar.

Die Computertechnologie besitzt eine hohe Entwicklungsgeschwindigkeit, sodass teilweise Technologien, wie beispielsweise Transputer, sich nicht lange am Markt behaupten können. Dementsprechend sind diese Systeme auch inzwischen vom Markt verschwunden.
Als Konsequenz daraus ergibt sich für den Betrieb von Anlagen, dass neue Schnittstellen entwickelt werden müssen, die die Messanlagen zum heutigen Stand der Technik kompatibel macht.

Die Schwerpunkte des Projektes sind wie folgt gewählt:

- Analyse der Hardwarespezifikationen der Schnittstelle „Transputerlink“.
- Adaption der Hardwareschnittstelle auf dem Mikrocontrollersystem BECK IPC.
- Entwicklung einer transparenten Datenverbindung, welche die Daten eines Transputerlink vom BECK IPC auf ein PC Programm und umgekehrt weiterleitet.
- Analyse der Befehlsstruktur des Transputerlink.
- Nachbildung dieser Befehle in einem User Interface, um eine Kommunikation zwischen PC und Transputer herzustellen.

1.5 Problembeschreibung

Aktuell werden die Messanlagen über spezielle Hardwareschnittstellen an Steuercomputern angeschlossen. Das Kernproblem liegt darin, dass sowohl die Steuercomputer als auch die Schnittstellenhardware das Ende ihrer theoretischen Einsatzdauer erreicht haben. Ein Ausfall wird somit immer wahrscheinlicher. Während genügend Ersatzteile für die Steuercomputer verfügbar sind, gilt dies nicht für die Schnittstellenhardware.

1.6 Ausgangsdaten

Sowohl Hardware als auch Softwarespezifikationen und Befehlssatz der alten Transputertechnik sowie die neue Technik der BECK IPC sind in Form von Datenblättern vorhanden. Ein experimenteller Aufbau kann mit Teilen aus dem Lagerbestand und Neuteilen realisiert werden.

Über die Messanlagen existiert eine umfangreiche Dokumentation, welche auch Softwarebeispiele enthält.

Das Ziel der Bachelorarbeit soll ein System sein, das sich für die Kommunikation von vorhandener Software mit der Messanlage vollkommen transparent verhält.
Ebenso öffnet dieser Ansatz aber auch alle Möglichkeiten für die Entwicklung von neuen Softwareanwendungen.

1.7 Schwerpunkte der Arbeit

1. Analyse der Hardwarespezifikationen der Schnittstelle „Transputerlink“

2. Adaption der Hardwareschnittstelle auf dem Mikrocontrollersystem BECK IPC. Diese Schnittstelle muss verstanden und in einen Schaltplan umgesetzt werden.
   Danach gilt es, den geeigneten BECK IPC Baustein aus einer Reihe von Möglichkeiten auszuwählen und funktional in den Schaltplan zu integrieren Mit den I/O Pins des BECK IPC muss nun das Verhalten des Transputerlink nachgebildet werden.

3. Entwicklung einer transparenten Datenverbindung, welche die Daten eines Transputerlink vom BECK IPC auf ein User Interface und umgekehrt weiterleitet. Am Ende muss ein Befehl oder Datenwort, welches über Ethernet zum BECK IPC geschickt wird, von diesem so aufbereitet und in den Transputerlink geschickt werden, dass der Transputer es fehlerfrei versteht.

4. Analyse der Befehlsstruktur des Transputerlink und Nachbildung dieser Befehle im User Interface, um eine Kommunikation zwischen PC und Transputer herzustellen. Um die Schnittstelle auch für neue Programme nutzen zu können, muss der Befehlssatz der Transputer sowie die Funktion der Befehle bekannt sein.

2 Theoretische Grundlagen und Entwicklungsumgebung

2.1 Transputer

Der Transputer basiert auf einer Mikroprozessor-Architektur, die vor allem in den 1980er Jahren Verwendung fand. Der Name setzt sich zusammen aus *Transistor* und *Computer*. Der Transputer wurde speziell für Multiprozessorsysteme mit Nachrichtenaustausch zwischen den einzelnen Prozessoren entwickelt. Weiterhin existiert zwischen den einzelnen Transputern kein gemeinsamer Speicher. Eine detaillierte Beschreibung dieser Technologie erfolgt unter [12].

2.1.1 Architektur des Transputers

Die Entwicklung des Transputers begann Ende der 1970er Jahre. Zu dieser Zeit wurden große Fortschritte in der Halbleitertechnik erzielt, wie beispielsweise die Erhöhung der Anzahl der Schaltkreise auf einem Chip. Wie in [1] beschrieben, ermöglichte es diese Erhöhung, ehemals externe Bauteile zusammen mit dem Mikroprozessor auf einem Chip zu vereinen. Bei einem Transputer sind Prozessor, Speicher, Timer, Scheduler und externe Schnittstellen in einem Chip zusammengefasst. Die externen Schnittstellen sind zum einen das Link-Interface zur Kommunikation mit anderen Transputern und zum anderen eine Schnittstelle, mit der dem internen Speicher weiterer externer hinzugefügt werden kann.

Das Black Box Verhalten aller Transputer ist gleich. Sie können gegeneinander ausgetauscht und in beliebigen Kombinationen in einem Transputer-Netz verbaut werden. Dies rührt daher, dass das Link-Interface eine standardisierte Schnittstelle ist. Allerdings variert die Architektur und kann nach [12] in zwei Generationen eingeteilt werden:

- *Erste Generation:*
  - 16 bit Prozessoren ohne Floating Point Unit (Abk.: FPU)
  - 32 bit Prozessoren mit FPU
- *Zweite Generation*
  - 64 bit Prozessoren mit FPU

In Abbildung 5 ist die Architektur des Transputers T800 dargestellt. Dieser Transputer der ersten Generation basiert auf einem 32 bit Prozessor und einer FPU. Neben der erwähnten FPU verfügt er über einen sequteniellen RISC-Prozessor, 4 kB RAM Speicher, einen Timer, vier

Link-Interfaces, eine Schnittstelle für eine externe Speichererweiterung sowie verschiedene System- und Link-Services. Zu den System-Services gehören, wie in [9] beschrieben, unter anderem Funktionen zur Steuerung des Boot- und Reset-Vorgangs.

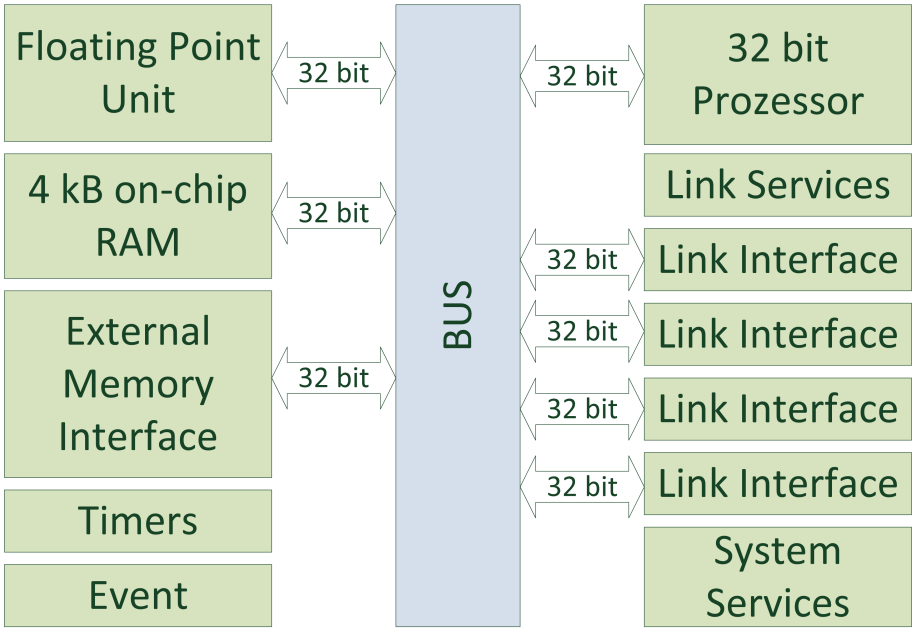

Abbildung 5: Architektur des Transputers IMS T800 nach [9]

Die internen Komponenten werden in dieser Arbeit nicht eingehender beschrieben, da sie nicht von Relevanz für dieses Projekt sind. Einzig auf die Link-Interfaces wird genauer eingegangen, da sie die Schnittstelle zu anderen Transputern darstellen und somit auch zum *Transputerlink over Ethernet* Datenbussystem, im Folgenden kurz EthLink genannt, welches Gegenstand dieser Arbeit ist.

2.1.2 Link-Interfaces

Das Link-Interface dient zur seriellen Kommunikation zwischen genau zwei Kommunikationspartnern. Sie sind die Endpunkte der Transputerlink (Abk.: T-Link)-Kommunikation. Es handelt sich im Prinzip um serielle Schnittstellen mit Direct Memory Access (Abk.: DMA). Bei einem Lesevorgang werden die Daten auf der seriellen Leitung empfangen und direkt in den Speicher geschrieben, beim Schreibvorgang verhält es sich genau umgekehrt. Der Speicherzugriff der Link-Interfaces wird von einem DMA-Controller koordiniert, sämtliche Link-Interfaces sind hierbei gleichberechtigt. Im Gegensatz zu den Link-Interfaces ist der Prozessor priviligiert beim Speicherzugriff. Führt dieser gerade eine Lese- oder Schreiboperation auf den Speicher aus, ist kein Zugriff durch die Link-Interfaces möglich. Dieses in [9] erläuterte Verhalten ist in Abbildung 6 dargestellt.

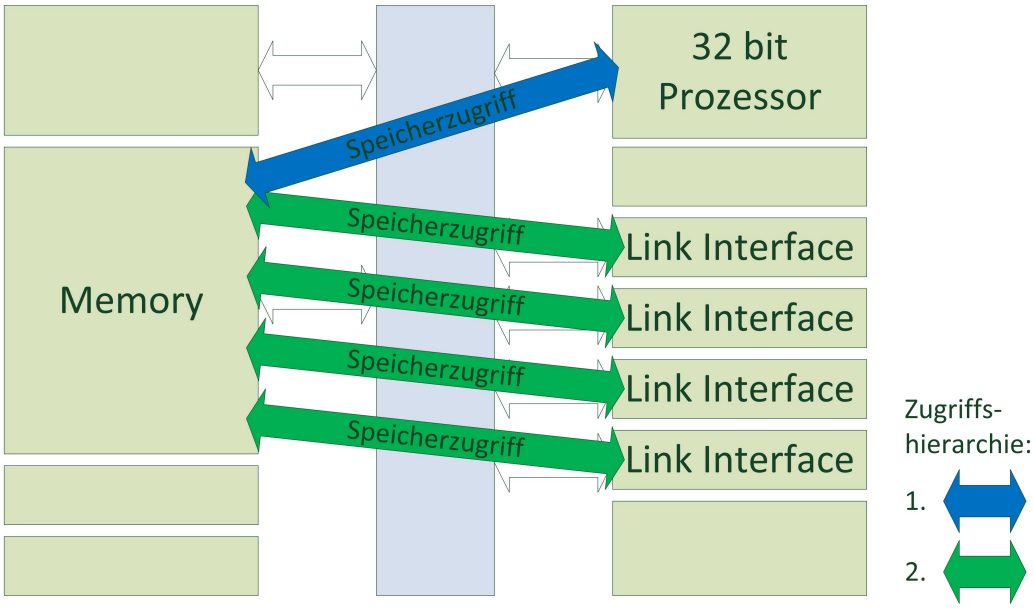

Abbildung 6: Internes Speicherzugriffsverhalten von Transputern nach [12]

In einem Transputer-Netz werden in der Regel beide Kommunikationsendpunkte durch die Link-Interfaces eines Transputers dargestellt.
Da ein Transputer über vier Link-Interfaces verfügt, können auch vier Kommunikationspartner angeschlossen werden. Ein Transputer kann also mit bis zu vier weiteren Transputern eine serielle Kommunikation betreiben. Dies ist in Abbildung 7 schematisch dargestellt.

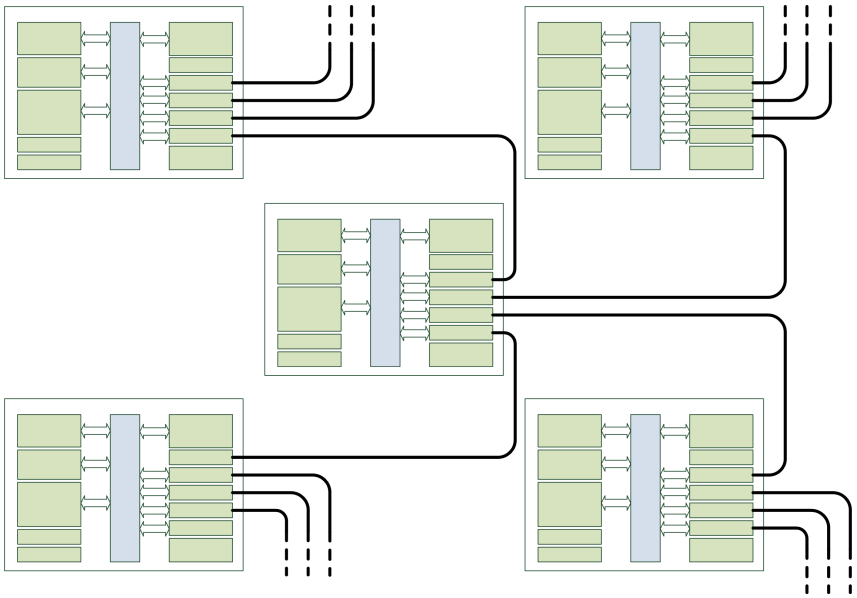

Abbildung 7: Kommunikation zwischen Transputern

Das Ziel dieser Bachelorarbeit ist die Entwicklung eines EthLink Datenbussystems. Dieses kann ebenfalls einen Kommunikationsendpunkt darstellen und so einen Transputer aus Abbildung

7 ersetzen. Allerdings verfügt es nur über ein einziges Link-Interface. Abbildung 8 zeigt das resultierende System.

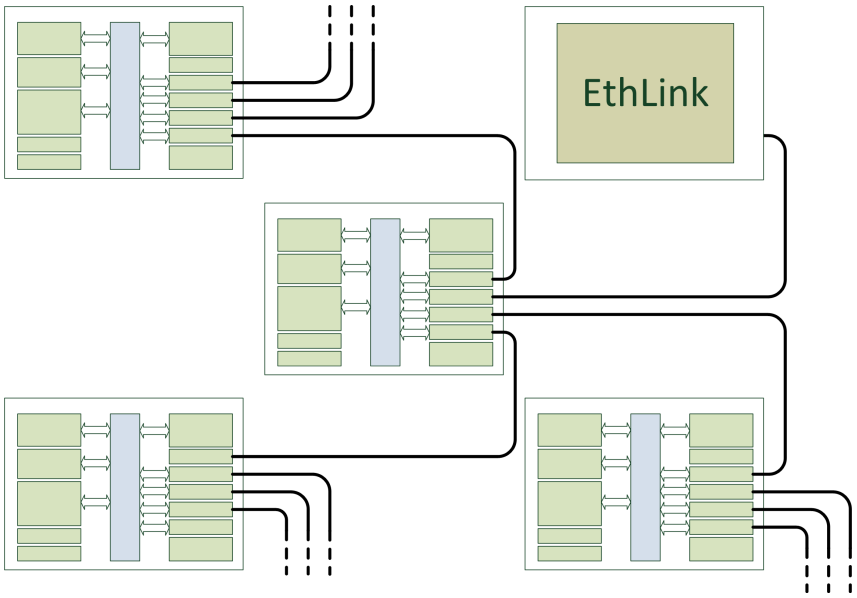

Abbildung 8: Kommunikation zwischen Transputern und EthLink

2.1.3 Transputerlink-Protokoll

Das in [12] eingehend beschriebene Transputerlink- (Abk.: T-Link-)Protokoll legt fest, auf welche Weise Daten über die Link-Interfaces ausgetauscht werden.
Die serielle Übertragung der Datenpakete wird byteweise bestätigt, es liegt also ein Hand-Shake-Verfahren vor. Nach jedem übertragenen Byte wartet der Sender auf die Empfangsbestätigung des Empfängers. Aus diesem Grund ist ein 1-Byte Buffer im Link-Interface ausreichend. Zu beachten ist, dass es sich lediglich um eine Empfangsbestätigung handelt. Die Korrektheit der Daten wird an dieser Stelle nicht überprüft, um die Kommunikation zu beschleunigen. Fehlererkennungs- und -korrekturverfahren können bei Bedarf softwareseitig implementiert werden.

Die mittels des T-Link-Protokolls übertragenen Pakete werden als T-Link-Pakete bezeichnet. Jedes T-Link-Paket beginnt mit einem `high`-Signal als Startbit. Ein `high`-Signal entspricht einer logischen `1`, ein `low`-Signal einer logischen `0`. Das dem Startbit folgende Bit zeigt an, ob es sich um ein Datenpaket (2. Bit `high`) oder eine Acknowledgementpaket (2. Bit `low`) handelt. In einem Datenpaket werden nach diesen zwei Bits acht Datenbits gesendet. Abgeschlossen wird das Datenpaket von einer logischen `0` als Stopbit. Ein Acknowledgementpaket endet bereits nach dem zweiten Bit.

In Abbildung 9 sind die beiden Pakettypen inklusive der Meta-Daten dargestellt.

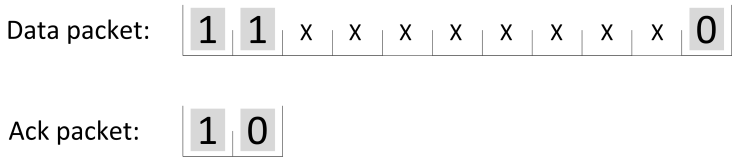

Abbildung 9: T-Link Daten- und Acknowledgement-Packet

Für die Übertragung ist laut [9] keine Synchronisation zwischen den einzelnen Transputern erforderlich, beide Kommunikationspartner müssen lediglich eine gemeinsame Bitrate aufweisen. Dementsprechend kann ein Transputer-Netz wahlweise von einem zentralen Taktgeber gesteuert werden oder jeder einzelne Transputer hat einen eigenen. In der Regel wird eine Bitrate von 10 Mbit/s oder 20 Mbit/s verwendet. Vereinzelt kommt auch 5 Mbit/s zum Einsatz.

2.1.4 Programmiersprache Occam

Die Programmiersprache *Occam* wird in diesem Projekt zur Erstellung von Testfällen eingesetzt (siehe Abschnitt 5.1). Sie zählt sowohl zu den imperativen als auch zu den prozeduralen und parallelen Programmiersprachen und wurde speziell für parallele Programmierung auf Transputern entwickelt.

Ein in Occam implementiertes Programm kann in *Prozeduren* unterteilt werden. Das Schlüsselwort ist `PROC`. Prozeduren enthalten Konstrukte, die sich in *Channels*, *Inputs* und *Outputs* untergliedern lassen. Ein Channel entspricht einem Thread. Input und Output dienen zur Kommunikation zwischen den Channels.
Für die Kommunikation zwischen den Channels ist es irrelevant, ob sich diese auf demselben oder auf verschiedenen Transputern befinden. Dementsprechend können mehrere Channels auf einem Transputer im Time-sharing-Verfahren ablaufen. Der Scheduler des Transputers übernimmt die Zuweisung der Prozessorzeit.

In Occam geschriebene Programme lassen sich laut [12] auf drei Primitive zurückführen:

- Zuweisung des Ausdrucks `e` an die Variable `v`:
  `v := e`
- Ausgabe des Ausdrucks `e` an Channel `c`:
  `c ! e`
- Zuweisung des von Channel `c` erhaltenen Ausdrucks an die Variable `v`
  `c ? v`

Die aus diesen Primitiven zusammengesetzten Konstrukte können wieder Bestandteil weiterer Konstrukte sein. Wie in [13] beschrieben, lassen sie sich auf verschiedene Art und Weise miteinander verknüpfen. Dies erfolgt mit den Schlüsselwörtern `SEQ`, `PAR` und `ALT`. `SEQ` bewirkt ein sequentielles Ausführen des Konstruktes, `PAR` führt zu einem parallelen Abarbeiten. `ALT` bietet eine Alternative. Es wird der Befehl ausgeführt, für den als erstes alle Vorbedingungen erfüllt sind. Sind mehrere Befehle gleichzeitig bereit, so wird der ausgeführt, dessen Vorbedingungen zuerst überprüft werden. Weiterhin stehen die Kontrollstrukturen `IF` und `WHILE` zur Verfügung.

Die Deklaration einer Variablen unterscheidet sich nicht von anderen Programmiersprachen. Anweisungen, die außerhalb eines Konstruktes stehen, werden mit einem Doppelpunkt beendet. Ein Integer wird wie folgt deklariert: `INT i:`. Bei einem Array kann die Größe als Präoder als Postfix geschrieben werden: `[5]INT arr:` bzw. `INT[5] arr:`. Wird die Größe nicht angegeben, handelt es sich um ein Array mit dynamischer Größe: `[]INT arr:` bzw. `INT[] arr:`.
Kommentare werden durch `--` gekennzeichnet. Die Strukturierung erfolgt durch Einrückung. Geschweifte Klammern oder `BEGIN` - `END` Blöcke werden nicht verwendet.

Die folgenden Beispiele zeigen das Beschriebene. In den Quellcode-Beispielen stehen `link1`, `link2` sowie `link3` für Channels.

In Quellcode 1 ist die sequentielle Ausführung zu sehen.

```
1  INT x:
2  INT y:
3     SEQ -- Sequentielle Ausführung
4        x := 2
5        y := 3
6        x := x + y
7        y := 2 * x
```

Quellcode 1: Occam Beispiel: SEQ

Zunächst werden die Variablen `x` und `y` deklariert (Zeile 1-2). Anschließend gibt das Schlüsselwort `SEQ` an, dass eine sequentielle Ausführung vorgenommen wird (Zeile 3). Nachdem die Variablen mit einem Wert initialisiert wurden (Zeile 4-5), werden mathematische Operationen auf diesen ausgeführt. Die Variable `x` enthält am Ende den Wert `5`, in `y` ist der Wert `10` gespeichert.

Die paralle Ausführung, für die ein Netz aus Transputern prädestiniert ist, wird in Quelltext 2 gezeigt.

```
1  INT x:
2  INT y:
3     PAR -- Parallele Ausführung
4        x := 3
5        y := 4
```

Quellcode 2: Occam Beispiel: PAR

Die deklarierten Variablen `x` und `y` werden mit Werten initialisiert (Zeile 4-5). Die Initialisierungen der Variablen sind voneinander unabhängig, somit können sie parallel ausgeführt werden. Dies wird, wie erwähnt, durch das Schlüsselwort `PAR` ausgelöst.

Quellcode 3 zeigt die Verwendung des Schlüsselwortes `ALT`.

```
1  INT x:
2     link1 ? x
3     ALT -- Alternative
4        x == 2
5           link2 ! x
6        x == 3
7           link3 ! x
```

Quellcode 3: Occam Beispiel: ALT

Der von Channel `link1` erhaltene Ausdruck wird in `x` gespeichert (Zeile 2). Abhängig von dem in `x` gespeicherten Wert wird dieser entweder an den Channel `link2` oder `link3` ausgegeben (Zeile 3-7).

In Quellcode 4 ist ein *Hello World* Programm dargestellt. Es wird vorausgesetzt, dass es sich bei dem Channel `link1` um einen Bildschirm handelt.

```
1  PROC HelloWorld()
2     []BYTE message:
3     SEQ
4        message := "Hello World"
5        link1 ! message -- Ausgabe an den Bildschirm
```

Quellcode 4: Occam Beispiel: Hello World

Zu Beginn wird der Name der Prozedur festgelegt (Zeile 1), bevor ein Byte-Array mit dynamischer Größe deklariert wird (Zeile 2). Nachdem der String `Hello World` in dieser Variablen gespeichert wurde, wird dieser String an den Channel `link1` ausgegeben. Da der Channel `link1` ein Bildschirm ist, wird der String nun angezeigt.

2.2 Mikrocontroller

Die Erläuterungen bezüglich Mikrocontroller ist an [4] angelehnt. Ein Mikrocontroller ist ein eigenständiges Rechnersystem, das sich vollständig oder nahezu vollständig in einem einzigen Halbleiterchip befindet. Neben einem Mikroprozessor sind beispielsweise auch Analog-Digital-Wandler, Interruptcontroller und Schnittstellenbausteine integriert. Weiterhin sind oft Arbeits- und Programmspeicher in diesem Chip verbaut. Der Speicher ist in der Regel ein Flash oder EPROM, sodass wiederholtes Aufspielen von Programm-Code während der Entwicklungsphase möglich ist.

Die für Mikrocontroller-Programmierung häufig verwendeten Programmiersprachen sind C und Assembler, allerdings sind auch andere Sprachen möglich. Welche Programmiersprachen zulässig sind, ist vom verwendeten Mikrocontroller abhängig.

Mikrocontroller finden in vielen elektrischen Geräten Verwendung. Beispiele sind Fernbedienungen, Klimaanlagen oder der Chip auf einer EC-Karte.

2.3 Programmiersprache C und Bibliothek CLIB.H von Beck-IPC

Die Programmiersprache C wurde von Dennis Ritchie im Jahr 1972 entwickelt. Die Hochsprache C zählt zu den imperativen und strukturierten Programmiersprachen. Eine detaillierte Beschreibung der Programmiersprache erfolgt unter [11].
Da C auch eine hardwarenahe Programmierung ermöglicht, wird es oft für Mikrocontroller verwendet. Allerdings muss hierfür eine spezielle, zur Architektur des Mikrocontrollers passende Bibliothek bereitgestellt werden.

Die Bibliothek `CLIB.H` von BECK IPC ist eine solche Bibliothek. Sie enthält eine Reihe von Funktionen speziell für die Mikrocontroller der Firma BECK IPC. Beispiele hierfür sind die Ansteuerung der I/O-Pins, das Aktivieren bzw. Deaktivieren des Datenbusses oder das Senden und Empfangen von Bytes über diesen.
Auch eine Reihe von gebräuchlichen C-Funktionen liegen in einer speziell für BECK IPC Mikrocontroller modifizierten Form vor. Hierzu zählen beispielsweise die Netzwerk-Funktionen wie das Öffnen eines Sockets oder das Senden von Daten per TCP/IP.
Die Funktionen aus `CLIB.H` rufen einen Inline-Assembler-Code auf. Der Inline-Assembler-Code kann auch direkt im C-Code verwendet werden, allerdings mindert dies die Übersichtlichkeit des Quelltextes. Eine komplette Auflistung der Funktionen in der Bibliothek `CLIB.H` sowie der entsprechenden Inline-Assembler-Codes kann [3] entnommen werden.

2.4 Verwendete Software

2.4.1 Borland C++ und Borland C++ Compiler

Borland C++ ist eine integrierte Entwicklungsumgebung. Sie unterstützt die Programmiersprachen C und C++.

Bei dem Borland C++ Compiler handelt es sich um einen Cross-Compiler. Mit ihm können Programme auf einem Host-System erstellt werden, die auf einem Zielsystem mit einem anderen Betriebssystem lauffähig sind. Dies erhält eine besondere Bedeutung, wenn es sich bei der Zielplattform um ein eingebettetes System handelt, auf dem die Kompilierung aus Performance-Gründen nicht möglich ist.

2.4.2 OrCAD

OrCAD ist ein Programmpaket zur Entwicklung von elektrischen Schaltungen. Die verwendeten Bestandteile dieses Pakets sind OrCAD Capture und OrCAD Layout. OrCAD Capture wird zum Erstellen von Schaltplänen verwendet, OrCAD Layout zum Routen der Platine.

2.4.3 CHIPtool

Die Software CHIPtool von BECK IPC dient zur Netzwerk-Konfiguration der Mikrocontroller dieses Herstellers. Weiterhin stellt es eine Telnet- und ein FTP-Client zur Verfügung.

3 Auswahl und Beschreibung der verwendeten Hardware

3.1 Definition der geforderten Hardware-Eigenschaften

Die Aufgabe des EthLink Datenbussystems ist, zwischen Ethernet und Transputerlink (Abk.: T-Link) zu übersetzen. Hieraus folgt zwingend, dass es sowohl eine Ethernet- als auch eine T-Link-Schnittstelle aufweisen muss. Ist eine der Schnittstellen nicht vorhanden, so muss diese simuliert werden.

Das Vorhandensein einer Ethernet-Schnittstelle gehört zu den Eigenschaften vieler Mikrocontroller. Eine Hardware-Schnittstelle kann also als Anforderung vorausgesetzt werden.
Das T-Link-Protokoll jedoch findet seit Anfang der 1990er Jahre nahezu keine Verwendung. Dementsprechend kann nicht von der Verfügbarkeit eines Mikrocontrollers ausgegangen werden, der dies als Hardware-Schnittstelle aufweist. Einen Ersatz zur T-Link-Schnittstelle des Mikrocontrollers stellt der Link-Adapter IMS C012 der Firma Inmos dar. Dieser konvertiert zwischen T-Link und einem 8 bit A/D-Bus. Sowohl 10 Mbit/s als auch 20 Mbit/s sind als Geschwindigkeit auswählbar. Eine ausführliche Beschreibung ist Abschnitt 3.4 zu entnehmen.

Eine Alternative stellt die Simulation der T-Link-Schnittstelle mittels der I/O-Pins des Mikrocontrollers dar. Um die exakten Anforderungen an diese I/O-Pins zu ermitteln, sind einige Überlegungen notwendig:

- Das T-Link-Protokoll verfügt über zwei verschiedene Übertragungsraten, die von den Transputern in TEDAS II unterstützt werden. Dies sind 10 Mbit/s und 20 Mbit/s. Zur Kommunikation über dieses Protokoll müssen die I/O-Pins dementsprechend eine Taktrate von 10 Mbit/s bzw. 20 Mbit/s aufweisen.

- Die Transputer in der TEDAS-II-Messanlage sind auf 20 Mbit/s eingestellt. Die Veränderung der Taktrate auf dem speziellen T-Link-Segment, an den EthLink angeschlossen wird, ist möglich, sollte jedoch vermieden werden. Folglich sollten die I/O-Pins die Taktrate 20 Mbit/s unterstützten.

- Die gewünschte Taktrate von 20 Mbit/s bedeutet, dass ein Bit exakt 50 ns auf dem Bus anliegt. Ist eine Taktrate von 20 Mbit/s nicht möglich, muss auf 10 Mbit/s zurückgegriffen werden. Hier hat ein Bit eine Dauer von 100 ns.

Die jeweilige Bitrate muss exakt eingehalten werden. Sowohl beim Lesen als auch beim Schreiben auf dem T-Link ist harte Echtzeit erforderlich. Das Setzen eines I/O-Pins besteht aus

mehreren internen Vorgängen. Liegt das zu sendende Byte als String aus `0` und `1` vor, müssen folgende Vorgänge durchgeführt werden:

1. Zeiger zu nächstem Bit verschieben
2. Einlesen des Bit aus dem String
3. Wert mit `0` oder `1` abgleichen
4. Vergleichen des aktuellen Timestamp mit dem Referenzwert, gegebenenfalls Ausführen einer Nulloperation
5. Speichern des aktuellen Timestamp
6. Setzen des I/O-Pins auf den entsprechenden Wert

Bei den folgenden Überlegungen wird vorausgesetzt, dass jede dieser Operationen den Prozessor für genau einen Taktzyklus belegt:
Werden nur diese Operationen durchgeführt, ist eine sechsmal höhere Prozessor-Taktrate des Mikrocontrollers nötig als vom T-Link-Protokoll verwendet wird. Da in der Regel jedoch parallel weitere Aufgaben wie der Betriebssystem-Task ablaufen, ist dies nicht ausreichend. Die interne Taktrate sollte noch einmal vervierfacht werden, sodass von einem Erfüllen der Echtzeitbedingungen ausgegangen werden kann. Für eine T-Link-Taktrate von 20 Mbit/s sollte der Mikrocontroller also mindestens 480 MHz aufweisen, für 10 Mbit/s sind 240 MHz nötig. Selbst mit einer entsprechenden Taktrate muss ausgiebig getestet werden, ob die Echtzeitbedingungen tatsächlich stets eingehalten werden.

Die Anforderungen lassen sich also wie folgt zusammenfassen:

- Ethernet-Schnittstelle
- wahlweise (absteigend nach Eignung sortiert)
    1. T-Link-Schnittstelle
    2. paralleler 8 bit A/D-Bus
    3. mindestens 480 MHz Taktrate
    4. mindestens 240 MHz Taktrate

3.2 Auswahl des Mikrocontrollers

Die zur Verfügung stehenden Mikrocontroller des Herstellers BECK IPC (siehe Aufgabenstellung in Abschnitt 1.4) sind die in Tabelle 1 dargestellten Modelle.

Modell	Ethernet	T-Link	8 bit A/D-Bus	Taktrate
SC11	nein	nein	ja	40 MHz
SC12	nein	nein	ja	20 MHz
SC13	ja	nein	ja	40 MHz
SC2x	ja	nein	nein	96 MHz
SC1x3	ja	nein	ja	96 MHz
SC243	ja	nein	nein	400 MHz

Tabelle 1: Mikrocontroller-Modelle von BECK IPC

Aus Tabelle 1 ist ersichtlich, dass keines der Modelle über eine T-Link-Schnittstelle verfügt. Folglich muss entweder der Link-Adapter IMS C012 verwendet oder die Schnittstelle mit den I/O-Pins simuliert werden.
Die Modelle SC11 und SC12 verfügen nicht über eine Ethernet-Schnittstelle und scheiden somit aus. Die Modelle der Serie SC2x weisen weder einen 8 bit A/D-Bus noch eine ausreichend hohe Taktrate auf. Dementsprechend sind auch diese nicht geeignet.
Von den verbleibenden Modellen verfügen SC13 und SC1x3 über einen 8 bit A/D-Bus. Das Modell SC243 ist nicht mit solch einem Bus ausgestattet, allerdings ist die Taktrate hier mit 400 MHz ausreichend für die T-Link-Taktrate von 10 Mbit/s.

Zur Realisierung des EthLink Datenbussystems wurden der BECK IPC SC13 und der Link-Adapter IMS C012 gewählt. Im Vergleich zum SC243 ist mit dieser Kombination auch die höhere Taktrate von 20 Mbit/s möglich. Der SC13 ist im Gegensatz zum SC1x3 im Institut für Flugsystemtechnik vorrätig und muss nicht beschafft werden.

3.3 Beschreibung des Mikrocontrollers BECK IPC SC13

In diesem Projekt wird der Mikrocontroller SC13 von BECK IPC verwendet. Die in diesem Abschnitt beschriebenen Eigenschaften des Mikrocontrollers sind dem Hardware Handbuch [2] sowie der Dokumentation des Betriebssystems [3] des Herstellers entnommen. Eine aus [2] entnomme Abbildung des Mikrocontroller ist in Abbildung 10 dargestellt.

Der Mikrocontroller verfügt als Betriebssystem über ein Real-Time Operating System (Abk.: RTOS) mit Dateisystem. Durch das Dateisystem ist es möglich, vom Betriebsystem unabhängige Anwendungen zu erstellen.

Abbildung 10: Mikrocontroller BECK IPC SC13 [2]

In diesem Betriebsystem sind ein FTP- und ein Webserver integriert. Ebenfalls vorhanden ist ein Telnetserver, der den Zugriff auf die Kommandozeile des Mikrocontrollers ermöglicht.

Der SC13 enthält einen Intel 80186-kompatiblen Prozessor mit 40 MHz, der aufgrund seines Aufbaus für die Verwendung in Mikrocontrollern besonders geeignet ist. Unter anderem weist er einen geringen Stromverbrauch (300 mA) und eine niedrige benötige Versorgungsspannung (5 V) auf. Der Mikrocontroller verfügt über 512 kB Arbeitsspeicher sowie 512 kB Flashspeicher. Ein Teil des Flashspeichers wird vom RTOS des Mikrocontrollers belegt, der restliche Speicher steht für Anwendungsprogramme zur Verfügung.
Der SC13 verfügt über eine Reihe an Interfaces. Zu den in diesem Projekt genutzten Schnittstellen gehört die 10/100Base-T Ethernet-Schnittstelle. Diese ist intern unterteilt in die Schichten 1 und 2 des ISO OSI Referenzmodells. Auf Schicht 2 befindet sich die Media Access Control (Abk.: MAC), auf Schicht 1 die physikalischen Schnittstellen, also die Pins.

Ebenfalls verwendet wird der gemultiplexte 8 bit A/D-Bus, der sechs Chip Selects bietet. Für jeden Chip Select stehen 256 verschiedene Adressen zur Verfügung, die letzen drei Bit der Adresse können auf separaten Steuerleitungen abgegriffen werden. Ein Lese- bzw. Schreibvorgang wird durch die Steuerleitung `RD#` und `WR#` angezeigt. Das EthLink Datenbussystem verwendet von den genannten Leitungen einen Chip Select sowie die erwähnten Steuerleitungen. Weiterhin werden zwei Interrupts sowie zwei PIO-Pins genutzt.

Der SC13 verfügt über einen sogenannten Hardware-Watchdog. Hierbei handelt es sich um eine Timer-Funktion, die die Tasks auf dem Mikrocontroller überwacht. Eine Funktion zum Resetten des Timers muss periodisch aufgerufen werden. Dies kann vom RTOS durchgeführt werden oder durch den Aufruf einer bestimmten Methode der Bibliothek `CLIB.H`. Geschieht dies nicht, wird ein Neustart des Systems durchgeführt. Das Nichtaufrufen kann durch eine Endlosschleife in einem Task verursacht werden. Die Periodendauer, in der die entsprechende Funktion aufgerufen werden muss, beträgt laut [2] 838 ms.

Die zur Entwicklung des EthLink Datenbussystems verwendeten Komponenten sind in Abbildung 11 dargestellt. Die Darstellung wurde einer Abbildung in [2] nachempfunden und den projektspezifischen Anforderungen angepasst. So wurden nicht genutzte Komponenten zur Erhöhung der Übersichtlichkeit nicht übernommen.

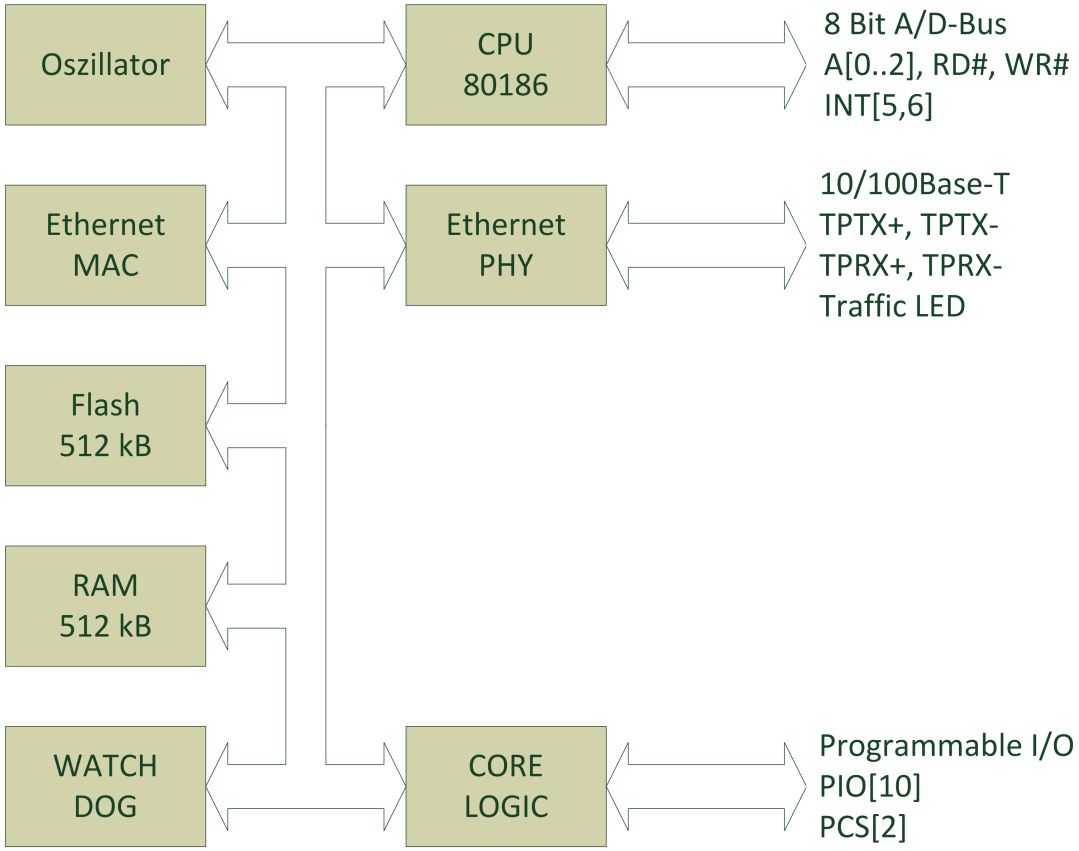

Abbildung 11: Verwendete Komponenten des BECK IPC SC13 nach [2]

3.4 Link-Adapter IMS C012

Der Link-Adapter IMS C012 dient zur Kommunikation zwischen Mikrocontrollern und Transputern. Der folgende Abschnitt basiert auf [14], dem Datenblatt des Link-Adapters. Das Bauteil ist in Abbildung 12 dargestellt.

Der IMS C012 konvertiert zwischen einem 8 bit A/D-Bus und einer auf dem T-Link-Protokoll basierenden Verbindung. T-Link wird, wie in Abschnitt 2.1.3 erwähnt, durch zwei Leitungen realisiert, `LinkIn` und `LinkOut`. Weiterhin wird ein externer Quarz benötigt, der an den Pin `ClockIn` angeschlossen wird. Ebenfalls zur Verfügung steht der `Reset`-Eingang, mit dem der Link-Adapter in den Ausgangszustand zurückversetzt werden kann.

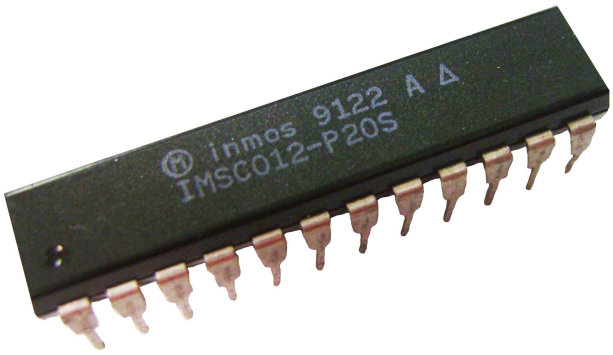

Abbildung 12: Link-Adapter IMS C012

Bei T-Link werden 10 Mbit/s und 20 Mbit/s unterstützt. Bei beiden Geschwindigkeiten handelt es sich um eine Vollduplex-Kommunikation.

Zwei Interrupts stehen am IMS C012 zur Verfügung, `InputInt` und `OutputInt`. `InputInt` wird ausgelöst, wenn vom Transputer gesendete Daten verfügbar sind. `OutputInt` zeigt an, dass der Link-Adapter bereit zum Empfangen von Daten auf dem A/D-Bus ist. Durch die Nutzung der Interrupts ist kein Polling nötig.

Der 8 bit A/D-Bus ist ein bidirektionaler Bus. Dieselben Pins werden also als Ein- und als Ausgang genutzt. Der Link-Adapter verfügt über den Pin `RnotW`, mit dem der A/D-Bus zwischen Ein- und Ausgang umgeschaltet werden kann. Ein `high`-Signal bewirkt die Schaltung auf einen Eingang, ein `low`-Pegel verursacht einen Ausgang.

Für die Kommunikation auf dem A/D-Bus verfügt der IMS C012 über drei Register:

- Data Register
- Input Status Register
- Output Status Register

Das Data Register dient der Übertragung der Nutzdaten. Der Mikrocontroller schreibt dort seine ausgehende Daten hinein, vom Transputer ausgehende Daten können hier gelesen werden. Input und Output Status Register liefern im Lesemodus Informationen über den Zustand, in dem sich der Link-Adapter befindet. Das LSB des Input Status Registers gibt an, ob vom Transputer gesendete Daten anliegen und ausgelesen werden können. Am LSB des Output Status Registers ist zu erkennen, ob der IMS C012 zum Empfangen von Daten auf dem A/D-Bus bereit ist. Eine `1` bedeutet hier, dass Daten verfügbar sind oder der Link-Adapter bereit ist. Die jeweils vorletzten Bits der Register betreffen die Interrupts. Hier steht eine `0` dafür, dass die Interrupts deaktiviert sind.

Die Steuerleitungen `RS0` und `RS1` bewirken die Auswahl der Register. Zusammen mit der Steuerleitung `RnotW` ergeben sich drei Steuerleitungen, die drei Bit repräsentieren. Tabelle 2 zeigt die Auswahl der Register und des Zugriffs-Modus:

RS1	RS2	RnotW	Register
0	0	1	Read Data
0	0	0	Invalid
0	1	1	Invalid
0	1	0	Write Data
1	0	1	Read Input Status
1	0	0	Write Input Status
1	1	1	Read Output Status
1	1	0	Write Output Status

Tabelle 2: Auswahl der Register und des Zugriffs-Modus für den IMS C012

Abbildung 13 stellt die Ein- und Ausgänge des Link-Adapters dar sowie die internen Blöcke.

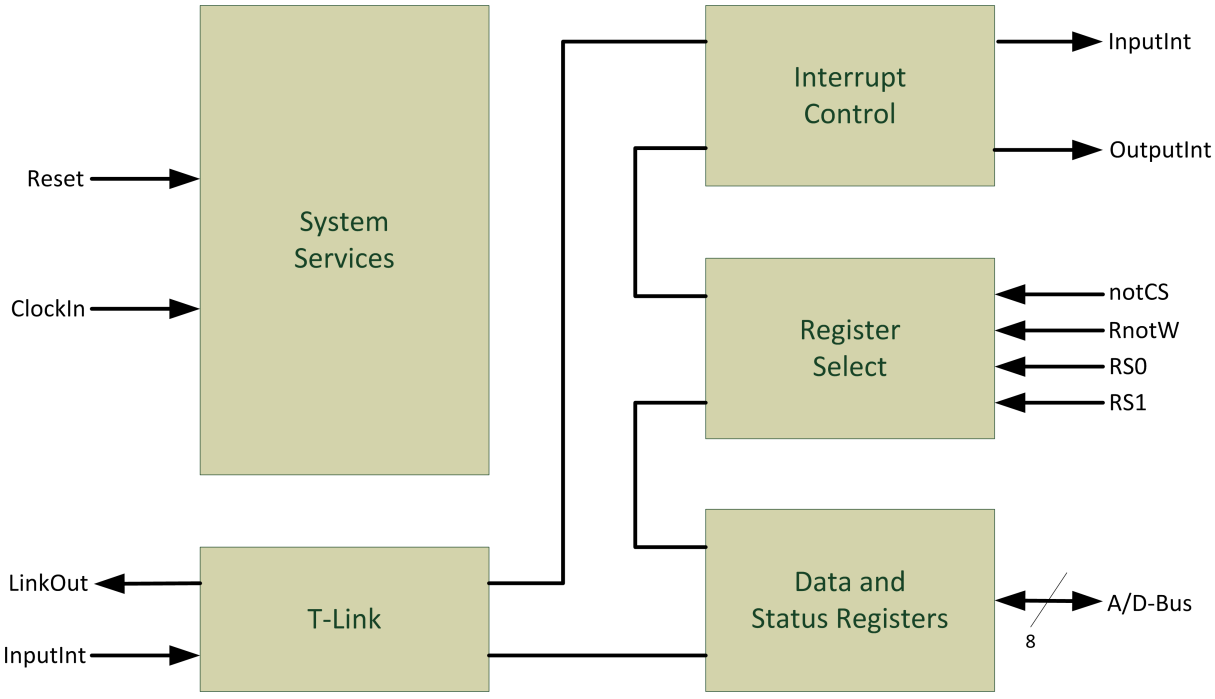

Abbildung 13: Architektur des IMS C012 nach [14]

3.5 Hardware-Testumgebung

Die Testumgebung wurde zusammengestellt, um ein von TEDAS II unabhängiges Entwickeln und Testen zu ermöglichen.

3.5.1 ParsyTec Multi Transputer Module

Das Multi Transputer Modul (Abk.: MTM)-2 von ParsyTec gehört der Megaframe-Serie an. Es ist mit zwei Transputern der Serie T800 bestückt.

Beide Transputer verfügen jeweils über 1 MB DRAM. Die T-Link-Verbindungen entsprechen dem RS-422-Standard, bei dem es sich um eine differentielle serielle Übertragung handelt. Die Verwendung dieses Standards ermöglicht Übertragungswege von einer Länge über 30 cm. Das MTM-2 verfügt über einen 96-Pin DIN Connector als externe Schnittstelle. Auf diesem werden sämtliche Links sowie Reset-Leitungen nach außen geführt.

Das MTM-2 von ParsyTec ist in Abbildung 14 dargestellt.

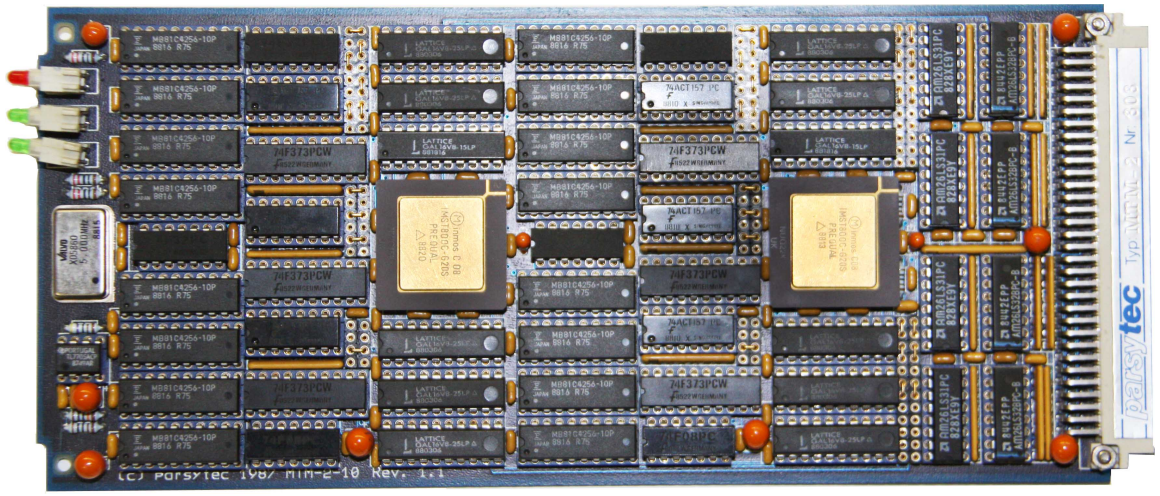

Abbildung 14: ParsyTec MTM-2

3.5.2 ParsyTec Megaframe-/IBM-Adapter

Der Megaframe/IBM-Adapter von ParsyTec dient der Ansteuerung von Modellen der Megaframe-Serie mit einem ISA- oder XT-Bus von IBM. In Zuge dieses Projektes wird ein MTM-2 angesteuert. Die Kommunikation mit dem MTM-2 erfolgt über einen 96-Pin DIN Connector.

Neben der Verbindung mit dem ISA- oder XT-Bus ermöglicht der Megaframe/IBM-Adapter auch

die Kommunikation zwischen externen Links und dem MTM-2. Die externen Links müssen bereits als RS-422-Signal vorliegen.

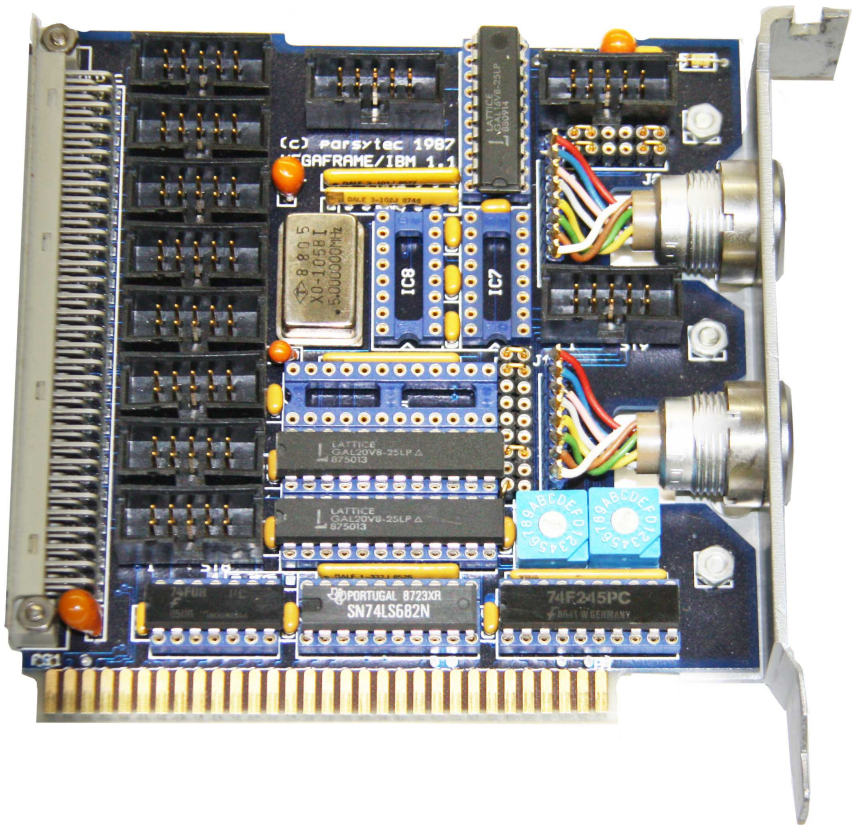

Abbildung 15: Parsytec Megaframe-/IBM-Adapter

4 Entwurf und Umsetzung

4.1 Planung

Der Bearbeitungszeitraum der Bachelorarbeit beträgt 12 Wochen beziehungsweise 70 Arbeitstage. Die zu erledigenden Aufgaben lassen sich in die Bereiche *Dokumentes*, *Theorie*, *Hardware*, *Software* und *Tests* einteilen. Die Summe der aufgeführten Arbeitstage übersteigt den Bearbeitungszeitraum von 70 Arbeitstagen, da parallel an verschiedenen Arbeitspaketen gearbeitet wird.

Der Bereich *Theorie* umfasst die Arbeitspakete, die sich mit der Erarbeitung der nötigen theoretischen Kenntnisse befassen. Mit dem zehntägigen Arbeitspaket *Analyse Transputer* startet das Projekt. Später erfolgt die Analyse des Befehlssatzes von Transputern.
Insgesamt sind für den Bereich *Theorie* 13 Arbeitstage vorgesehen. Aufgrund einer Unterbrechung in der Bearbeitung liegen zwischen Beginn und Ende der Arbeitspakete dieses Bereiches 46 Tage.

Die Erstellung des Dokumentes teilt sich auf in *Schreiben*, *Drucken* sowie *Abgabe* der Arbeit. In den Unterbereich *Schreiben* fällt neben der Verfassung des Textes auch die Korrektur von diesem. Die Abgabe der Bachelorarbeit stellt den Abschluss des Projektes dar.
Der Bereich *Dokument* beginnt in der dritten Woche der Bearbeitungszeit und endet am Abgabedatum der Bachelorarbeit. Insgesamt umfasst er 56 Tage.

Im Bereich *Hardware* sind alle zur Erstellung der Platine nötigen Arbeitsschritte gesammelt. Der Prozess der Platinenerstellung beginnt mit der Analyse der Anforderungen an die Hardware. Darauf folgt das Arbeitspaket *Entwurf Schaltplan*. Nachdem der Schaltplan entworfen wurde, wird dieser zunächst auf einer Lochrasterplatine umgesetzt und gegebenenfalls modifiziert. Bei einem erfolgreichen Test wird mit dem Erstellen des Platinenlayouts fortgefahren und die Platine bei einer externen Firma in Auftrag gegeben. Anschließend wird die Platine mit den elektrischen Bauteilen bestückt.
Für die Entwicklung und Fertigung der Platine sind insgesamt 40 Arbeitstage angesetzt. Die Arbeit an diesem Bereich beginnt, sobald das erste Arbeitspaket des Bereiches *Theorie* abgeschlossen ist.

In den Bereich *Software* fallen die Arbeitspakete für die Software des Mikrocontrollers und des User Interface. Nach einer anfänglichen Analyse der Anforderungen folgen jeweils Arbeitspakete zum Entwurf und zur Implementierung.
Für die Erstellung der Software sind 46 Arbeitstage eingeplant. Hiervon entfallen zehn Tage auf die Analyse, 23 auf die Software für den Mikrocontroller und zehn für das User Interface.

Dieser Bereich wird ebenfalls begonnen, sobald im Bereich *Theorie* die nötigen Grundlagen erarbeitet wurden.

Der Bereich *Tests* umfasst zunächst den Aufbau der Testumgebung. Weiterhin sind Tests angesetzt für die gefertigte Platine sowie die entwickelte Software.
Der Aufbau der Testumgebung ist mit drei Tagen veranschlagt. Die Tests der Platinen dauern zehn, die Abnahmetests der Programme zwei Tage. Somit entfallen auf diesen Bereich 15 Arbeitstage. Mit dem Aufbau der Testumgebung kann bereits früh begonnen werden. Die Tests können jedoch erst nach Fertigstellung von Hard- oder Software stattfinden. Dementsprechend streckt sich der Bereich *Tests* über ingesamt 47 Arbeitstage.

Die erwähnten Arbeitspakete der Bereiche sowie die Abhängigkeiten sind in Abbildung 16 dargestellt.

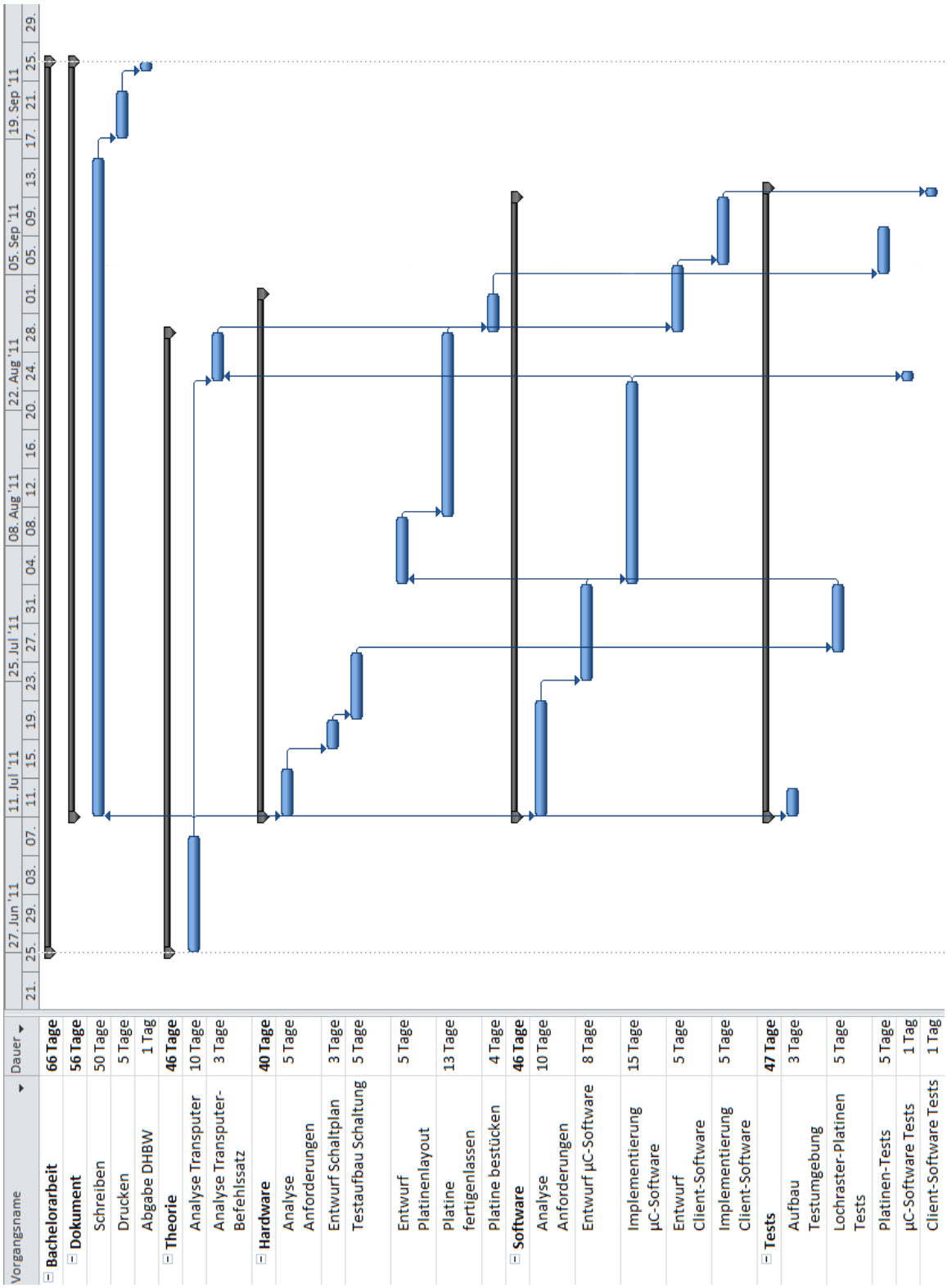

Abbildung 16: Gantt-Diagramm der Zeitplanung

4.2 Analyse der Software-Anforderungen

Die Anforderungen an die Software für den Mikrocontroller sind wie folgt in der Aufgabenstellung (siehe Abschnitt 1.7) angegeben:

> „Entwicklung einer transparenten Datenverbindung, welche die Daten eines Transputerlink vom BECK IPC auf ein User Interface und umgekehrt weiterleitet. Am Ende muss ein Befehl oder Datenwort, welches über Ethernet zum BECK IPC geschickt wird, von diesem so aufbereitet und in den Transputerlink geschickt werden, dass der Transputer es fehlerfrei versteht."

Dementsprechend hat die Software des Mikrocontrollers eine bidirektionale, transparente Datenverbindung zur Verfügung zu stellen. Eine Analyse der geplanten Kommunikation hat ergeben, dass diese stets vom Benutzer initiiert wird. Ein Verbindungsaufbau von der Messanlage TEDAS II findet nicht statt. Ein Sequenz-Diagramm des Ablaufs der Kommunikation ist in Abbildung 17 dargestellt.

Weiter heißt es in der Aufgabenstellung:

> „Analyse der Befehlsstruktur des Transputerlink und Nachbildung dieser Befehle im User Interface, um eine Kommunikation zwischen PC und Transputer herzustellen."

Die Analyse der geplanten Kommunikation hat ebenfalls ergeben, dass stets der Boot- und Programmcode oder der Befehl zur Ausführung eines Nullabgleichs übertragen wird. Eine Veränderung dieser Codes ist nicht geplant. Folglich genügt es, die Übertragung der Codes über ein User Interface starten zu können. Eine weitere Konfigurationen durch den Nutzer ist nicht erforderlich.

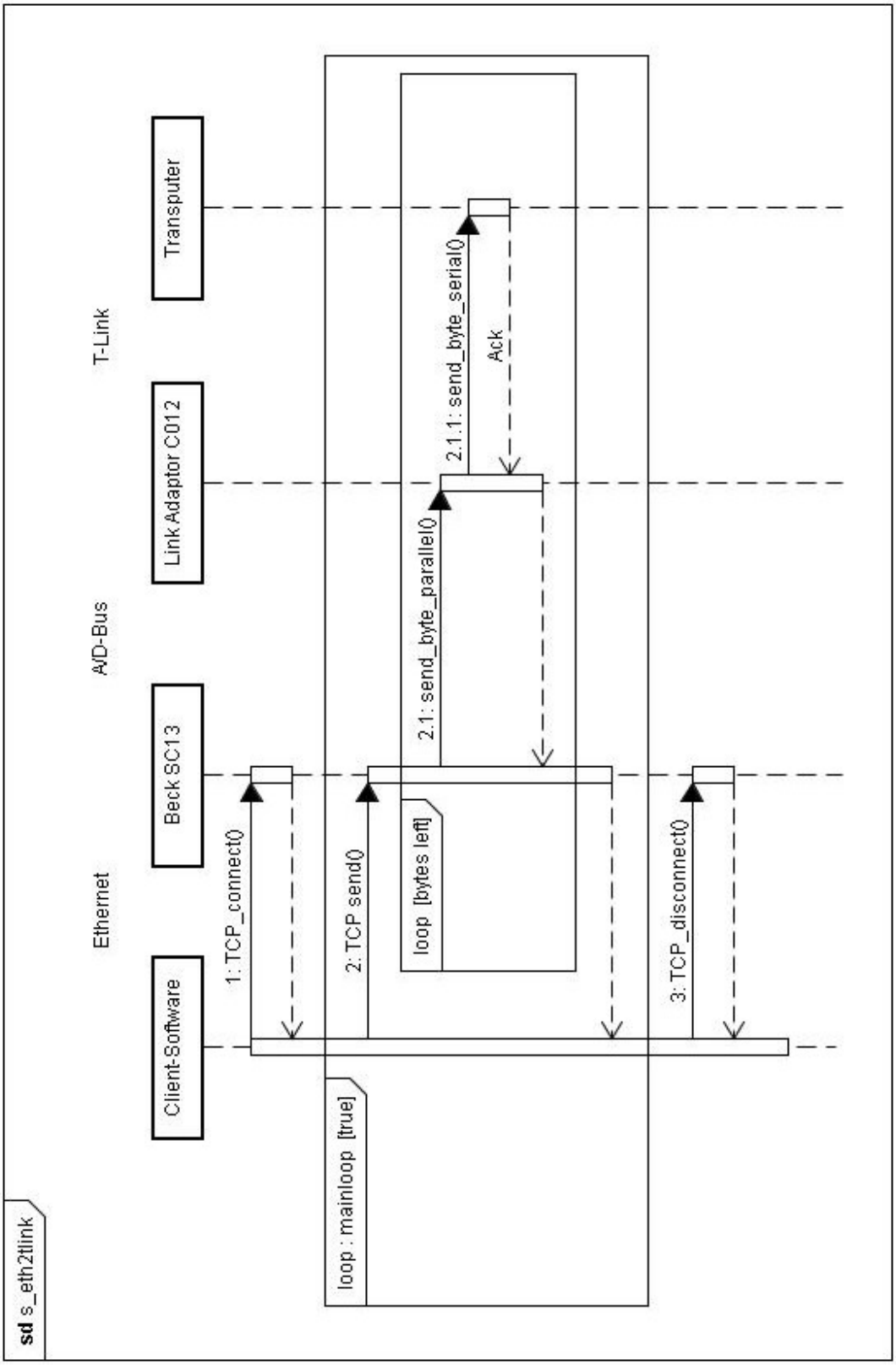

Abbildung 17: Sequenz-Diagramm der Kommunikation

4.3 Vorbereitung der Umsetzung

4.3.1 Aufbau der Testumgebung

Die Testumgebung dient zu einem von TEDAS II unabhängigem Testen, da die Messanlage nicht durchgehend verfügbar ist. Sie besteht aus dem mit zwei Transputern bestückten ParsyTec MTM-2 und dem ParsyTec Megaframe-/IBM-Adapter. Eine genauere Beschreibung zu diesen ist Abschnitt 3.5 zu entnehmen.

Das EthLink Datenbussystem wird über einen der 5x2-Stecker des Megaframe-/IBM-Adapters angeschlossen.

Der Megaframe-/IBM-Adapter ist für den Gebrauch in einem ISA- oder XT-Bus konzipiert und bezieht seine Betriebsspannung aus diesem Anschluss. Da weder ISA- noch XT-Bus in modernen Computern enthalten sind, wird das Mainboard eines im Institut für Flugsystemtechnik vorhandenen PCs verwendet, das über einen solchen Bus verfügt. Die aufgebaute Testumgebung ist in Abbildung 18 dargestellt.

Abbildung 18: Testumgebung aus MTM-2, Megaframe-/IBM-Adapter und Mainboard

4.3.2 Erstellen der Testfälle

Der Mathematiker Edsger W. Dijkstra hat in [5] folgende Aussage über Software-Tests getätigt:

> „Program testing can be used to show the presence of bugs, but never to show their absence!“ [5]

Demzufolge ist es also möglich, mit Tests die Anwesenheit von Fehlern in Software festzustellen, aber niemals deren Abwesenheit. Da mit einem breiteren Spektrum der Tests auch die Wahrscheinlichkeit zunimmt, alle Fehler zu finden, muss eine möglichst breite Testabdeckung das Ziel sein.

Der Ansatz einer möglichst vollständigen Test-Abdeckung führt in diesem Projekt zu einer Aufteilung in die Kategorien *Software*, *Hardware* sowie *System*. In den Bereich der Software-Tests fällt die Überprüfung der Funktionalität. Die Tests der Hardware beziehen sich auf die Überprüfung der elektrischen Eigenschaften. So müssen die Leiterbahnen auf der Platine ebenso getestet werden wie die Stabilität der Spannung. Die reine Funktionalität des Ethernet-Anschlusses beispielsweise fällt ebenfalls in diesen Bereich, da die Ansteuerung bereits im RTOS des Mikrocontrollers integriert ist. Lediglich die Hardware der Ethernet-Schnittstelle wird in diesem Projekt erstellt. Da für viele Software-Tests ebenfalls die Hardware korrekt arbeiten muss, sind die Hardware-Tests zuerst durchzuführen.
Abschließend werden System-Tests durchgeführt. Diese überprüfen das Zusammenspiel von Hard- und Software und die Stabilität des Systems. Bei einem erfolgreichen Ergebnis dieser Tests kann mit einer ausreichend hohen Wahrscheinlichkeit von einem funktionierenden Gesamt-System ausgegangen werden.
Die erstellten Testfälle sind zusammen mit einer kurzen Beschreibung in Tabelle 3 aufgeführt.

Testfall	**Kategorie**	**Beschreibung**
Leiterbahnen	Hardware	Überprüfung der Leiterbahnen der Platine
Versorgungs-spannung	Hardware	Messung der anliegenden Versorgungsspannung unter Last
Ethernet	Hardware	Test der Ethernetverbindung
TCP Senden bzw. Empfangen	Software	Isolierte Tests des Sendens bzw. Empfangens von Daten per TCP
TCP Gesamt	Software	Test der bidirektionalen TCP-Verbindung mittels Daten-Request- und -Response-Verfahren
Einlesen Byte-Codes	Software	Einlesen von Boot- und Programm-Codes
Website	Software	Anzeigen der Website und Erkennung von Nutzereingaben
Ansteuerung A/D-Bus	Software	Schreiben und Lesen auf dem 8 bit A/D-Bus
Transputer-Kommunikation	System	Test der bidirektionalen Kommunikation zwischen dem EthLink Datenbussystem und einem Transputer
Stabilität	System	Einwöchiger Dauertest des EthLink Datenbussystems

Tabelle 3: Erstellte Testfälle

Der Test *Transputer-Kommunikation* wird an dieser Stelle exemplarisch genauer erläutert. Die bidirektionale Kommunikation wird getestet, indem durch eine Übertragung vom EthLink Datenbussystem an den Transputer eine Antwort ausgelöst wird. Um ein simples Echo der übertragenen Daten auszuschließen, ist eine Verarbeitung und deterministische Veränderung der empfangenen Daten durch den Transputer nötig.
Für die zu übertragenden Daten wurden positive Zahlen vom Typ `Integer` gewählt. Die vom Transputer durchzuführende Modifikation ist die Quadrierung der empfangenen Zahl. Die Quadrierung einer natürlichen Zahl ist als Test geeignet, da diese Funktion injektiv ist, jedes Element der Ergebnismenge kann also nur durch genau einen Eingabeparameter erzeugt werden. Weiterhin liefert eine Überprüfung auf Gleichheit ein zuverlässiges Ergebnis, da Zahlen vom Typ `Integer` kein Maschinenepsilon aufweisen.
Die Prozedur, die für diesen Test auf dem Transputer ausgeführt wird, ist in Quellcode 5 angegeben.

```
1  PROC Square()
2     INT number:
3     INT result:
4     SEQ
5        link1 ? number
6        result := number * number
7        link1 ! result
```

Quellcode 5: Testfall Transputer-Kommunikation

Zunächst werden die Variablen `number` und `result` deklariert (Zeile 2-3). Anschließend wird die Test-Zahl von dem mit `link1` bezeichneten Channel empfangen und in der Variablen `number` gespeichert (Zeile 5). Das Produkt dieser Zahl mit sich selbst wird der Variablen `result` zugewiesen (Zeile 6) und an den Channel `link1` ausgegeben (Zeile 7).

4.4 Schaltungsentwurf

Sämtliche Bauelemente sollen auf einer Platine vereint werden. Hierzu gehören neben dem Mikrocontroller BECK IPC SC13 und dem Link-Adapter IMS C012 auch die Spannungsversorgung, die Ethernet-Anbindung sowie die für RS-422 benötigten differentiellen Treiber.

Die Spannungsversorgung muss konstant 5 V betragen, um den Anforderungen von Mikrocontroller und Link-Adapter zu genügen. Zunächst verringert ein Transformator die Spannung von 230 V auf etwa 9 V. Es folgt ein Brückengleichrichter, der die Wechselspannung in Gleichspannung wandelt, mit Kapazitäten zur Spannungsglättung. Die Verringerung der Versorgungs-

spannung auf 5 V wird von einem Spannungsregler durchgeführt. Parallel zu diesem sind eine weitere Kapazität sowie eine Zener-Diode verbaut, um die Spannung zu stabilisieren. In Abbildung 19 ist der Ausschnitt des erstellten Schaltplans abgebildet, der die Spannungsversorgung enthält.

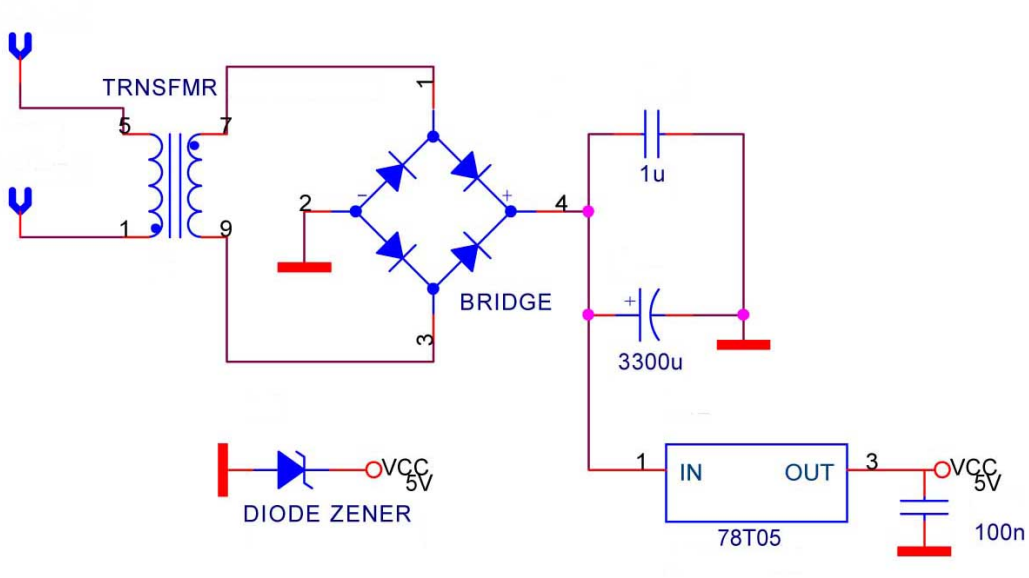

Abbildung 19: Schaltplan der Spannungsversorgung

Damit eine Kommunikation zustande kommt und die elektrischen Bauteile nicht beschädigt werden, müssen sich die Spannungspegel auf dem A/D-Bus in den vorgegebenen Bereichen befinden. Die an den Ausgängen erzeugten Spannungslevel eines Bauteils müssen innerhalb der Grenzen für die Eingänge des anderen Bauteils liegen. Die in den Hardware-Handbüchern des BECK SC13 [2] und IMS C012 [14] für eine Versorgungsspannung von 5 V angegebenen Werte sind in Tabelle 4 gegenübergestellt.

	BECK SC13		**IMS C012**	
	Min	Max	Min	Max
Spannung Output `low`	-	1.5 V	-	0.4 V
Spannung Output `high`	3.5 V	-	4 V	-
Spannung Input `low`	-	1.5 V	-0.5 V	0.8 V
Spannung Input `high`	3.5 V	-	2 V	5.5 V

Tabelle 4: Elektrische Spezifikationen von BECK SC13 und IMS C012 nach [2] und [14]

Wie zu erkennen ist, sind der Mikrocontroller und der Link-Adapter nicht kompatibel. Ein vom BECK SC13 erzeugtes `low`-Signal weist eine Spannung von maximal 1.5 V auf. Vom IMS C012 werden `low`-Signale jedoch nur im Bereich von -0.5 V bis 0.8 V erkannt. Um diese Differenz zu

beheben, müssen weitere elektrische Bauteile zur Anpassung der Spannung zwischen diesen installiert werden.

Die letzten drei Bits der vom BECK SC13 angesprochenen Adresse liegen, wie bereits in Abschnitt 3.3 erwähnt, auf separaten Adressleitungen an. Diese Steuerleitungen `A0` bis `A2` des Mikrocontrollers werden mit den Steuerleitungen `RS0`, `RS1` und `RnotW` des Link-Adapters verbunden. Die Paare sind `A0` ↔ `RS1`, `A1` ↔ `RS0` sowie `A2` ↔ `RnotW`. Auf diese Weise werden mit der Adresse das Register und der Zugriffsmodus ausgewählt. So bewirkt eine Adresse, die auf `07` endet, dass das Output Status Register im Lesemodus angesprochen wird, während `02` das Schreiben in das Data Register vorbereitet.
Um eine fehlerfreie Kommunikation zu gewährleisten, ist die Einhaltung der Timing-Vorschriften beider ICs Voraussetzung. Eigenschaften des A/D-Bus des Mikrocontrollers sind nach [3] wie folgt:

- Schreiben auf den A/D-Bus: Die Daten auf dem A/D-Bus sind gültig, während der Chip-Select aktiv ist
- Lesen vom A/D-Bus: Der A/D-Bus wird gelesen, während der ChipSelect aktiv ist
- Die Adressleitung `A0` bis `A2` sowie Lesesignal `RD#` und Schreibsignal `WR#` sind gültig, während der ChipSelect aktiv ist.
- Wird weder vom A/D-Bus gelesen noch auf diesen geschrieben, wird dieser für interne Speicherzugriffe verwendet

Ein Auszug aus den Timing-Bedingungen des Link-Adapters ist in Abbildung 20 dargestellt. Die kompletten Timing-Vorschriften können in [14] nachgeschlagen werden.

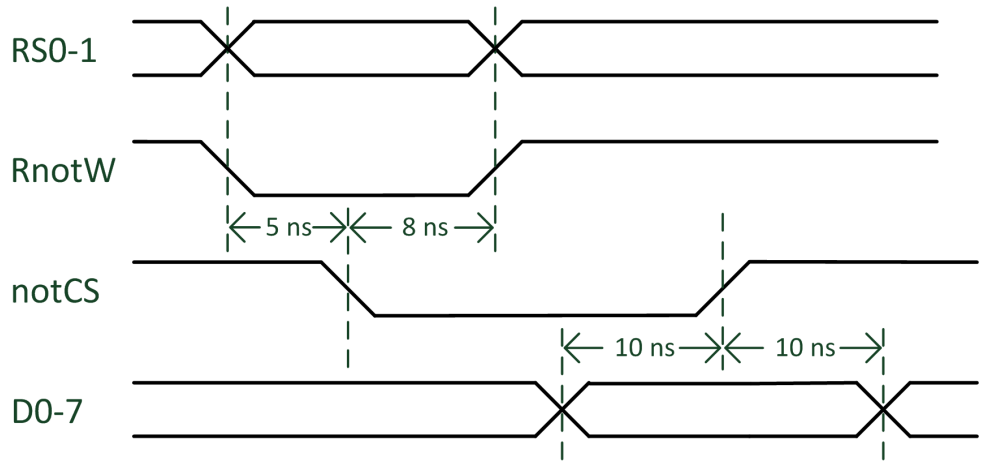

Abbildung 20: Timing-Vorschriften beim Schreiben auf Link-Adapter IMS C012 nach [14]

Abbildung 20 ist zu entnehmen, dass die Steuerleitungen `RS0`, `RS1` und `RnotW` mindestens 5 ns vor der Aktivierung des ChipSelects und 8 ns nach dieser gültig sein müssen. Die Steuerleitungen dienen, wie in Abschnitt 3.4 erwähnt, der Auswahl des Registers und des Zugriffsmodus. Weiterhin müssen die Daten auf dem A/D-Bus mindestens 10 ns vor und 10 ns nach der Aufhebung des ChipSelects valide sein.
Ein Vergleich der Timing-Bedingungen von Mikrocontroller und Link-Adapter ergibt, dass diese nicht kompatibel sind. Der BECK SC13 nutzt den A/D-Bus nach Aufhebung des ChipSelects für interne Speicherzugriffe, wohingegen für den IMS C012 die Daten für weitere 10 ns gehalten werden müssen. Um diese Unstimmigkeit im Timing zu beheben, ist der Einsatz einer Reihe von elektronischen Bauteilen erforderlich.
Zunächst einmal gilt es, den bidirektionalen A/D-Bus des Mikrocontrollers vom Link-Adapter abzukoppeln, während kein Lese- oder Schreibvorgang auf dem A/D-Bus durchgeführt wird. Auf diese Weise werden die internen Speicherzugriffe des Mikrocontrollers nicht vom IMS C012 beeinflusst. Hierfür wird ein Tri-State Bus Transceiver verwendet, der neben den Zuständen `1` und `0` zusätzlich einem hochohmigen Zustand annehmen kann. Wird kein Lese- oder Schreibvorgang auf dem A/D-Bus durchgeführt, schaltet der Bus Transceiver in den hochohmigen Zustand und trennt somit Mikrocontroller und Link-Adapter voneinander.
Beim Schreibvorgang des Mikrocontrollers auf dem A/D-Bus tritt das erwähnte Timing Problem auf. Um dieses zu beheben, müssen die Daten auf dem A/D-Bus länger gehalten werden. Dies wird duch ein flankengesteuertes FlipFlop bewerkstelligt. Es triggert, wenn sowohl ChipSelect als auch das Schreib-Signal des Mikrocontrollers aktiv sind.
Um Lese- und Schreibvorgang voneinander zu isolieren, wird zusätzlich jeweils ein Tri-State Buffer verwendet.

Die eingesetzten elektronischen Bauteile sind schematisch in Abbildung 21 dargestellt. Abbildung 22 enthält den kompletten Schaltplan der EthLink-Platine.

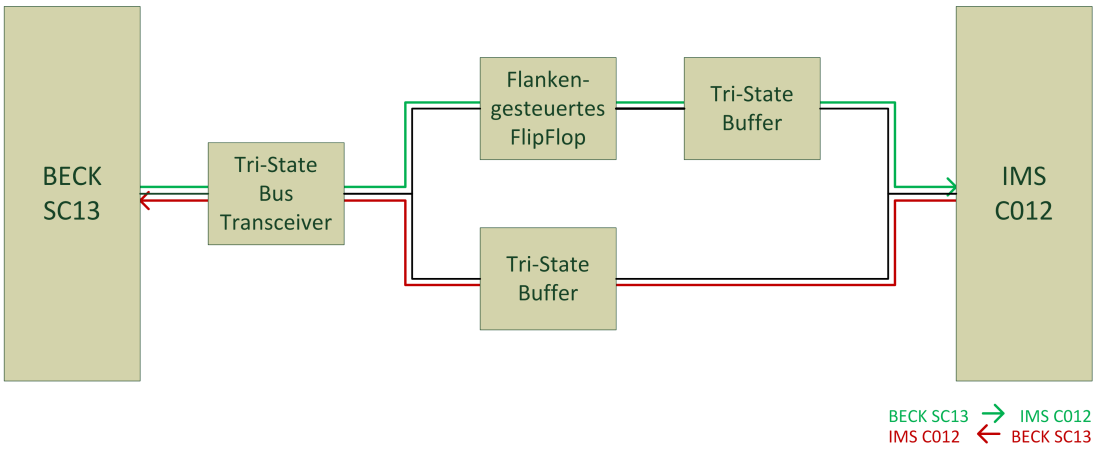

Abbildung 21: An der Kommunikation zwischen BECK SC13 und IMS C012 beteiligte ICs

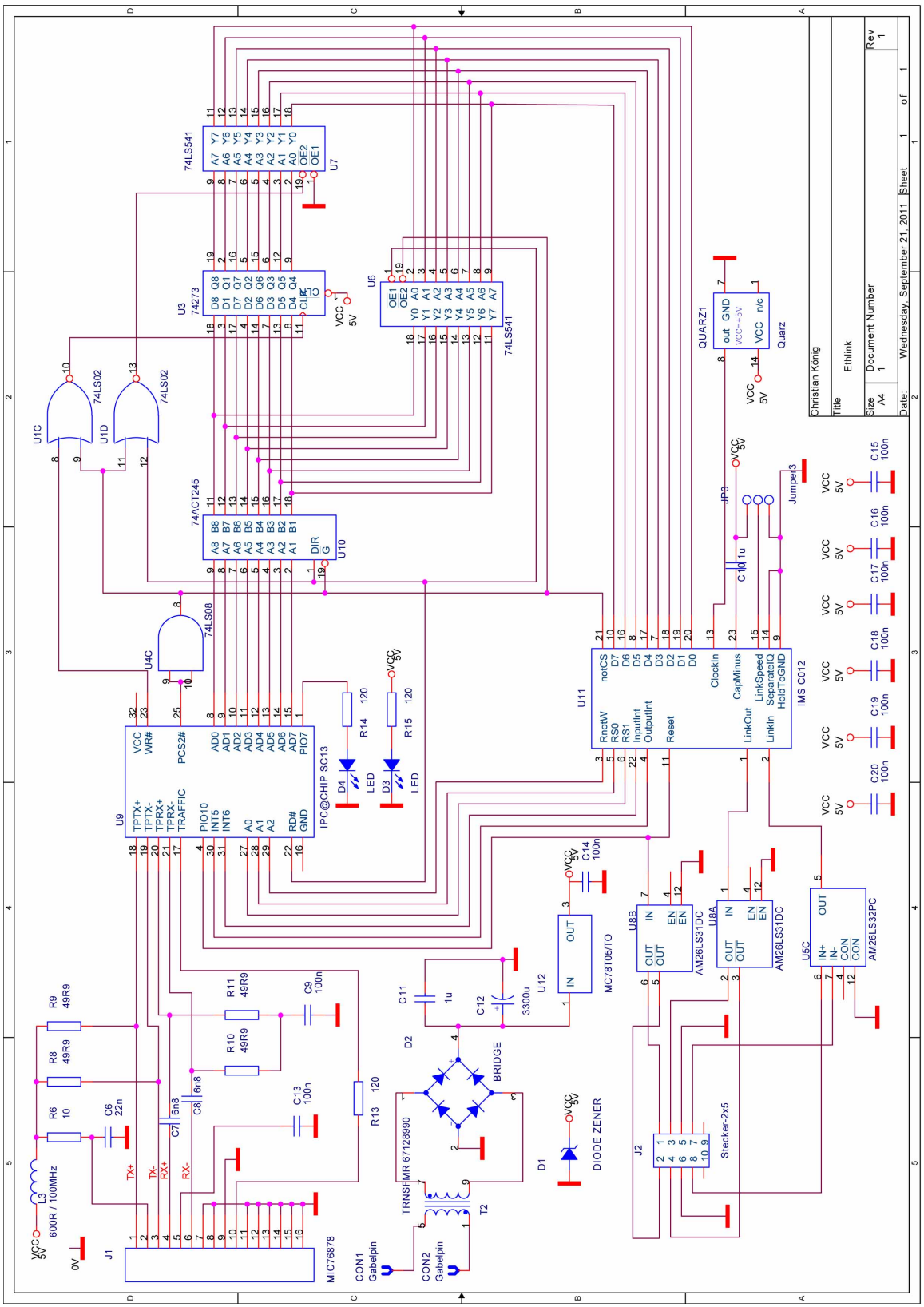

Abbildung 22: Schaltplan der EthLink-Platine

4.5 Programmentwurf

Das User Interface wird als Website bereitgestellt, die auf dem Webserver des Mikrocontrollers läuft. Auf diese Weise stehen alle geforderten Funktionalitäten (siehe Abschnitt 4.2) zur Verfügung. Weiterhin ist der Zugriff von jedem PC im Netzwerk des Rotorversuchsstands möglich, eine Installation von Software ist nicht mehr erforderlich.

Die bidirektionale, transparente Datenverbindung wird durch ein Anwendungsprogramm auf dem Mikrocontroller realisiert. Sie soll stets für einen Zugriff durch den Nutzer verfügbar sein, weswegen die Software nach der Initialisierung eine Endlos-Schleife durchläuft. Der Programmablaufplan (Abk.: PAP) der gesamten Software ist in Abbildung 23 grob dargestellt.

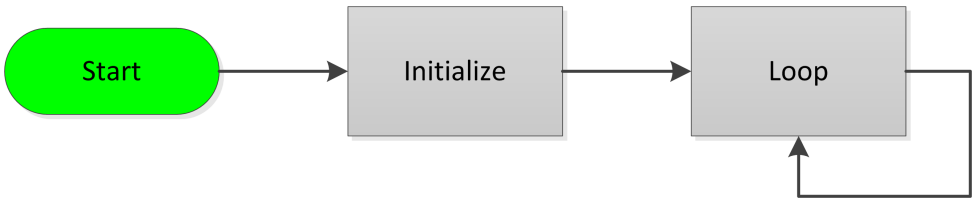

Abbildung 23: Gesamt-PAP der Mikrocontroller-Software

In der Operation `Initialize` werden sowohl die bidirektionale, transparente Datenverbindung als auch der Webserver initialisiert. In der Operation `Loop` befindet sich die Funktionalität der Datenverbindung. Weitere Informationen zum User Interface sind in Abschnitt 4.5.2 aufgeführt.

4.5.1 Programmentwurf der bidirektionalen Datenverbindung

Das Programm der bidirektionalen Datenverbindung unterteilt sich in die Bereiche *Initialisierung* und *Funktionalität*. Die Initialisierung der bidirektionalen Datenverbindung umfasst folgende Punkte:

- Aktivierung des 8 bit A/D-Bus
- Modi der PIOs setzen
- Aktivierung des TCP-Servers. Dies umfasst das Öffnen des Sockets (`opensocket()`), das Binden von diesem an einen Port (`bind()`) sowie das Lauschen auf diesem auf eingehende Verbindungen (`listen()`). Weiterhin wird der Socket in den non-blocking-Mode versetzt.
- Aktivierung der Interrupts für die A/D-Bus-Kommunikation

Der funktionale Abschnitt des Programms läuft, wie in Abbildung 23 dargestellt ist, in einer nicht-terminierenden Schleife. Ein Durchlauf dieser Schleife ist in Abbildung 24 als PAP dargestellt.

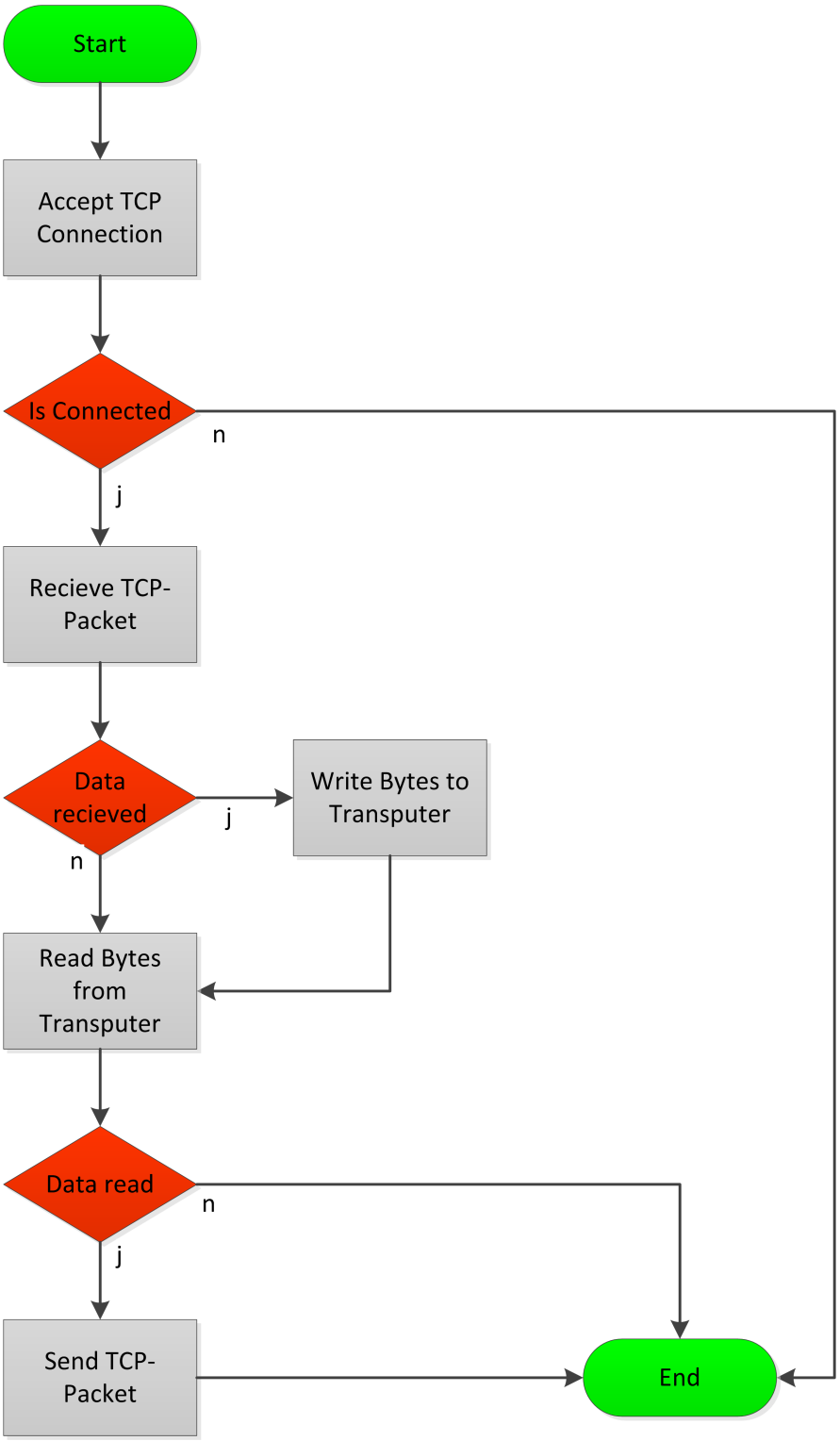

Abbildung 24: PAP des Main-Loops

Zunächst wird der erste Client in der TCP-Warteschlange akzeptiert. Da sich der Socket, wie erwähnt, im non-blocking-Mode befindet, terminiert diese Funktion unmittelbar. Abhängig vom Rückgabewert kann bestimmt werden, ob das Akzeptieren der Verbindung fehlgeschlagen ist, da sich kein Client in der Queue befindet. Im Erfolgsfall wird auf diese Weise die Verbindung mit dem Client hergestellt. Ist kein Client verfügbar, wird der Durchlauf der Schleife beendet. Eine Überprüfung, ob Datenpakete auf dem Link-Adapter verfügbar sind, ist nicht notwendig, da eine Session, wie in Abschnitt 4.2 erwähnt, stets vom Nutzer initiiert wird.
Es folgt der Versuch, zum Empfang bereite TCP-Pakete abzurufen. Auch hier returned die Funktion unmittelbar und es kann am Rückgabewert erkannt werden, ob Daten empfangen wurden. Wenn dem so ist, werden die Daten an den Link-Adapter und somit an den Transputer gesendet.
Unabhängig davon, ob Daten empfangen wurden, werden nun vom Link-Adapter ankommende Daten gelesen, die im Erfolgsfall an den TCP-Client gesendet werden. Anschließend ist der Schleifendurchlauf beendet.

Die Operation `Write Bytes to Transputer` ist im PAP der Hauptschleife aus Abbildung 24 enthalten. Ebenso wird die Operation `Read Bytes from Transputer` aufgerufen. Diese werden nun im Folgenden genauer erläutert.
Ein PAP der Operation `Write Bytes to Transputer` ist detailliert in Abbildung 25 dargestellt.

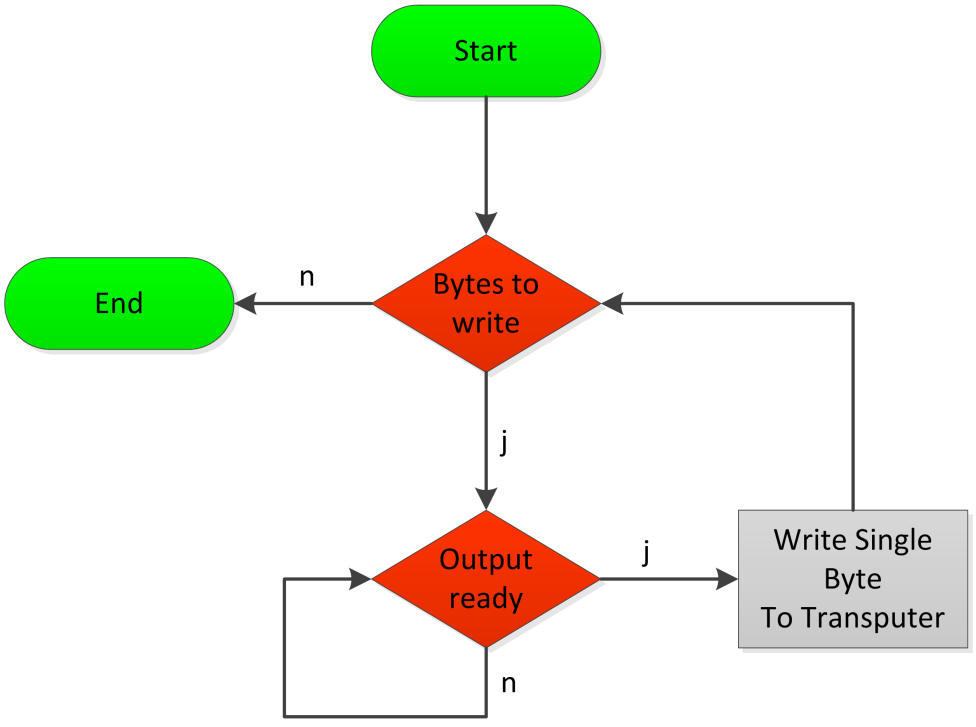

Abbildung 25: PAP der Operation „Write Bytes To Transputer“

Es werden Daten an den Link-Adapter und damit an den Transputer gesendet. Ein Byte kann allerdings nur gesendet werden, wenn der IMS C012 seine Bereitschaft signalisiert hat. Dieser Vorgang wird solange wiederholt, bis alle zu sendenden Daten an den Link-Adapter übertragen sind.
Die Operation `Read Bytes from Transputer`, deren PAP in Abbildung 26 dargestellt ist, überprüft ebenfalls den Status des Link-Adapters. Solange dieser signalisiert, dass weitere Daten verfügbar sind, werden diese ausgelesen. Sind alle zur Verfügung stehenden Daten gelesen, terminiert diese Operation.

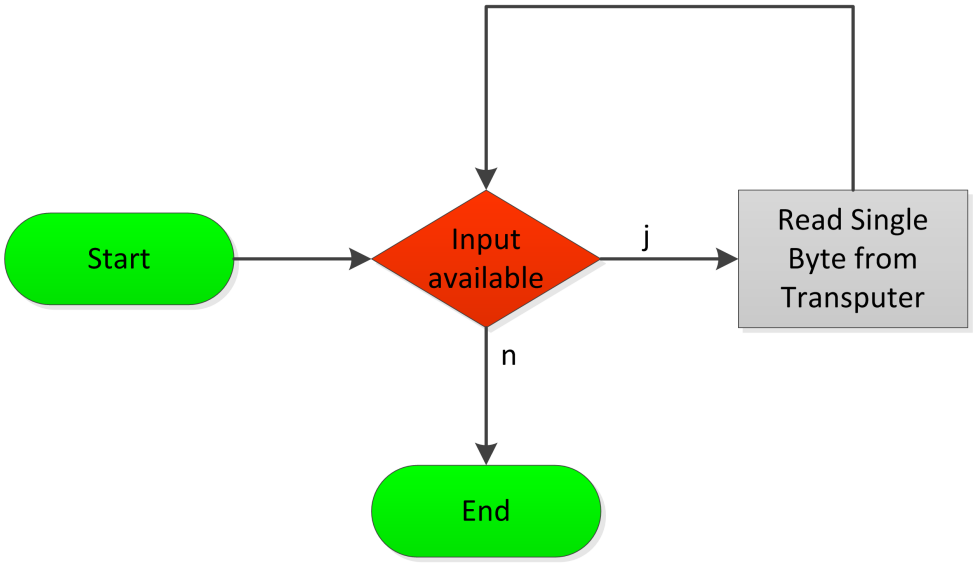

Abbildung 26: PAP der Operation „Read Bytes from Transputer“

4.5.2 Programmentwurf User Interface

Das User Interface benötigt zwei Schaltflächen: `REBOOT` und `ZERO ADJUSTMENT`. Eine Betätigung der Schaltfläche `REBOOT` führt zu einem Neustart der Transputer in der TEDAS-Messanlage, mit `ZERO ADJUSTMENT` wird ein Nullabgleich der Sensoren durchgeführt.

Eine komplexe Funktionalität muss nicht in das User Interface integriert werden, da Funtkionen der bidirektionalen Datenverbindung genutzt werden können. So muss beispielsweise die Transputer-Kommunikation nicht erneut implementiert werden. Bei der Aktivierung einer der beiden Schaltflächen wird der jeweilige Byte-Code der Kommandos eingelesen und an den Transputer gesendet. Ein PAP dieses Vorgangs ist in Abbildung 27 dargestellt.

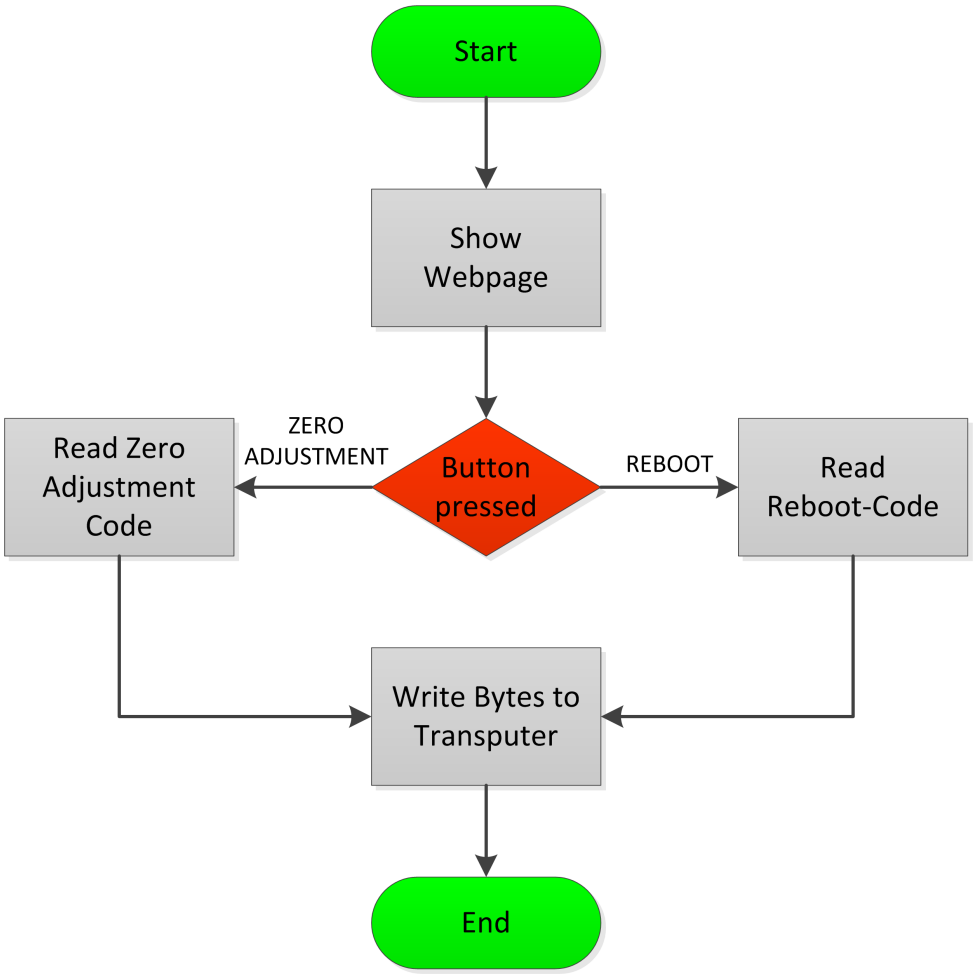

Abbildung 27: PAP der Webpage

4.6 Implementierung

4.6.1 Implementierung der transparenten Datenverbindung

Die transparente Datenverbindung ist vollständig in der Software für den Mikrocontroller implementiert. Die in Abschnitt 4.5.1 beschriebenen Programmablaufpläne wurden umgesetzt.

Die Kommunikation über das EthLink Datenbussystem lässt sich in zwei Abschnitte unterteilen:

$$\text{Client-PC} \longleftrightarrow \text{EthLink}$$
$$\text{EthLink} \longleftrightarrow \text{Transputer}$$

Das Empfangen von Daten verläuft in den Abschnitten unterschiedlich. Im Abschnitt zwischen dem Client-PC und EthLink wird Polling angewendet. In jedem Durchlauf wird auf TCP-Pakete abgefragt und überprüft, ob Daten empfangen wurden. Im Abschnitt zwischen EthLink und Transputer hingegen wird durch einen Interrupt des Link-Adapters angezeigt, wenn neue Daten verfügbar sind.
Der Mikrocontroller BECK SC13 verfügt über einen Hardware-Watchdog (siehe Abschnitt 3.3), der in der Standardkonfiguration durch das RTOS refreshed wird. Beim Senden eines umfangreichen Datensatzes an den Link-Adapter kommt es zu einem exzessiven Auftreten von Interrupts, was das Zurücksetzen des Watchdog-Timers verzögert. Wird der Aufruf der Watchdog-Funktion so stark verzögert, dass das maximale Intervall des Watchdogs von 838 ms überschritten wird, kommt es zu einem Reboot durch den Watchdog. Um unerwünschte Reboots zu vermeiden und ein stabiles Arbeiten des EthLink Datenbussystems zu gewährleisten, wird der Modus des Watchdogs gewechselt. Statt eines Refreshes durch das RTOS wird die entsprechende Funktion von der erstellten Software selbst aufgerufen. Die Stellen im Quellcode, an denen der Aufruf der Funktion platziert ist, wurden durch eine Analyse der maximalen Funktions-Laufzeiten ermittelt, sodass die Einhaltung des Intervalls stets gewährleistet ist.
Der Aufruf der Reset-Funktion des Watchdogs ist exemplarisch in Quellcode 6 dargestellt. Es handelt sich um die auf das wesentliche reduzierte Funktion zum Senden eines Bytes an den Link-Adapter C012 und damit an den Transputer.

```
1  int write_byte_to_c012( int port , unsigned char value )
2  {
3     //(...)
4     outportb( port , value );
5     hal_refresh_watchdog();
6
7     return 0;
8  }
```

Quellcode 6: Aufruf der Funktion zum Zurücksetzen des Watchdog-Timers

Es werden die Adresse des Registers im Link-Adapter sowie das zu sendende Zeichen an die Funktion übergeben (Zeile 1). Anschließend wird das Zeichen per A/D-Bus übertragen (Zeile 4). Die Funktion `outportb(int,char)` dient zum Senden eines einzelnen Bytes an den A/D-Bus. Bevor die Funktion terminiert (Zeile 7), wird der Watchdog-Timers resettet (Zeile 5). Hierfür wird die Funktion `hal_refresh_watchdog()` aus der Bibliothek `CLIB.H` (siehe Abschnitt 2.3) des Mikrocontrollerherstellers verwendet.

4.6.2 Implementierung des User Interfaces

Das User Interface wird, wie erwähnt, als Website umgesetzt. Die eingesetzten Techniken sind Hypertext Markup Language (Abk.: HTML) und Common Gateway Interface (Abk.: CGI).

Der Datenaustausch zwischen dem Webserver und der Software der transparenten Datenverbindung erfolgt mittels CGI. In der CGI-Funktion wird ausgewertet, ob und wenn ja welcher Button aktiviert wurde. Anschließend wird der entsprechende Byte-Code eingelesen und an den Transputer gesendet (siehe Abbildung 27 in Abschnitt 4.5.2). Weiterhin werden auf der Website Statusmeldungen des EthLink Datenbussystems angezeigt.

Die Website wird in HTML erstellt. Um die Ladezeiten bei einem Neuaufbau der Seite zu verkürzen, wird der interaktive Teil der Website in einem eingebetteten Frame, einem IFrame, dargestellt. Auf diese Weise ist ein erneutes Laden der Bilder nicht nötig.
Wie bereits in 4.5.2 erwähnt, werden zwei Schaltflächen im User Interface benötigt. Die erstellte Website ist in Abbildung 28 dargestellt.

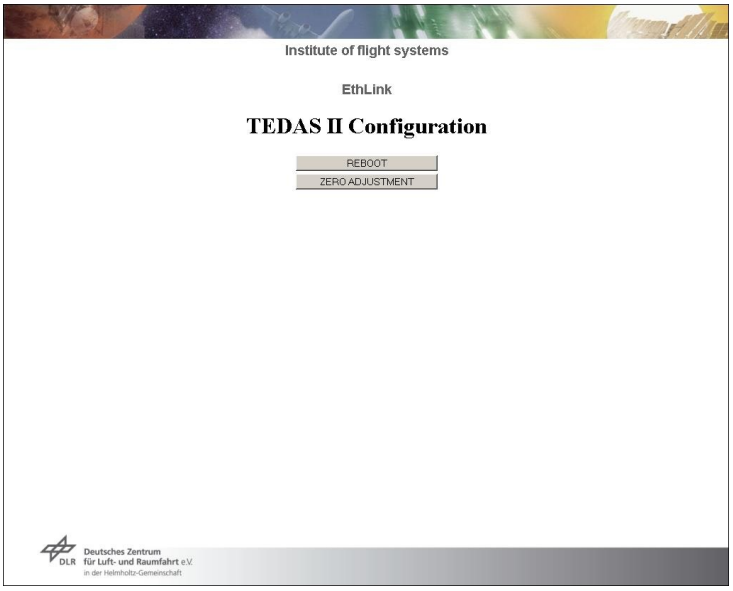

Abbildung 28: User Interface

5 Tests und Systemintegration

5.1 Hardware-Tests

Die Testfälle im Bereich der Hardware sind in Tabelle 5 aufgeführt. Ebenfalls in dieser Tabelle enthalten sind die Ergebnisse der Tests.

Testfall	Beschreibung	Ergebnis
Leiterbahnen	Überprüfung der Leiterbahnen der Platine	Bestanden
Versorgungs-spannung	Messung der anliegenden Versorgungsspannung unter Last	Bestanden nach Modifikation
Ethernet	Test der Ethernetverbindung	Bestanden

Tabelle 5: Ergebnis der Hardware-Tests

Sämtliche Testfälle wurden bestanden, allerdings war für das Bestehen des Tests *Versorgungsspannung* eine Modifikation erforderlich. Mit dem zunächst verwendeten Transformator wurde die Spannung bereits bei einer Leistungsentnahme von ca. 75 % des im Datenblatt angegebenen Wertes instabil. Nach dem Austausch gegen einen leistungsstärkeren Transformator konnte auch dieser Test erfolgreich abgeschlossen werden.

5.2 Software-Tests

Die in Abschnitt 4.3.2 beschriebenen Testfälle im Bereich Software sind sämtlich erfüllt. Tabelle 6 enthält eine Übersicht dieser Tests.

Testfall	Beschreibung	Ergebnis
TCP Senden bzw. Empfangen	Isolierte Tests des Sendens bzw. Empfangens von Daten per TCP	Bestanden
TCP Gesamt	Test der bidirektionalen TCP-Verbindung mittels Daten-Request- und -Response-Verfahren	Bestanden
Einlesen Byte-Codes	Einlesen der Byte-Codes der Boot- und Nullabgleich-Kommandos	Bestanden
Website	Anzeigen der Website und Erkennung von Nutzereingaben	Bestanden
Ansteuerung A/D-Bus	Schreiben und Lesen auf dem 8 bit A/D-Bus	Bestanden

Tabelle 6: Ergebnis der Software-Tests

5.3 System-Tests

In den Bereich der System-Tests fallen die Testfälle *Transputer-Kommunikation* und *Stabilität*. Beide Testfälle wurde erfolgreich abgeschlossen. Die Ergebnisse sind in Tabelle 7 dargestellt.

Testfall	Beschreibung	Ergebnis
Transputer-Kommunikation	Test der bidirektionalen Kommunikation zwischen dem EthLink Datenbussystem und einem Transputer	Bestanden
Stabilität	Einwöchiger Dauertest des EthLink Datenbussystems	Bestanden

Tabelle 7: Ergebnis der Systemtests

5.4 Systemintegration

Die Integration in den Rotorversuchsstand ist abgeschlossen, das EthLink Datenbussystem kommuniziert erfolgreich mit TEDAS II. Die Installation in das Gehäuse der Vielkanalmessanlage wurde ebenfalls vorgenommen, sodass einer Verwendung im operativen Betrieb nichts im Wege steht.

6 Fazit & Ausblick

Das EthLink Datenbussystem erfüllt die in der Aufgabenstellung formulierten Anforderungen komplett. Ein Bedienen der Vielkanalmessanlage TEDAS II ist über das Web-Interface in vollem Umfang möglich, sodass die vorhandenen, am Ende ihrer theoretischen Einsatzdauer angelangten Schnittstellenkarten ersetzt werden können. Die langfristige Einsatzbereitschaft von TEDAS II ist somit nicht mehr gefährdet.
Für die Ansteuerung von TEDAS II ist keine Software-Installation erforderlich, lediglich ein Browser wird benötigt. Dies stellt jedoch keine Einschränkung dar, da ein Browser bereits auf allen Computern im Netzwerk des Rotorversuchsstands vorhanden ist.

Ebenfalls einsatzbereit ist die transparente, bidirektionale Datenverbindung zwischen einem TCP-Client und einem Transputer. Sie ermöglicht die Weiterentwicklung und Modifikation des Programm-Codes von TEDAS II und somit die Anpassung der Vielkanalmessanlage an neue Anforderungen.

Das funktionsbereite EthLink Datenbussystem ist in Abbildung 29 dargestellt.

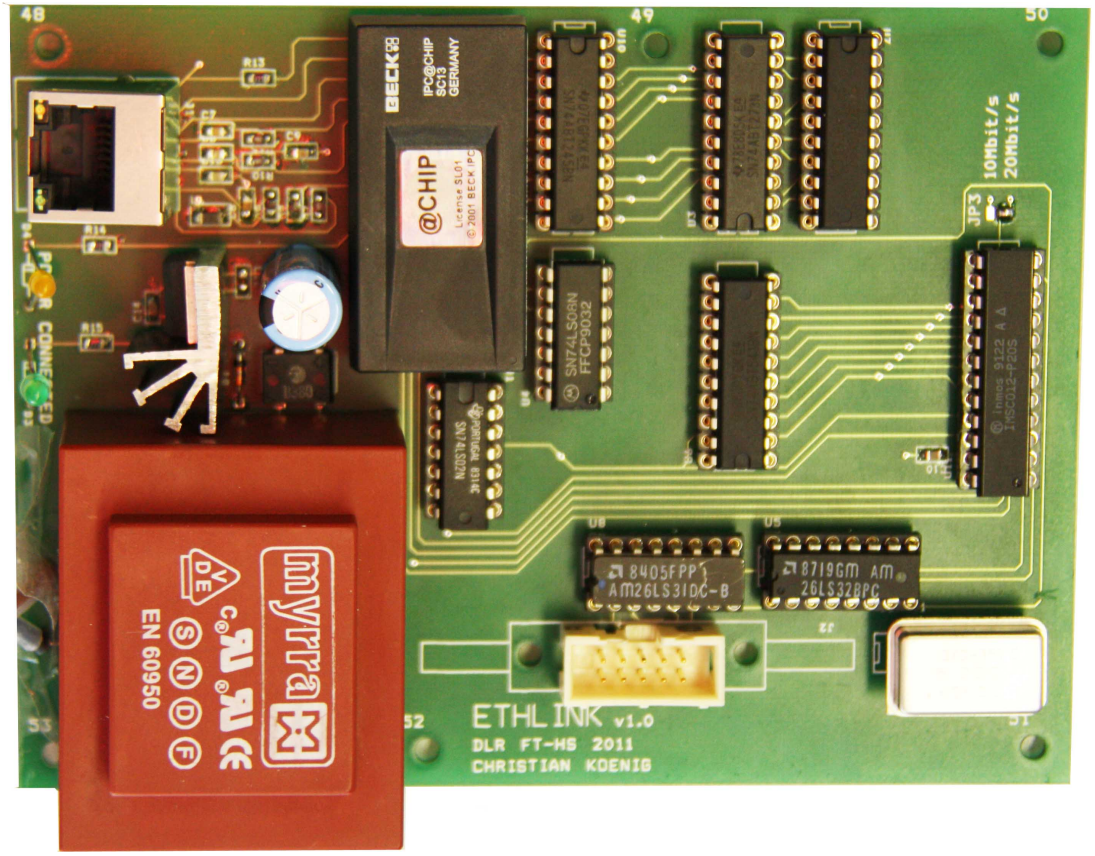

Abbildung 29: EthLink Datenbussystem

Literaturverzeichnis

[1] ASPINALL, D. et a.: *The Microprocessor and its Application - An advanced course*. 1. Auflage. Cambridge University Press, 1978

[2] BECK IPC: *SC1x Hardware Handbuch*. http://www.beck-ipc.com/files/manual/SC1XHW_v19_080801.pdf, 2008

[3] BECK IPC: *IPC@CHIP RTOS Documentation*. http://www.beck-ipc.com/files/api/scxxx/index.htm, 28.07.2011

[4] BÄHRING, Dr. H.: *Anwendungsorientierte Mikroprozessoren - Mikrocontroller und Digitale Signalprozessoren*. 4. Auflage. Springer Verlag, 2010

[5] DIJKSTRA, Prof. Dr. E. W.: Notes on Structured Programming / Technological University Eindhoven. 1970. – Forschungsbericht. – Second Edition

[6] DLR: *Corporate Design*. 15.08.2011

[7] DLR: *Portal*. http://www.dlr.de, 22.07.2011

[8] GELHAAR, B.: TEDAS II - High Speed Data Acquisition, Storage and Distribution. In: *Proceedings of the European Telemetry Conference* (2000)

[9] HOGENBOOM, Paul: *Mikroprozessor Datenbuch 1*. 2. Auflage. Elektor Verlag GmbH, 1990

[10] INMOS LIMITED: *Transputer Instruction Set*. Cambridge University Press, 1988

[11] PRINZ, Peter ; KIRCH, Ulla: *C. Einführung und professionelle Anwendung. IT-Studienausgabe*. 2., aktualisierte Auflage. REDLINE GMBH, 2007

[12] SARI, Sedar: Transputer Architecture and Parallel Applications / Bogaziçi Üniversitesi. 2003. – Forschungsbericht. – CMPE 511 Computer Architecture Term Paper

[13] SCHWANECK, H.-P. ; NIETSCHKE, W. ; BESEL, K.-G.: Der Transputer - Vorstellung und Anwendungsmöglichkeiten einer neuen Technologie / Deutsche Forschungs- und Versuchsanstalt für Luft- und Raumfahrt. 1986. – Forschungsbericht. – Institutsbericht IB 111-86/6

[14] SGS-THOMSON MICROELEKTRONICS: *IMS C012*. http://www.classiccmp.org/transputer/documentation/inmos/4254.pdf, 1995