SS ‘96

Bundesanstalt für Wasserbau, Karlsruhe
und
Fachhochschule Ostfriesland, Emden
Fachbereich Elektrotechnik und Informatik,
Labor für parallele Prozesse

Diplomarbeit

Programmierung und Test des Transputernetzes für die Hardware-Framegrabber der Strömungsmessung

Verfasser: Claus Brunzema
Matrikelnummer: 052964

Prüfer: Prof. Dr. Siemsen

Zweitprüfer: Prof. Dr. Schumacher

Inhaltsverzeichnis

1 Einleitung . . . 3
1.1 Die Bundesanstalt für Wasserbau . . . 3
1.2 Das Meßprinzip . . . 3
1.3 Der Aufbau des Meßsystems . . . 3
1.4 Die Aufgabenstellung . . . 4

2 Das Transputernetzwerk . . . 5
2.1 Aufbau der Transputer-Hardware . . . 5
2.2 Das Laden der Betriebssoftware auf das Transputernetz . . . 6

3 Die Software des Auswerterechners und der Framegrabber . . . 6
3.1 Die Betriebsarten . . . 6
3.1.1 Die Betriebsarten für das Transputernetzwerk . . . 6
3.1.2 Die zulässigen Reihenfolgen von Betriebsarten . . . 8
3.1.3 Das Synchronisieren der Betriebsart START . . . 9
3.2 Die Aufnahme aus der Sicht des Auswerterechners . . . 10
3.2.1 Das Programm FRGRAB . . . 10
3.2.2 Ablauf einer Aufnahme mit FRGRAB . . . 12
3.2.3 Kapazität des Systems . . . 13
3.2.4 FRGRAB in den Prozessen der Strömungsmessung SM . . . 14
3.3 Die Prozesse des Systems . . . 16
3.3.1 Überblick über die Prozesse des Testsystems . . . 16
3.3.2 Die Prozesse auf dem T1 . . . 17
3.3.2.1 Der Prozeß “T1/Testbild erzeugen” . . . 17
3.3.2.2 Der Prozeß “T1/BA verarbeiten” . . . 18
3.3.2.3 Der Prozeß “T1/LCK senden” . . . 18
3.3.3 Die Prozesse auf dem T2 . . . 19
3.3.3.1 Der Prozeß “T2/T1.manager” . . . 19
3.3.3.2 Der Prozeß “T2/BA.manager” . . . 20
3.3.4 Die Prozesse auf dem root-Transputer . . . 21
3.3.4.1 Der Prozeß “root/BA.manager” . . . 21
3.3.4.2 Der Prozeß “root/KF.manager” . . . 22
3.3.5 Die Prozesse im Programm FRGRAB . . . 22
3.3.5.1 Der Prozeß “FRGRAB/Parameter einlesen und LCK-Dateien öffnen” . . . 24
3.3.5.2 Der Prozeß “FRGRAB/Bildfolge holen” . . . 25
3.3.5.3 Der Prozeß “FRGRAB/FIFO2-Überlauf verarbeiten” . . . 26

Anhang A - Die einzelnen Programme des Testsystems . . . 27
1 Das Modul LINK . . . 27

2 Das System BASPEED 28
3 Das System NETSPEED 29
4 Das System LNKSPEED 31
5 Das System FRGRAB/FRAMExx 32
6 Das System J3TEST 34

Anhang B - Weiteres Vorgehen 36

Anhang C - Verwendete Komponenten 37

Anhang D - Literaturverzeichnis 38

Anhang E - Das System SMVIDEO für die Strömungsmessung 40

Anhang F - Programmlistings 42
1 Programmlistings für das Modul LINK 43
2 Programmlistings für das System BASPEED 49
3 Programmlistings für das System NETSPEED 58
4 Programmlistings für das System LNKSPEED 67
5 Programmlistings für das System FRGRAB/FRAMExx 73
6 Programmlistings für das System J3TEST 92

1 Einleitung

1.1 Die Bundesanstalt für Wasserbau

Die Bundesanstalt für Wasserbau (BAW) in Karlsruhe ist das zentrale Institut des Bundes für die praktische und wissenschaftliche Forschungsarbeit auf den Gebieten Wasserbau, Geotechnik und Bautechnik.

1.2 Das Meßprinzip

Von einem Fluß, dessen Strömungsverhalten untersucht werden soll, wird ein Modell gebaut. In diesem Modell strömt Wasser durch den Flußlauf. Auf der Wasseroberfläche schwimmen Styroporteilchen. Die Wege der Styroporteilchen werden von Videokameras aufgezeichnet. Da das Modell zu groß ist, um von einer einzigen Kamera mit akzeptabler Auflösung aufgenommen zu werden, werden mehrere Kameras eingesetzt, die jeweils einen Abschnitt des Modells aufzeichnen. Aus dem Weg der Teilchen auf dem Wasser werden Geschwindigkeiten und Beschleunigungen berechnet. In /H. Greiber/ und /J. Heyen/ ist das Prinzip der Strömungsmessung beschrieben.

1.3 Der Aufbau des Meßsystems

Bilddaten von bis zu 31 gleichzeitig laufenden Kameras werden zu einem Auswerterechner übertragen. Die Bilddaten werden kameraweise in verschiedenen Dateien gespeichert. Die Dateien mit den Bilddaten werden von einem Softwarepaket (Strömungsmessung SM /D. Siemens, W. Rülander, C. Brunzema/) ausgewertet.

Ein externer Synchrongenerator (SYNC) synchronisiert die Kameras und den Auswerterechner (vgl. Bild 1). Die Bilder der Kameras werden kameraweise digitalisiert (A/D-Wandlung). Ein LCA wandelt die Bilder in Bilder der beiden Graustufen weiß und schwarz, komprimiert sie mit einem Lauflängenverfahren und speichert sie im Format LCK (Lauflängen Codiertes Kompaktformat /W. Saathoff/) in einem FIFO1. Aus diesem FIFO1 werden die Daten von einem Transputer T1 ausgelesen und an einen Transputer T2 weitergereicht. Der Transputer T2 speichert die Bilddaten in einem FIFO2. Der T2 ist an ein Koppelfeld angeschlossen. Die Stationen von einem Kameraanschluß bis zu einem Koppelfeldanschluß werden als ein Framegrabber bezeichnet. Bis zu 31 Framegrabber übergeben ihre Informationen von ihren T2 an das Koppelfeld.

Das Koppelfeld wird von einem root-Transputer gesteuert. Der root-Transputer wählt eine Verbindung im Koppelfeld, holt die Bilddaten von dem Framegrabber ab und speichert sie in einem FIFO3. Der root-Transputer befindet sich auf einer Linkadapterkarte in einem Auswerterechner (PC). Das Programm FRGRAB auf dem Auswerterechner holt die Bilddaten über die Linkadapterkarte aus FIFO3 und speichert sie kameraweise getrennt auf Platte.

Bild 1 zeigt die Übersicht über den Aufbau des Systems.

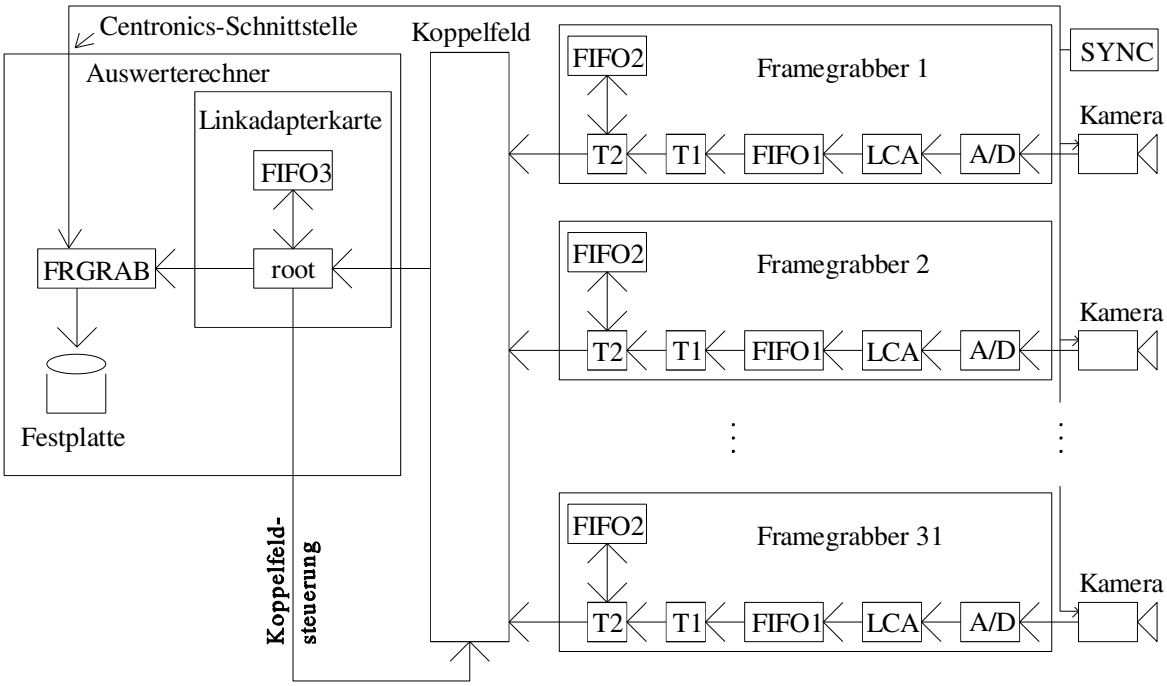

Bild 1: Überblick über das System Strömungsmessung

1.4 Die Aufgabenstellung

Die Einheiten Synchrongenerator, A/D-Wandler, LCA, FIFO1 und der Prozeß auf dem T1 sind noch nicht vollständig fertiggestellt. Diese Arbeit entwirft und implementiert die gesamten weiteren Komponenten des Systems. Zum Test wird auf T1 ein Prozeß installiert, der künstliche Bilddaten im Format LCK erzeugt. Insoweit ist der Prozeß auf T1 als vorläufig anzusehen. Die Prozesse auf T2, die Koppelfeldsteuerung, die Prozesse auf dem root-Transputer und das Programm FRGRAB werden entworfen, implementiert, dokumentiert und mit den künstlichen Bilddaten getestet. Die Implementation dieser Teile für das Gesamtsystem ist damit abgeschlossen. Der Synchronimpuls des Synchrongenerators wird am Auswerterechner mit einer Taste oder einem Pulsgenerator nachgebildet.

2 Das Transputernetzwerk

2.1 Aufbau der Transputer-Hardware

Die Linkadapterkarte mit dem root-Transputer im Auswerterechner (AR), das Koppelfeld (KF) und die Karten mit T1 und T2 werden mit ihren Links folgendermaßen verbunden:

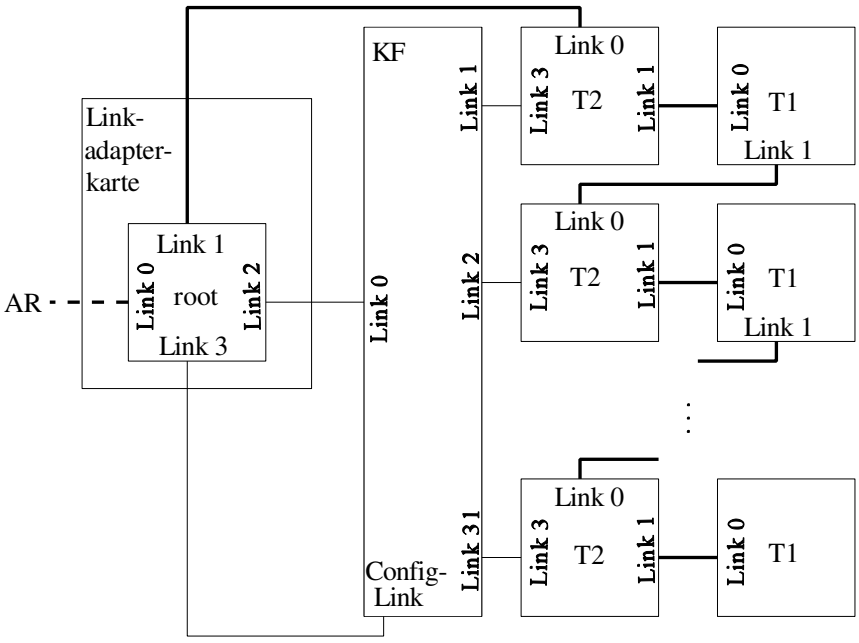

Bild 2: Hardware-Aufbau des Auswerterechners, des root-Transputers, des Koppelfeldes und der Framegrabber bis zum T1

Bei den fett gezeichneten Verbindungen werden nicht nur LinkIn, LinkOut und GND, sondern auch Error, Reset und Analyse verbunden. Alle Transputer bilden ein Up-Down-System. Die gestrichelt fett gezeichnete Verbindung von Link 0 des root-Transputers zum Auswerterechner führt über den ISA-Bus des Auswerterechners. Sie ist mit dem Einstecken der Linkadapterkarte automatisch geschaltet.

Am Koppelfeld darf zwischen zwei Framegrabbern kein Linkanschluß freigelassen werden. Weiterhin ist am ersten Anschluß des Koppelfeldes der erste Framegrabber anzuschließen. Installiert man z.B. ein System mit sieben Framegrabbern, so dürfen diese nur an die Anschlüsse eins bis sieben des Koppelfeldes angeschlossen werden. Will man später nur die Daten der Framegrabber zwei, drei und sechs auswerten, so werden die dazwischen liegenden Framegrabber softwaremäßig ausgeschlossen.

2.2 Das Laden der Betriebssoftware auf das Transputernetz

Die occam2-Tools liefern bootfähige Dateien mit der Endung *.BTL. Für jede mögliche Anzahl von Framegrabbern (1-31) wird eine eigene Datei erzeugt (FRAME01.BTL, FRAME02.BTL usw.). Mit dem Programm ISERVER wird eine *.BTL-Datei durch folgenden Aufruf in das Transputernetz geladen /INMOS/:

Prompt> ISERVER /sr /sc FRAMExx.BTL

Durch die Option /sr wird vor dem Programmstart ein Reset des Transputernetzwerkes ausgelöst, die *.BTL-Datei nach dem Parameter /sc wird in das Netzwerk geladen. Die Datei FRAMExx.BTL muß zu dem aufgebauten System passen: xx steht für die Anzahl der physisch angeschlossenen Framegrabber.

Das Programm ISERVER und später das Programm FRGRAB greifen auf die Basisadresse der Linkadapterkarte zu. Diese Adresse wird mit der Umgebungsvariablen TRANSPUTER eingestellt. Normalerweise ist TRANSPUTER=#150, d.h. die Basisadresse der Linkadapterkarte liegt auf hexadezimal 150.

Nach dem Aufruf von ISERVER ist das Transputernetzwerk betriebsbereit. Über die Linkadapterkarte werden Betriebsarten gesendet und Bilddaten empfangen.

3 Die Software des Auswerterechners und der Framegrabber

3.1 Die Betriebsarten

3.1.1 Die Betriebsarten für das Transputernetzwerk

Die Funktionen des Transputernetzwerkes werden durch das Senden von elementaren Betriebsarten gesteuert. Jede Betriebsart wird durch das Senden eines Kommandobytes vom Auswerterechner an den root-Transputer gesetzt. Es gibt genau fünf Betriebsarten, die das Netzwerk verarbeitet:

RESET <n> <FG-Nr> <FG-Nr> ... (Kommandobyte 1)

Mit dieser Betriebsart wird das Netzwerk einschließlich der Framegrabber zurückgesetzt. Nach dem Kommandobyte für die Betriebsart RESET erwartet das Netzwerk ein Byte n, das die Anzahl der für die folgende Aufnahme eingesetzten Framegrabber angibt. Diese Zahl muß kleiner oder gleich der Zahl der tatsächlich angeschlossenen Framegrabber sein. Nach der Anzahl folgen die einzelnen Nummern der Framegrabber in beliebiger Reihenfolge. In der Folge dieser Num-

mern werden die Framegrabber nach Informationen abgefragt. Eine Framegrabbernummer ergibt sich aus der Nummer des Koppelfeldlinks, an das der Framegrabber angeschlossen ist.

STEP <Schwelle> <Schwelle> ... (Kommandobyte 4)

Mit dieser Betriebsart werden die Schwellen der einzelnen Framegrabber eingestellt. Jede Schwelle ist ein Bytewert zwischen 16 und 240. Es werden genau so viele Schwellen gesendet, wie Framegrabber in der Betriebsart RESET eingestellt wurden. Die erste gesendete Schwelle bezieht sich auf den Framegrabber, dessen Nummer bei der Betriebsart RESET zuerst gesendet wurde, die zweite Schwelle auf den Framegrabber, dessen Nummer bei der Betriebsart RESET als zweites gesendet wurde usw.

START (Kommandobyte 2)

Mit dem Befehl für die Betriebsart START wird die Aufnahme gestartet. In der Betriebsart START werden Bilddaten einer Bildfolge in den FIFOs des Netzwerkes gespeichert. Sie können anschließend mit der Betriebsart BF (Bildfolge) abgeholt werden.

BF (Kommandobyte 5)

Mit der Betriebsart BF (Bildfolge) werden die Bilddaten vom Transputernetzwerk angefordert. Das Netzwerk antwortet auf die Betriebsart BF mit 32-Bit-Integerwerten. Der erste Wert vom Netzwerk ist die Anzahl der folgenden 32-Bit-Daten. Auf die Anzahl folgen die Daten. Zusätzlich zu den Bilddaten im Format LCK enthält der Datenstrom aus dem Netzwerk Informationen, die die Herkunft der Daten mit Framegrabbernummern, FIFO2-Überlauf und die vollständige Übertragung aller Daten signalisieren. Diese zusätzlichen Informationen sind mit 32-Bit-Werten codiert, die nicht als LCK-Datum auftreten können (HB und OF gelöscht, das Längenfeld ist auf Null gesetzt /W. Saathoff/). Die Zusatzinformationen haben folgende Form:

Framegrabbernummer

Die Framegrabbernummer (FG-Nr. 1-31) wird in das X-Feld des LCK-Datums geschrieben; OF, HB, das Längenfeld, HI und das Y-Feld sind gelöscht (hexadezimal 00000001 bis 0000001F). Diese Information gibt an, daß die folgenden LCK-Daten von dem bezeichneten Framegrabber stammen. Mit Hilfe dieser Informationen teilt das Programm auf dem Auswerterechner die Daten kameraspezifisch in verschiedene Dateien auf.

FIFO2-Überlauf

Bei einem FIFO2-Überlauf (FIFO2-OF) werden alle Felder des LCK-Datums gelöscht (hexadezimal 00000000). Wird ein FIFO2-Überlauf von einem Framegrabber gemeldet, hält der Auswerterechner die Aufnahme der Sequenz an (er sendet die Betriebsart STOP), liest die restlichen Bilddaten aus dem Netzwerk aus (Betriebsart BF) und startet automatisch eine neue Sequenz (Betriebsarten RESET, STEP, START, BF ...). Im Falle eines Überlaufs wird also die Bildfolge für alle Framegrabber beendet und automatisch eine weitere Sequenz angehängt (vgl. Abschnitt 3.3.5.3).

Datenende (einer Bildfolge)
Wird die Aufnahme durch die Betriebsart STOP angehalten und treffen keine neuen Daten von den Framegrabbern ein, signalisiert der root-Transputer die vollständige Übertragung aller Daten der Bildfolge mit dem folgenden LCK-Datum: HB, OF und das Längenfeld sind gelöscht; HI und alle Bits des Y- und X-Feldes sind gesetzt (hexadezimal 000FFFFF).

Die zusätzlichen Informationen Framegrabbernummer, FIFO2-Überlauf und Datenende werden vom Programm auf dem Auswerterechner herausgefiltert und verarbeitet. Sie gelangen nicht in die LCK-Dateien der Strömungsmessung. Die Programme, die Dateien im Format LCK verarbeiten und auswerten, werden nicht geändert.

STOP (Kommandobyte 3)
Mit dieser Betriebsart wird die Aufnahme angehalten. Nach dem Anhalten der Aufnahme holt das Programm auf dem Auswerterechner die restlichen Bilddaten mit der Betriebsart BF aus den Fifos des Transputernetzwerkes ab. Die Bildfolge ist beendet.

3.1.2 Die zulässigen Reihenfolgen von Betriebsarten

Mit dem Transputernetzwerk können Einzelbilder (für das Bestimmen der Schwelle und zum Vermessen) oder Bildfolgen (für die Strömungsmessung) gegrabbt werden. Für beide Anwendungsfälle gibt es je eine Reihenfolge von Betriebsarten, die sicherstellt, daß das Transputernetzwerk nicht in einen undefinierten Zustand gerät. Die zulässigen Reihenfolgen sind in Bild 3 dargestellt.

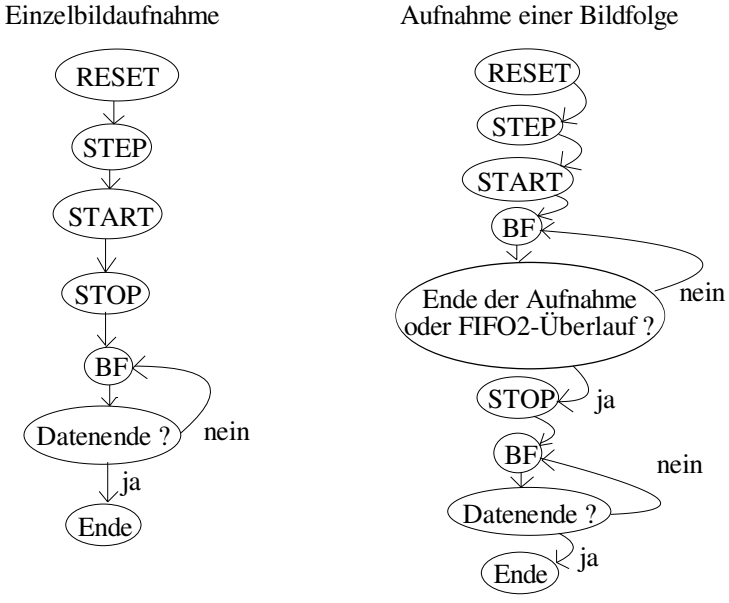

Bild 3: Zulässige Reihenfolgen von Betriebsarten

Das Aufnehmen eines Einzelbildes (zum Vermessen und zum Bestimmen der Schwelle) ist dadurch gekennzeichnet, daß auf die Betriebsart START unmittelbar die Betriebsart STOP folgt. Anschließend werden die Bilddaten (das Einzelbild) ausgelesen. Beim Aufnehmen einer Bildfolge (für die Strömungsmessung) wird gestartet und die Bildfolge übertragen. Signalisiert der Anwender das Ende der Messung oder tritt ein Überlauf in FIFO2 auf, so wird mit der Betriebsart STOP diese Bildfolge beendet. Die restlichen Daten der Bildfolge werden mit der Betriebsart BF abgeholt.

3.1.3 Das Synchronisieren der Betriebsart START

Die Betriebsart START wird mit dem Bildtakt der Kameras synchronisiert, damit alle Kameras die Aufnahme mit dem gleichen Bild beginnen. Unter dem Begriff Bildtakt ist der Start eines Synchronrahmens (FBAS) zu verstehen. Der Synchrongenerator für die Kameras schickt einen Impuls an den Auswerterechner, sobald die Betriebsart START gesendet werden darf. Dieser Impuls löst die Betriebsart START aus.

Nach /M. Hiltermann/ muß die Betriebsart START beim Einsatz von 31 Kameras innerhalb von 1,29 Millisekunden nach dem Synchronimpuls am LCA des ersten Framegrabbers anliegen. Das LCA des zweiten Framegrabbers erhält die Betriebsart START spätestens nach weiteren 1,29 Millisekunden. Werden weniger Kameras verwendet, so steht jeweils mehr Zeit zur Verfügung.

Der Synchronimpuls des Synchrongenerators wird an Pin 10 der Centronics-Schnittstelle (LPT1) des Auswerterechners geführt. Nachdem der Synchrongenerator den Pin 10 für mindestens 5 Mikrosekunden auf 0 (/H.Bernstein, Seite 146/) gezogen hat, sendet der Auswerterechner die Betriebsart START an das Transputernetzwerk. Zum Testen der Anbindung des Synchrongenerators an den Auswerterechner wird eine Taste zwischen Pin 10 der Centronics-Schnittstelle und Masse geschaltet. Die Taste kann von einem Pulsgenerator (z.B. WAVETEK 20 MHz Pulse/Function Generator Model 191) nachgebildet werden.

Die Zeit, die das Verteilen der Betriebsart START im Transputernetzwerk dauert, wird mit dem Programm BASPEED gemessen (Anhang A 2). Sie liegt für einen Transputer bei etwa 27 Mikrosekunden.

Die Betriebsart START erreicht also alle LCAs rechtzeitig vor dem Start des jeweils nächsten Bildes.

3.2 Die Aufnahme aus der Sicht des Auswerterechners

3.2.1 Das Programm FRGRAB

Das Programm FRGRAB läuft auf dem Auswerterechner. FRGRAB übernimmt die Kommunikation mit dem root-Transputer und sendet automatisch die Betriebsarten RESET, STEP, START, STOP und BF in der richtigen Reihenfolge.

Mit FRGRAB wird das Transputernetzwerk und das Koppelfeld in die Verarbeitungskette der Strömungsmessung eingebunden. Innerhalb von FRGRAB werden die Betriebsarten - gesteuert von der Kommandozeile - weitgehend automatisch ausgelöst. Vor dem Start von FRGRAB wird die Transputersoftware mit ISERVER auf das Transputernetzwerk geladen (Abschnitt 2.2). Mit Kommandozeilenparametern werden der Modus der Aufnahme (Einzelbild oder Bildfolge), die Framegrabbernummern und -schwellen und der Dateiname der LCK-Dateien eingestellt. Die Kommandozeilensyntax von FRGRAB lautet:

Prompt> FRGRAB s|m <FG-Information> <Dateiname> [-a] [-n]

Die erste Option (s oder m) gibt den Aufnahmemodus an. Mit s wird ein Einzelbild gegrabbt (s -> single), mit m wird eine Bildfolge gegrabbt (m -> multi). Im Einzelbildmodus wird FRGRAB nach der Aufnahme eines Einzelbildes verlassen. Bei der Aufnahme einer Bildfolge wird die Aufnahme von FRGRAB gestartet und durch Drücken einer beliebigen Taste beendet. <FG-Information> ist der Name einer Textdatei, in der zeilenweise Informationen über die Framegrabber für diese Aufnahme stehen. Der Anfang jeder Zeile ist eine Framegrabbernummer. Auf die Nummer folgt - durch ein Leerzeichen getrennt - die Schwelle des Framegrabbers. Will man für eine Aufnahme z.B. Framegrabber drei mit Schwelle 120, Framegrabber fünf mit Schwelle 143 und Framegrabber zwei mit Schwelle 100 verwenden, so stehen in der FG-Informationsdatei folgende Zeilen:

3 120
5 143
2 100

Framegrabber drei ist am Koppelfeld über Link drei angeschlossen, Framegrabber fünf ist am Koppelfeld über Link fünf angeschlossen usw. In diesem Beispiel sind mindestens noch die Framegrabber eins und vier an das Koppelfeld angeschlossen, sie werden jedoch nicht aktiviert. Die Reihenfolge der Zeilen in der <FG-Information> ist beliebig.

Die Daten im Format LCK, die FRGRAB aus dem Transputernetzwerk empfängt, werden framegrabberweise auf verschiedene Dateien verteilt. FRGRAB erwartet für jeden Framegrabber ein Unterverzeichnis mit dem Namen KAMERAxx im aktuellen Verzeichnis. xx steht für die zweistellige Framegrabbernummer. Jedes der KAMERAxx-Verzeichnisse enthält ein Verzeichnis mit dem Namen LCK. In diese LCK-Verzeichnisse schreibt FRGRAB Dateien im Format LCK

unter dem in der Kommandozeile angegebenen Dateinamen. Der Dateiname ist für ein Vermessungbild üblicherweise VERMESS.LCK, für ein Uferbild UFER.LCK; für eine Strömungsmessung ist der Dateiname <Modellname>.LCK. Bild 4 zeigt die Verzeichnisstruktur, in die FRGRAB in dem obigen Beispiel schreibt.

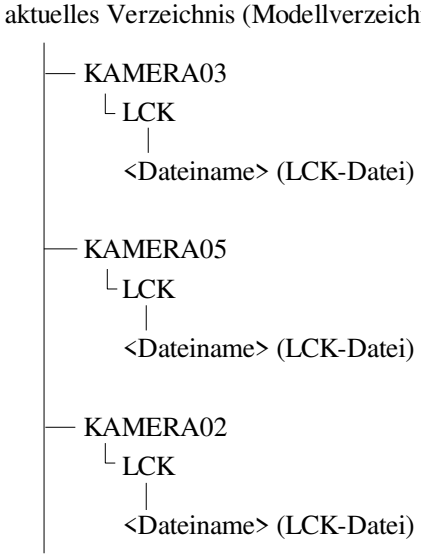

Bild 4: Verzeichnisstruktur für FRGRAB

Diese Verzeichnisstruktur stimmt mit der Verzeichnisstruktur überein, die von der Strömungsmessung im Verzeichnis DATA_SRC\<Modellname> angelegt wird /W. Rülander/.

Wird an FRGRAB die Option -a übergegeben, so werden die Daten an bestehende Dateien angehängt. Ohne die Option -a werden bereits bestehende Dateien von den neuen Daten überschrieben.

Wird die Option -n an FRGRAB übergeben, so wird das Synchronisieren der Betriebsart START auf den Synchronimpuls des Synchrongenerators unterdrückt (vgl. Abschnitt 3.1.3). FRGRAB sendet die Betriebsart START, ohne auf einen externen Impuls zu warten (Testbetrieb).

Fehlen für FRGRAB Kommandozeilenparameter (Modus, <FG-Information> und <Dateiname> müssen für eine Bildfolge angegeben werden), so erscheint eine Hilfe.

FRGRAB liefert die Quelldaten für die Strömungsmessung. FRGRAB wird benutzt, um Schwellen zu bestimmen (nicht implementiert); die Prozesse VM, UFER und SM lesen mit FRGRAB Daten für die Strömungsmessung (vgl. Abschnitt 3.2.4, nicht implementiert).

3.2.2 Ablauf einer Aufnahme mit FRGRAB

Für eine Aufnahme arbeiten das Programm FRGRAB, der Prozeß auf dem root-Transputer und die Prozesse auf T2 und T1 zusammen. Bild 5 zeigt einen Überblick über die Prozesse.

Zuerst schickt das Programm FRGRAB dem Transputersystem die Betriebsart RESET zusammen mit der Anzahl der Framegrabber und den Framegrabbernummern aus der FG-Informationsdatei.

Der root-Transputer löscht FIFO3. Aus der Anzahl der Framegrabber wird die maximale Blockgröße berechnet, die jeder Framegrabber senden darf. Diese maximale Blockgröße stellt sicher, daß in einer 20 Millisekunden-Zeitscheibe jeder Framegrabber Daten senden kann. Sind zwei Framegrabber angeschlossen, so wird die maximale Blockgröße so berechnet, daß das Senden eines maximal großen Blockes höchstens 10 Millisekunden benötigt, bei drei Framegrabbern wird die Blockgröße so berechnet, daß das Senden eines maximalen Blockes höchstens 6,6 Millisekunden dauert usw.

Der root-Transputer speichert die Framegrabbernummern, die von FRGRAB gesendet werden. Der root-Transputer wählt die Framegrabber an und übermittelt jedem T2 die Betriebsart RESET zusammen mit der maximalen Blockgröße. Die T2 löschen ihr FIFO2, speichern die maximale Blockgröße und leiten die Betriebsart RESET an die T1 weiter.

In der Betriebsart STEP empfängt der root-Transputer die einzelnen Schwellen von FRGRAB und wählt die Framegrabber an, deren Nummern bei RESET gespeichert wurden. Jedem Framegrabber wird die Betriebsart STEP und die zugeordnete Schwelle übermittelt. Die T2 empfangen Betriebsart und Schwelle und leiten beides an T1 weiter.

Mit der Option -n sendet FRGRAB unmittelbar mit dem Aufruf die Betriebsart START an den root-Transputer. Fehlt die Option -n, so wartet FRGRAB auf den Synchronimpuls des Synchrongenerators. Nach dem Synchronimpuls schickt FRGRAB die Betriebsart START an den root-Transputer.

Der root-Transputer wählt nacheinander alle aktiven Framegrabber an (vgl. Übergabeparameter <FG-Information> von FRGRAB). Die T2 erhalten die Betriebsart START. Die Betriebsart START wird von den T2 an die T1 weitergeleitet. In der Betriebsart START fragt der root-Transputer selbsttätig die Framegrabber mit der Betriebsart BF ab. Die T2 erwarten in der Betriebsart START Daten von den T1.

Die Testdaten werden vom T1 an T2 gesendet und dort in FIFO2 gespeichert. Läuft FIFO2 über, so fügt der T2 eine Overflow- und eine Bildendekennung in den Datenstrom ein. Der T2 überwacht ständig die HI-Information in den Daten von T1. Eine Halbbildendekennung beim Überlauf wird mit der korrekten HI-Information eingetragen. Tritt der Überlauf in einem ungeraden Halbbild auf, so wird eine weitere Halbbildendekennung (mit der richtigen HI-Information)

in den Datenstrom eingetragen. Ist das Bild vollständig abgeschlossen, wird ein Sequenzende- und ein FIFO2-Überlaufkennzeichen eingetragen. Nach einem Überlauf in FIFO2 werden die Daten vom T1 noch entgegengenommen, aber nicht gespeichert. FIFO2 ist für Schreibzugriffe gesperrt. Erst wenn die Betriebsart RESET verschickt worden ist, wird die Sperre aufgehoben.

In der Betriebsart START werden die einzelnen Framegrabber vom root-Transputer selbsttätig mit der Betriebsart BF abgefragt. Bevor Bilddaten in FIFO3 eingetragen werden, wird die Framegrabbernummer des Framegrabbers, der die Daten geliefert hat, mit einer Sonderkennung in FIFO3 eingetragen. Jedes Sequenzendezeichen im Datenstrom wird vom root-Transputer für das Erkennen des Datenendes mitgezählt.

FRGRAB sendet für eine Einzelbildaufnahme nach der Betriebsart START die Betriebsart STOP. Bei der Aufnahme einer Bildfolge holt FRGRAB nach START die Bilddaten mit der Betriebsart BF vom root-Transputer ab. Die Betriebsart BF wird vom root-Transputer nicht an die Framegrabber weitergeleitet, da die Framegrabber vom root-Transputer selbsttätig parallel mit der Betriebsart BF abgefragt werden. FRGRAB überwacht den ankommenden Datenstrom. Bei einer neuen Framegrabbernummer wird die Zieldatei für die LCK-Daten gewechselt, bei einem Überlauf in FIFO2 wird die Aufnahme angehalten (Betriebsart STOP) und nach dem Leeren der Fifos (Betriebsart BF) neu gestartet (Betriebsarten RESET, STEP und START).

Sendet FRGRAB die Betriebsart STOP an den root-Transputer, so leitet der root-Transputer diese Betriebsart an alle aktuell benutzten Framegrabber weiter. Auch in der Betriebsart STOP fragt der root-Transputer selbsttätig die Framegrabber mit der Betriebsart BF ab, um die restlichen Daten der Aufnahme zu erhalten. Hat der root-Transputer so viele Sequenzendezeichen empfangen, wie Framegrabber benutzt werden, so fügt er die Datenendekennung in den Datenstrom ein. FRGRAB holt in der Betriebsart STOP noch solange Daten mit der Betriebsart BF aus dem Transputernetz ab, bis der root-Transputer die Datenendekennung übermittelt.

Der T2 leitet die Betriebsart STOP an T1 weiter. Der T2 verhält sich in der Betriebsart STOP wie in der Betriebsart START: Bilddaten werden von T1 abgeholt. T1 sendet in der Betriebsart STOP den Rest den aktuellen Bildes und das Sequenzendezeichen.

3.2.3 Kapazität des Systems

Im Übertragungsweg der Bilddaten ist die Übertragung vom root-Transputer zum Auswerterechner eine Engstelle (vgl. Anhang A 4). Ist FIFO3 gefüllt, so werden keine Daten von den Framegrabbern abgefragt, die Bilddaten stauen sich in den FIFO2 der Framegrabber. Läuft eines der FIFO2 über, so wird automatisch die Sequenz beendet; anschließend werden die aufgestauten Daten an den Auswerterechner übertragen. Eine weitere Sequenz wird automatisch angehängt usw.

Die FIFO2 speichern bei Transputerkarten von 4 MB (für T2) je etwa 4 MB LCK-Daten. Bei typisch 25 Teilchen pro Bild (400 Byte oder ca. 10 kB/s) reicht der Speicherplatz auf einer

Transputerkarte für etwa 400 Sekunden (ca. 7 Minuten). Bei 15 Framegrabbern werden pro Sekunde ca. 1,2 MBit/s auf den Auswerterechner übertragen; bei dieser Zahl von Framegrabbern hält das Aufzeichnen mit dem Erfassen also Schritt.

Werden mehr Framegrabber eingesetzt und sollen Lücken in der Aufzeichnung vermieden werden, sind entweder die Transputerkarten für T2 mit 8 MB auszustatten, oder (vermutlich die preiswertere Lösung) eine Linkadapterkarte mit einer effektiven Datenrate von mindestens 3 MBit/s ist einzusetzen. Vorerst erscheint eine *zusammenhängende* Sequenz von einigen Minuten ausreichend; mit Unterbrechungen nach diesen Zeiten kann weiter aufgezeichnet werden, bis der Speicherplatz für LCK-Dateien die Plattenkapazität erreicht. Bei 30 Kameras und einer Meßzeit von 15 Minuten liegen etwa 300 MB auf Platte.

3.2.4 FRGRAB in den Prozessen der Strömungsmessung SM

Die Prozesse VM, UFER und SM der Strömungsmessung erhalten über das Programm FRGRAB Bilddaten aus dem Transputernetzwerk.

Die Batchdatei SESSION_.BAT zum Eröffnen einer Sitzung der Strömungsmessung enthält das Kommando zum Laden der Betriebssoftware auf das Transputernetz (vgl. Abschnitt 2.2, nicht implementiert). Die Anzahl der an das Koppelfeld angeschlossenen Framegrabber wird in einer Konfigurationsdatei SM.cfg eingestellt. Diese Anzahl der ‘angeschlossenen’ Framegrabber kann verschieden von der Anzahl der ‘existierenden Kameras’ und der Anzahl der ‘benutzten Kameras’ aus der Datei <Modell>.SM eines Modells sein /W. Rülander, C. Brunzema/. Mit der modellunabhängigen Angabe der Anzahl der angeschlossenen Framegrabber beim Eröffnen einer Sitzung der Strömungsmessung wird ein Hardwareumbau beim Modellwechsel (mit SMCM) innerhalb einer Sitzung vermieden. Mit der Angabe der ‘benutzten Kameras’ in der Datei <Modell>.SM werden überzählige Framegrabber softwaremäßig von einer Aufnahme ausgeschlossen.

Innerhalb der Sitzung ist das Transputernetzwerk betriebsbereit.

Das Bestimmen der Schwelle wird mit dem Prozeß EM (Einmessen) automatisiert. Der Prozeß EM ist nicht implementiert. Zur Zeit wird die ideale Schwelle mit SMITER ermittelt. SMITER untersucht nur eine Kamera und trägt das Ergebnis nicht automatisch in die <Modell>.SM ein /C. Brunzema/. Der Prozeß EM bestimmt für jede benutzte Kamera die Schwelle und trägt die Schwellen automatisch in die Datei <Modell>.SM des Modells ein.

Der Prozeß EM verwendet von jeder Kamera ein Einzelbild, in dem nur Vermessungsteilchen sichtbar sind. EM setzt auf Daten vom Videorecorder oder auf Daten vom den Framegrabbern auf.

Quelldaten vom Videorecorder (EM-Startformat PCX) werden in das Format EBM gewandelt. Ein Bild im Format EBM wird mit einer Testschwelle in das Format LCK gewandelt. Das LCK-Bild wird über das Format LCH in das Format TLL gewandelt. Je nach der Anzahl der erkannten Teilchen im TLL-Bild wird die Testschwelle verändert und ein neuer Durchlauf begonnen (EBM->LCK->LCH->TLL).

Für Quelldaten von den Framegrabbern (EM-Startformat net, siehe unten) ruft EM das Programm FRGRAB mit der Option s (Einzelbild) auf. Die Batchdatei SMADJ.BAT der Strömungsmessung erzeugt aus der Datei <Modell>.SM eines Modells die FG-Informationsdatei für dieses Modell (vgl. Abschnitt 3.2.1, nicht implementiert). In der FG-Informationsdatei ist eine Testschwelle für jede Kamera eingetragen. Die von FRGRAB gelieferten Daten im Format LCK werden bis in das Format TLL gewandelt. Anhand der Anzahl der Teilchen in den TLL-Bildern werden die Testschwellen verändert. Für den nächsten Durchlauf wird FRGRAB mit den neuen Testschwellen aufgerufen.

Der Prozeß EM findet in mehreren Durchläufen mit einer binären Suche die ideale Schwelle für jede Kamera.

Die Prozesse EM, VM, UFER und SM erhalten ein Startformat net (für Netzwerk, nicht implementiert). Ist dieses Startformat in der Datei <Modell>.SM angewählt, so werden keine Bilddaten aus Dateien auf der Platte verarbeitet, sondern mit dem Aufruf der Prozesse werden neue Bilddaten mit FRGRAB aus dem Transputernetzwerk geholt. FRGRAB legt die Daten aus dem Transputernetzwerk im Format LCK auf Platte ab; die Daten werden - vom Stopformat gesteuert - von den Prozessen weiterverarbeitet. Ob Daten im Format LCK überschrieben werden, hängt von dem Schalter append_lck in der Datei <Modell>.SM ab.

Ist das Startformat net ausgewählt, so wechseln die Prozesse EM, VM, UFER und SM in das Verzeichnis DATA_SRC\<Modellname>. Die Prozesse EM, VM und UFER rufen in diesem Verzeichnis FRGRAB mit der Option s (Einzelbild) auf.

Der Prozeß SM ruft FRGRAB mit der Option m (Bildfolge) auf. Der Schalter append_lck aus der Datei <Modell>.SM entscheidet über die Option -a von FRGRAB. Der Dateiname für FRGRAB entspricht dem Modellnamen mit der Endung .LCK.

FRGRAB automatisiert die gesamte Steuerung des Transputernetzes (Senden der Betriebsarten, Empfang der Daten usw.). Nach dem Aufruf von FRGRAB liegen die Bilddaten im Format LCK auf Platte. Die weiteren Prozesse der Strömungsmessung, die automatisch bis zum Stopformat ablaufen, verarbeiten diese Dateien.

Das Einbinden von FRGRAB in die Strömungsmessung ist nicht implementiert.

3.3 Die Prozesse des Systems

3.3.1 Überblick über die Prozesse des Testsystems

Auf dem Testsystem läuft auf T1 ein Prozeß, der künstliche Bilddaten generiert. Dieser Prozeß ist so gestaltet, daß anhand der Bilddaten jeder T1 eindeutig vom anderen unterschieden werden kann. Auf T2, dem root-Transputer und auf dem Auswerterechner laufen dieselben Prozesse, die später reale Bilddaten verarbeiten. Um reale Bilddaten zu verarbeiten, wird der Prozeß auf T1 gegen einen Prozeß ausgetauscht, der Daten vom FIFO1 entgegennimmt und Betriebsarten an das LCA schickt (nicht implementiert, ein Minimalprogramm zum Testen der Kommunikation zwischen T1 und LCA trägt den Namen J3TEST, Anhang A 6). Die Prozesse auf T2, dem root-Transputer und dem Auswerterechner werden mit dem Testsystem getestet. Bild 5 zeigt einen Überblick über die Prozesse des Testsystems. Die Prozesse werden im folgenden beschrieben.

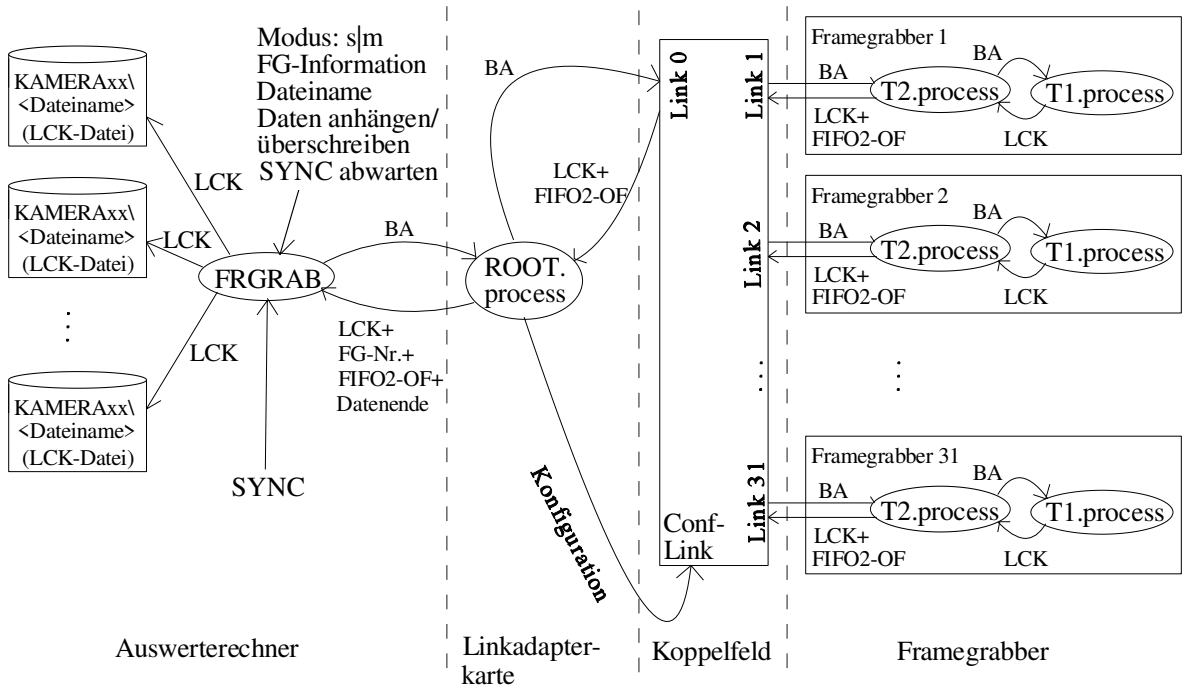

Bild 5: Übersicht über die Prozesse im Testsystem

3.3.2 Die Prozesse auf dem T1

Auf T1 wird zu Testzwecken im Speicher ein künstliches Bild im Format LCK gebildet. Das Bild enthält die maximale Anzahl Lauflängen (100 Teilchen zu je vier Lauflängen ergeben 400 Lauflängen). Die Lauflängen bilden weiße Blöcke, deren Anzahl von der Nummer des Framegrabbers abhängt. Das Bild von Framegrabber eins enthält einen großen Block, das Bild von Framegrabber zwei enthält zwei kleinere Blöcke (die 400 Lauflängen werden auf zwei Blöcke aufgeteilt) usw.

Von Betriebsarten gesteuert wird das Bild an T2 geschickt. Wurde das Testbild vollständig übertragen, wird es um drei Pixel verschoben. Das verschobene Bild wird als nächstes Bild übertragen usw. Bild 6 zeigt die Prozesse auf dem T1 im Überblick.

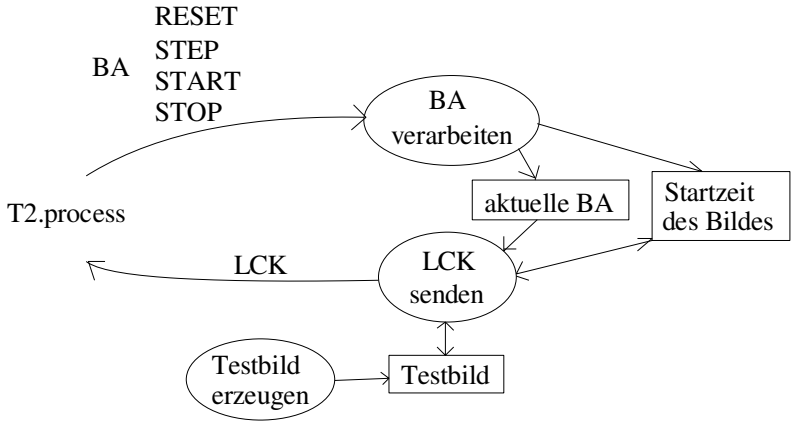

Bild 6: Prozesse auf dem T1 im Testsystem

Die Prozesse auf dem T1 sind in der occam2-Quelldatei T1.occ enthalten.

3.3.2.1 Der Prozeß “T1/Testbild erzeugen”

Beim Start des Transputersystems erzeugt dieser Prozeß ein Testbild aus 400 Lauflängen im Speicher des T1. Das Testbild enthält eine Anzahl weißer Blöcke entsprechend der Framegrabbernummer. Das Bild ist in zwei Halbbilder aufgeteilt. Nach jedem Halbbild wird die Sonderkennung für den Halbbildwechsel mit korrekter HI-Information eingetragen /W. Saathoff/. Insgesamt liegen so 402 32-Bit-Worte im Format LCK als Testbild im Speicher des T1.

3.3.2.2 Der Prozeß "T1/BA verarbeiten"

Dieser Prozeß nimmt die Betriebsarten vom T2 entgegen. Jede empfangene Betriebsart ändert die Zustandsvariable "aktuelle Betriebsart". Die Betriebsart RESET löst im Testsystem auf T1 keine Aktionen aus. Nach der Betriebsart STEP wird eine Schwelle entgegengenommen. Diese Schwelle wird im Testsystem auf T1 nicht weiter verwendet (das Testbild im Speicher des T1 ist bereits im Schwarz-Weiß-Format LCK). Die Betriebsart START startet das Übertragen der Bilddaten. Der Zeitpunkt des Bildstarts wird gespeichert. Das erste LCK-Datum des Bildes wird gesendet. Der Zeiger auf das nächste zu sendende LCK-Datum wird gesetzt. Die Betriebsart STOP hält das Übertragen der Bildfolge an, die Restdaten des aktuellen Bildes werden noch gesendet.

3.3.2.3 Der Prozeß "T1/LCK senden"

Der Prozeß T1/LCK senden prüft ständig die aktuelle Betriebsart. Ist die aktuelle Betriebsart START, wird das nächste LCK-Datum des Testbildes an T2 gesendet. Der Zeiger für das nächste zu sendende LCK-Datum wird weiterbewegt.

Der folgende Mechanismus beschreibt, wie aus dem Testbild eine Bildfolge entsteht. Wurde das letzte LCK-Datum eines Bildes gesendet, so werden die X-Felder aller LCK-Daten im Testbild um drei erhöht. Dadurch wandern die Blöcke von einem Bild zum nächsten nach rechts. Erreicht ein Block den rechten Bildrand, wird er wieder an den linken Bildrand gesetzt. Am Ende eines Bildes wird solange gewartet, bis 40 Millisekunden seit dem Bildstart verstrichen sind; in einem 40 Millisekunden-Abschnitt wird also nur ein (Voll-)Bild gesendet. Nach der Wartezeit wird die Zeit als Startzeit des neuen Bildes gespeichert.

Ist die aktuelle Betriebsart STOP, so wird das nächste LCK-Datum an T2 gesendet und der Zeiger für das nächste LCK-Datum weiterbewegt. Ist das Bildende erreicht, so wird ein Sequenzendezeichen an T2 gesendet. Nach dem Sequenzendezeichen werden keine Daten übertragen. Die Datenübertragung wird erst durch die Betriebsart START neu gestartet.

3.3.3 Die Prozesse auf dem T2

Auf T2 werden die Daten vom T1 entgegengenommen und im FIFO2 gespeichert (vgl. Bild 7). Mit der Betriebsart BF werden die Daten vom root-Transputer aus FIFO2 angefordert. Im Falle eines Überlaufs in FIFO2 trägt der T2 eine Sonderkennung in FIFO2 ein.

Die Betriebsarten RESET, STEP, START und STOP werden ausgewertet und an T1 weitergeleitet. Bild 7 zeigt die Prozesse auf dem T2 im Überblick, die folgenden Abschnitte beschreiben die Prozesse.

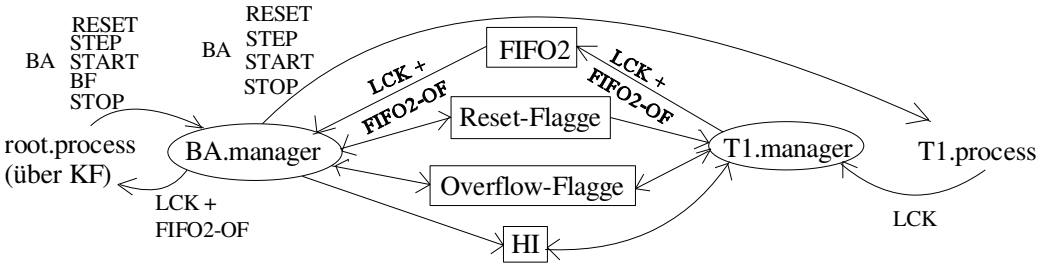

Bild 7: Prozesse auf dem T2

Die Prozesse auf dem T2 sind in der occam2-Quelldatei T2.occ enthalten.

3.3.3.1 Der Prozeß "T2/T1.manager"

Dieser Prozeß holt ständig Daten vom T1. Ist die Reset-Flagge nicht gesetzt, so wird versucht, das LCK-Datum im FIFO2 einzutragen. Der noch freie Platz in FIFO2 wird berechnet. Sind mehr als sechs Speicherplätze im FIFO2 frei und ist die Overflow-Flagge gelöscht, so wird das LCK-Datum eingetragen. Ist das LCK-Datum eine Sonderkennung für den Halbbildwechsel, so wird die Information HI des nächsten Bildes berechnet und gespeichert. Dem T2 ist so ständig die Information HI des laufenden Bildes bekannt.

FIFO2 bietet Platz für etwa 1000000 Lauflängen (Transputerkarte mit 4 MB RAM für T2). Sind noch genau sechs Speicherplätze im FIFO2 frei, so wird ein Überlauf signalisiert. Die Overflow-Flagge wird gesetzt. Mit fünf der verbleibenden Speicherplätze wird das Bild abgeschlossen. In FIFO2 werden ein LCK-Datum mit OF und ein Halbbildende mit der richtigen HI-Information eingetragen. Ist das aktuelle Halbbild ein ungerades Halbbild, so wird ein weiteres Halbbildende eingetragen. Anschließend wird ein Sequenzendekennzeichen und ein FIFO2-Überlaufkennzeichen eingetragen. So erzeugt der T2 ein Sequenzende mit korrektem Bildende, bevor der Überlauf in FIFO2 signalisiert wird.

Sind noch sechs Speicherplätze in FIFO2 frei, wird ein Überlauf in FIFO2 signalisiert. Bildabschluß, Sequenzendezeichen und FIFO2-OF-Kennung belegen maximal fünf Speicherplätze. Lese- und Schreibzeiger des FIFO2 sind nach dem Eintragen von Bildende, Sequenzende und FIFO2-OF-Kennung garantiert noch um mindestens eins verschieden. Zeigen Lese- und Schreibzeiger des FIFO2 auf die gleiche Speicherstelle, so ist das das Zeichen für ein leeres FIFO2.

3.3.3.2 Der Prozeß "T2/BA.manager"

Dieser Prozeß verarbeitet die Betriebsarten, die vom root-Transputer eintreffen.

In der Betriebsart RESET wird die Reset-Flagge gesetzt, die Overflow-Flagge gelöscht. Die Information HI wird mit 'ungerades Halbbild' initialisiert. FIFO2 wird gelöscht. Die Betriebsart RESET wird an T1 weitergeleitet. Vom root-Transputer wird nach RESET die maximale Blockgröße entgegengenommen und gespeichert.

In der Betriebsart STEP wird die Schwelle vom root-Transputer entgegengenommen und an T1 weitergeleitet.

In der Betriebsart START wird die Reset-Flagge gelöscht. Dies ist das Zeichen für T1.manager, Daten in FIFO2 einzutragen. Die Betriebsart START wird an T1 weitergeleitet.

Auf die Betriebsart BF antwortet der T2 mit einem Datenblock aus FIFO2, falls Daten im FIFO2 stehen. Die Größe des Datenblocks in FIFO2 wird berechnet. Der Datenblock wird an den root-Transputer geschickt. Ist der Datenblock größer als die maximale Blockgröße, so wird ein Datenblock mit der maximalen Blockgröße geschickt.

Die Betriebsart STOP wird an T1 weitergeleitet. Von T1 werden auch in der Betriebsart STOP restliche Daten entgegengenommen.

3.3.4 Die Prozesse auf dem root-Transputer

Der root-Transputer empfängt Betriebsarten von FRGRAB. Bilddaten werden an FRGRAB gesendet.

Der root-Transputer steuert das Koppelfeld und schickt Betriebsarten an die einzelnen Framegrabber. In den Betriebsarten START und STOP holt der root-Transputer selbsttätig Datenblöcke von den T2 in sein FIFO3. Die Herkunft der Daten (welcher T2) wird in einer Sonderkennung vermerkt. Sendet FRGRAB die Betriebsart BF, so schickt der root-Transputer einen Datenblock aus FIFO3 an FRGRAB. In der Betriebsart STOP wird als Antwort auf die Betriebsart BF eine Datenendekennung an FRGRAB gesendet, falls der root-Transputer von allen Framegrabbern das Sequenzendezeichen empfangen hat. Bild 8 zeigt die Prozesse auf dem root-Transputer im Überblick, die folgenden Abschnitte beschreiben die Prozesse.

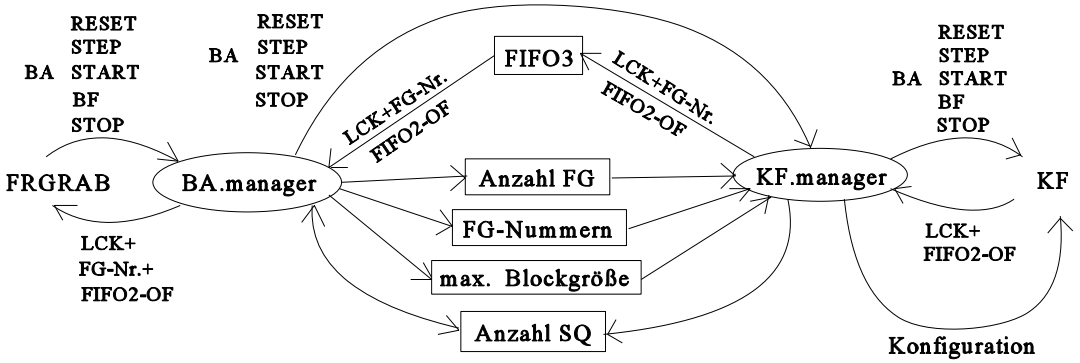

Bild 8: Prozesse auf dem root-Transputer

Die Prozesse auf dem root-Transputer sind in der occam2-Quelldatei ROOT.occ enthalten.

3.3.4.1 Der Prozeß “root/BA.manager”

Dieser Prozeß nimmt die Betriebsarten von FRGRAB entgegen. Die Betriebsart BF wird vom Prozeß “root/BA.manager” selbst verarbeitet. Die übrigen Betriebsarten werden an den Prozeß “root/KF.manager” weitergeleitet.

In der Betriebsart RESET wird FIFO3 gelöscht. Von FRGRAB wird die Anzahl der benutzten Framegrabber entgegengenommen. Aus dieser Anzahl wird die maximale Blockgröße für die Framegrabber nach folgender Formel ermittelt:

$$Blockgröße = TRUNC\left(\frac{20ms}{Anzahl\ \ Framegrabber} * 312\ \ Lauflängen\ \ pro\ \ ms\right) \qquad \textbf{(1)}$$

Die Anzahl der Lauflängen, die pro Millisekunde gesendet werden, wurde mit dem System NETSPEED ermittelt (Anhang A 3). Die Framegrabbernummern, die von FRGRAB nach der Anzahl der Framegrabber übermittelt werden, werden entgegengenommen und gespeichert. Anschließend wird die Betriebsart RESET an “root/KF.manager” geschickt.

In der Betriebsart STEP werden die Schwellen von FRGRAB angenommen und zusammen mit der Betriebsart STEP an “root/KF.manager” geschickt.

Die Betriebsart START wird an “root/KF.manager” weitergeleitet.

Die Betriebsart BF wird von “root/BA.manager” verarbeitet. Sie wird nicht gespeichert. Befindet sich “root/BA.manager” vor BF z.B. in der Betriebsart START, so bleibt diese Betriebsart nach der Verarbeitung der Betriebsart BF erhalten. Ein Datenblock wird als Antwort auf die Betriebsart BF aus FIFO3 an FRGRAB geschickt. Ist FIFO3 in der Betriebsart STOP leer und wurden bereits von allen Framegrabbern Sequenzendezeichen (SQ) empfangen, so wird das Datenendezeichen an FRGRAB geschickt.

Die Betriebsart STOP wird an “root/KF.manager” weitergeleitet.

3.3.4.2 Der Prozeß “root/KF.manager”

Dieser Prozeß verwaltet das Koppelfeld. Betriebsarten werden von “root/BA.manager” entgegengenommen. Die einzelnen Framegrabber werden der Reihe nach angewählt und die Betriebsarten werden an jeden einzelnen Framegrabber geschickt. Mit der Betriebsart RESET wird die maximale Blockgröße, mit der Betriebsart STEP die jeweilige Schwelle des Framegrabbers gesendet. Die Betriebsarten START und STOP werden ohne Zusatzangaben an die Framegrabber geschickt.

Liegt keine neue Betriebsart an und befindet sich “root/KF.manager” in der Betriebsart START oder STOP, so werden die Framegrabber mit der Betriebsart BF abgefragt. Ist im FIFO3 noch Platz für eine Kennung mit der Framegrabbernummer und einen Datenblock der maximalen Größe, so wird der nächste Framegrabber angewählt und die Betriebsart BF gesendet. Der angewählte Framegrabber antwortet mit einem Datenblock. Der Datenblock wird zusammen mit der Kennung für die Framegrabbernummer in FIFO3 geschrieben.

Nach der Datenübertragung wird geprüft, ob eine neue Betriebsart anliegt. Falls nein, wird der nächste Framegrabber abgefragt. Falls ja, wird die neue Betriebsart verarbeitet.

3.3.5 Die Prozesse im Programm FRGRAB

Das Programm FRGRAB läuft auf dem Auswerterechner (PC mit x86-Prozessor). FRGRAB greift *nicht* auf den Fileserver des Dienstprogramms ISERVER zu. Das Dienstprogramm ISERVER wurde vor Aufruf von FRGRAB einmal aufgerufen, um die Transputer selbst zu

initialisieren (vgl. Abschnitt 2.2). FRGRAB steuert das Transputernetzwerk. Die Betriebsarten werden automatisch in der richtigen Reihenfolge an den root-Transputer gesendet, Bilddaten werden entgegengenommen und auf verschiedene Dateien verteilt. Bei einem Überlauf in FIFO2 eines Framegrabbers wird die Aufnahme automatisch angehalten (Sequenzende); die restlichen Daten aller Framegrabber werden aus dem Netzwerk abgeholt und eine weitere Sequenz wird automatisch gestartet und an die vorige Sequenz angehängt. Bild 9 zeigt die Prozesse von FRGRAB im Überblick.

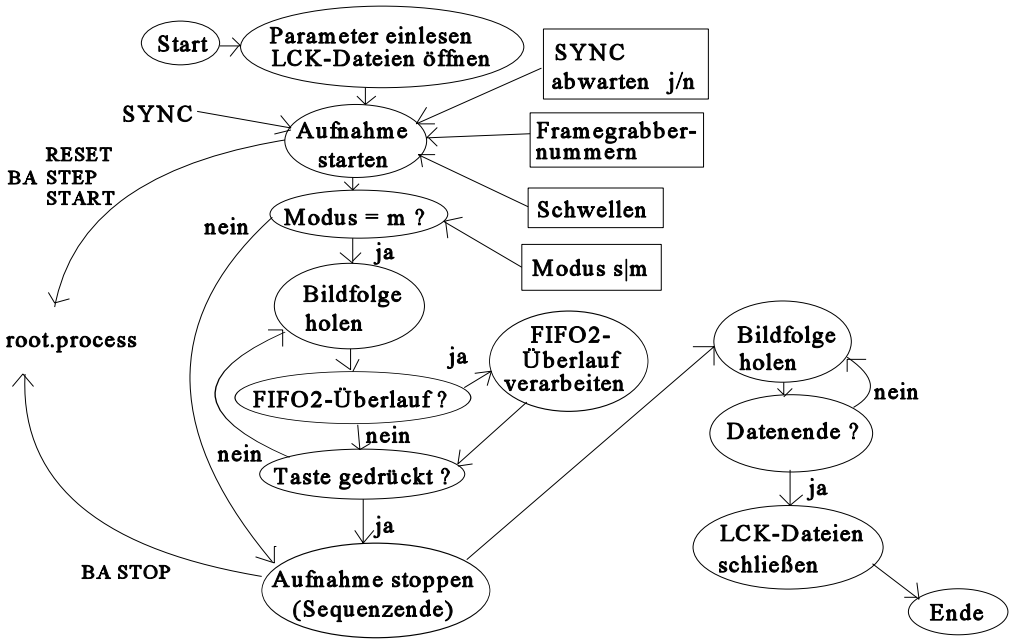

Bild 9: Prozesse in FRGRAB

Die Prozesse in FRGRAB sind in der C-Quelldatei FRGRAB.c enthalten.

Nach Programmstart werden die Parameter der Kommandozeile gelesen (vgl. Abschnitt 3.2.1) und die LCK-Dateien für die Bilddaten geöffnet. Die Aufnahme wird gestartet (ggf. nach Abwarten des Synchronisationsimpulses SYNC); im Einzelbildmodus wird die Aufnahme sofort angehalten. Für die Aufnahme einer Bildfolge werden Bilddaten aus dem Transputernetzwerk geholt. Bei einem Überlauf in FIFO2 wird eine Überlaufverarbeitung aufgerufen. Bei einer Einzelbildaufnahme wird nicht auf einen Überlauf in FIFO2 getestet. Die FIFO2 sind groß genug, um ein Einzelbild mit 100 Teilchen zu speichern. Nach dem Anhalten werden die restlichen Bilddaten bis zum Datenende aus dem Transputernetzwerk gelesen. Die LCK-Dateien werden geschlossen, FRGRAB wird verlassen.

3.3.5.1 Der Prozeß “FRGRAB/Parameter einlesen und LCK-Dateien öffnen”

Dieser Prozeß holt mit dem Start von FRGRAB die Parameter von der Kommandozeile und öffnet die LCK-Dateien, die in der Framegrabber-Information bezeichnet sind. Der Prozeß “FRGRAB/Parameter einlesen und LCK-Dateien öffnen” teilt sich in die Unterprozesse nach Bild 10 auf.

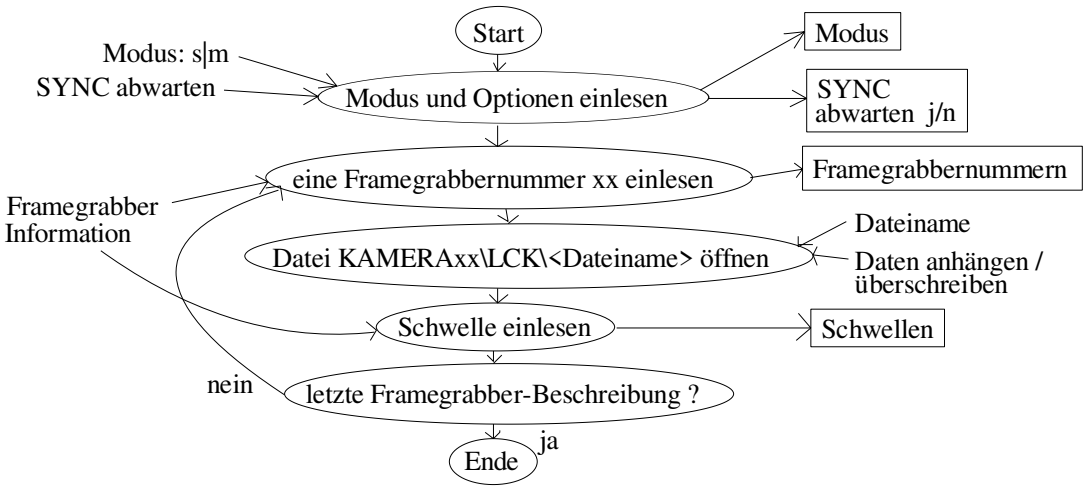

Bild 10: Der Prozeß “FRGRAB/Parameter einlesen und LCK-Dateien öffnen”

Der Modus, der Dateiname, das Zeichen, das angibt, ob die Daten an bestehende Daten angehängt werden oder bestehende Daten überschrieben werden und der Schalter für das Abwarten des Synchronimpulses werden direkt von der Kommandozeile eingelesen. Die Datei mit der Framegrabber-Information wird geöffnet und zeilenweise gelesen. Aus Framegrabber-nummer und Dateinamen (VERMESS.LCK, UFER.LCK oder <Modellname>.LCK) wird automatisch der Pfad der LCK-Datei gebildet. Die LCK-Datei wird geöffnet. Die Schwelle wird eingelesen und in FRGRAB lokal gespeichert.

3.3.5.2 Der Prozeß “FRGRAB/Bildfolge holen”

Dieser Prozeß sendet die Betriebsart BF an das Transputernetzwerk. Ein Datenblock aus dem Netzwerk wird entgegengenommen und in die passende LCK-Datei verteilt. Ein Überlauf in FIFO2 und das Datenende werden an den aufrufenden Prozeß zurückgemeldet. Der Prozeß “FRGRAB/Bildfolge holen” teilt sich in die Unterprozesse nach Bild 11 auf.

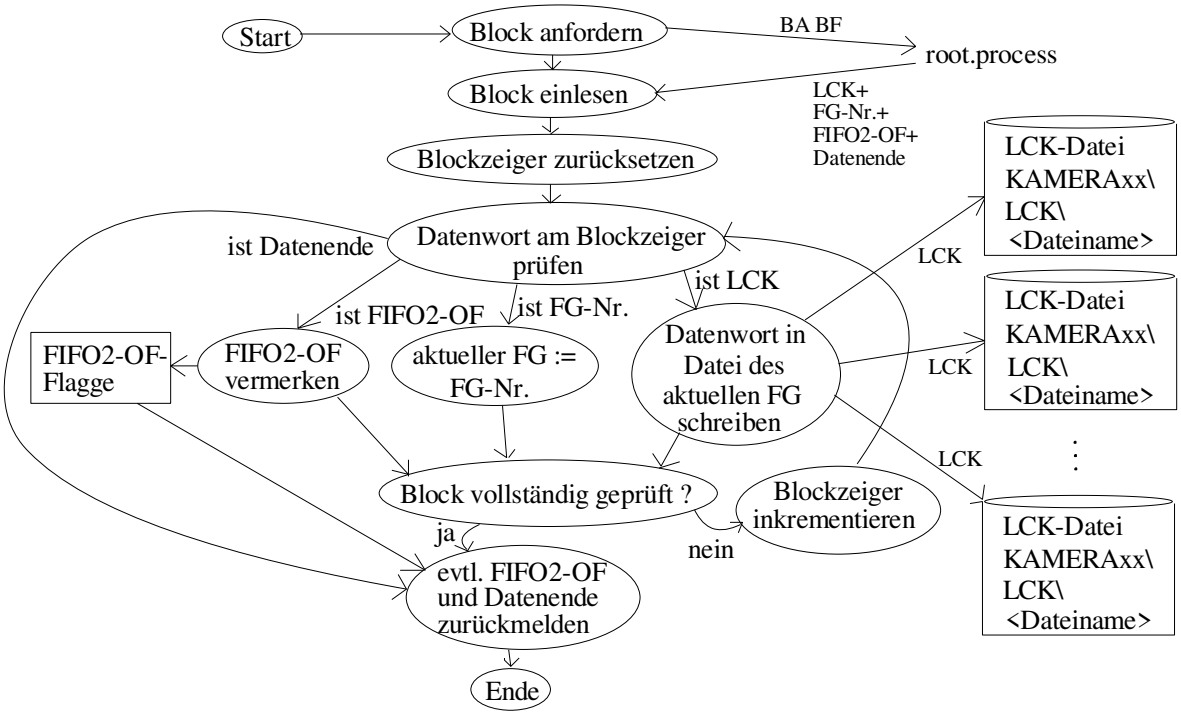

Bild 11: Der Prozeß “FRGRAB/Bildfolge holen”

Der Prozeß “FRGRAB/Bildfolge holen” merkt sich die Nummer des aktuellen Framegrabbers von Aufruf zu Aufruf. Die Daten am Anfang eines neuen Blockes werden in die Datei geschrieben, die am Ende des letzten Blockes beschrieben wurde, falls keine neue Framegrabbernummer empfangen wird. Der root-Transputer schreibt zu Beginn einer Aufnahme die Nummer des ersten abgefragten Framegrabbers in den Datenstrom. Der erste Datenblock hat so ein definiertes Ziel.

3.3.5.3 Der Prozeß “FRGRAB/FIFO2-Überlauf verarbeiten”

Dieser Prozeß hält die Aufnahme bei einem Überlauf in FIFO2 an. Nach einer Meldung an den Anwender werden die restlichen Bilddaten aus dem Netzwerk mit Hilfe von “FRGRAB/Bildfolge holen” abgeholt (bis Datenende). Anschließend wird die Aufnahme (ggf. nach Abwarten des Synchronimpulses) neu gestartet. Eine neue Sequenz wird angelegt. Der Prozeß “FRGRAB/FIFO2-Überlauf verarbeiten” teilt sich - wie in Bild 12 dargestellt - auf.

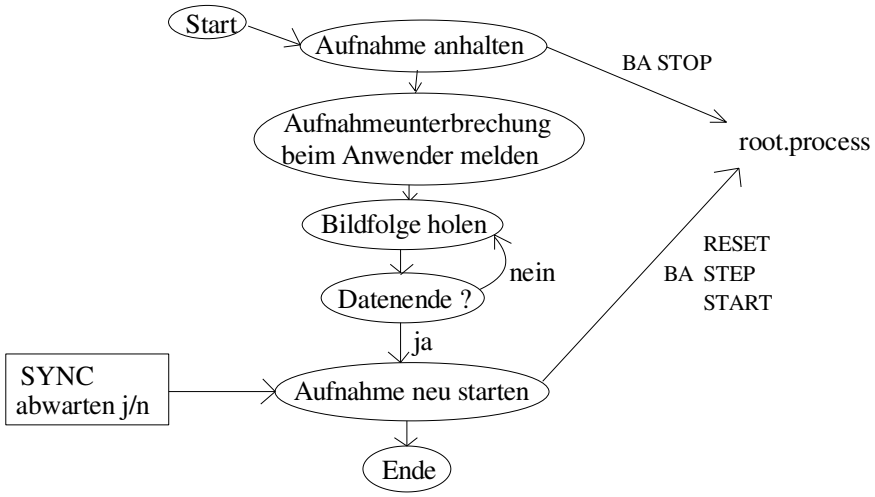

Bild 12: Der Prozeß “FRGRAB/FIFO2-Überlauf verarbeiten”

Auch im Falle eines Überlaufs in FIFO2 wird die korrekte Reihenfolge der Betriebsarten eingehalten.

Anhang A - Die einzelnen Programme des Testsystems

Alle C-Programme des Testsystems werden mit den C-Tools der DJGPP-Distribution GNU C Version 2.7.2 übersetzt. Alle occam2-Programme werden mit den occam2-Tools INMOS occam2 toolset TDS 7205 übersetzt. Das Dienstprogramm MAKE aus den C-Tools wird auch für die Steuerung der Übersetzung von occam2-Programmen eingesetzt.

1 Das Modul LINK

In dem Modul LINK sind die Funktionen zum Bedienen der Linkadapterkarte zusammengestellt. Dieses Modul wird von allen Programmen auf dem Auswerterechner eingebunden, die über die Linkadapterkarte ohne die Hilfe des Dateiservers aus dem Dienstprogramm ISERVER mit dem Transputernetzwerk kommunizieren (FRGRAB, BASPEED, LNKSPEED).

Die Objektdatei LINK.o wird mit Hilfe des Makefiles aus der der Headerdatei LINK.h und der C-Quelldatei LINK.c erzeugt. LINK.o und LINK.h werden in die Verzeichnisse der Programme kopiert, die das Modul einbinden.

Bei der Initialisierung der Schnittstelle liest das Modul LINK die Basisadresse der Linkadapterkarte aus der Umgebungsvariablen TRANSPUTER. Ist die Basisadresse der Linkadapterkarte z.B. hexadezimal 300, so ist TRANSPUTER=#300. Ist TRANSPUTER nicht gesetzt, so wird als Basisadresse hexadezimal 150 angenommen.

Das Modul LINK auf dem Auswerterechner schreibt zu sendende Bytes in den Sendeport der Linkadapterkarte und liest empfangene Bytes aus dem Empfangsport der Linkadapterkarte.

Die Linkadapterkarte schickt aus dem Sendeport einzelne Bytes zum root-Transputer und empfängt einzelne Bytes vom root-Transputer am Empfangsport. Mit jeweils einem Statusbit zeigt die Linkadapterkarte an, ob das Byte vollständig gesendet wurde bzw. ob ein Byte vollständig angekommen ist. Das Modul LINK überprüft das zugehörige Statusbit, bevor in den Sendeport geschrieben bzw. bevor aus dem Empfangsport gelesen wird.

Die Geschwindigkeit der Kommunikation zwischen root-Transputer auf der Linkadapterkarte und Auswerterechner wird mit dem Programm LNKSPEED gemessen (Anhang A 4).

Die Geschwindigkeit der Kommunikation zwischen root-Transputer und Auswerterechner wird mit einem FastPCLink-Modul der Firma MSC erhöht. Dieses Modul wird auf die Linkadapterkarte aufgesteckt. Durch den Einsatz dieses Moduls fallen Wartezeiten beim Polling der einzelnen Bytes weg.

Die schnelle Datenübertragung des FastPCLink-Moduls wird beim Polling-Betrieb nur bedingt ausgenutzt. Eine höhere Leistung wird im DMA-Betrieb erreicht. Die hema TR1-SMD-

Karte, die als Linkadapterkarte eingesetzt wird, bietet keine DMA-Übertragung. Für die optimale Ausnutzung des FastPCLink-Moduls ist die Linkadapterkarte durch ein DMA-fähiges Modell zu ersetzen (nicht getestet).

Wird eine DMA-fähige Linkadapterkarte eingesetzt, werden die Routinen in dem Modul LINK an den DMA-Betrieb angepaßt (nicht implementiert).

2 Das System BASPEED

Wird ein Synchronimpuls vom Synchrongenerator gesendet, wird anschließend die Betriebsart START im Transputernetzwerk verteilt. Die Betriebsart START muß schnell genug an die LCAs der Framegrabber gelangen, damit die Framegrabber die Aufnahme synchron zum richtigen Bild beginnen. Die Geschwindigkeit der Betriebsartenverteilung im Netzwerk wird mit dem System BASPEED gemessen.

Das System BASPEED wird aus mehreren Dateien erzeugt:

BASPEED.c	- Programm für den Auswerterechner (in C)
LINK.o und LINK.h	- Modul und Headerdatei für die Bedienung der Linkadapterkarte
PROTOCOL.inc	- Headerdatei für Protokollvereinbarungen (in occam2)
BASPEED.pgm	- Konfigurationsdatei für die occam2-Programme
ROOT.occ	- Programm für den root-Transputer (in occam2)
T2.occ	- Programm für T2 (in occam2)
T1.occ	- Programm für T1 (in occam2)

Mit Hilfe des Makefiles werden aus diesen Dateien BASPEED.EXE (für den Auswerterechner) und BASPEED.BTL (für das Transputernetzwerk) generiert.

Für die Messung der Zeit zum Übermitteln der Betriebsarten wird der gleiche Hardwareaufbau wie für die Strömungsmessung verwendet (vgl. Abschnitt 2.1). Ein Umbau ist nicht erforderlich.

BASPEED wird mit folgenden Kommandos ausgeführt:

```
Prompt> ISERVER /sr /sc BASPEED.BTL
Prompt> BASPEED
```

Auf dem Auswerterechner läuft ein Prozeß, der zehnmal ein Byte an das Transputernetzwerk schickt und auf die Antwort des Netzwerkes wartet. Die Zeit vom Senden bis zum Empfang der Antwort wird gemessen. Die maximale Zeit für ein Byte während der zehn Versuche wird ausgegeben.

Auf dem root-Transputer läuft ein Prozeß, der auf ein Byte vom Auswerterechner wartet. Nach Empfang des Bytes wird Framegrabber eins über das Koppelfeld angewählt und ein Byte gesendet. Der Prozeß auf dem root-Transputer wartet auf ein Antwortbyte von Framegrabber eins und schickt nach Empfang des Antwortbytes ein Byte an den Auswerterechner.

Auf Transputer T2 eines Framegrabbers läuft ein Prozeß, der Bytes vom root-Transputer an T1 weiterleitet. Bytes von T1 werden zurück an den root-Transputer geschickt.

Auf Transputer T1 läuft ein Prozeß, der auf ein Byte von T2 wartet. Nach Empfang wird ein Byte als Antwort an T2 zurückgeschickt.

Das Programm auf dem Auswerterechner mißt mit Hilfe des Timerbausteins 8253 die Zeit, die ein Byte braucht, um einmal bis nach T1 und wieder zurück zu wandern (inklusive Koppelfeldschaltzeit). Bei einer Linkgeschwindigkeit von 20 MBit/s braucht ein Byte maximal ca. 54 Mikrosekunden. Darin sind acht Mikrosekunden zum Schalten des Koppelfeldes enthalten (vgl. Anhang A 3). Das Byte wandert innerhalb von 46 Mikrosekunden vom Auswerterechner zum T1 und wieder zurück. Die Datenübertragungszeit vom Auswerterechner zum T1 beträgt also 23 Mikrosekunden. Die Betriebsart START erreicht den T1 nach 31 Mikrosekunden (Koppelfeldschaltzeit plus Datenübertragungszeit).

Beim Senden der Betriebsart START wird die Betriebsart nur einmal vom Auswerterechner an den root-Transputer gesendet. Der root-Transputer wählt die einzelnen aktiven Framegrabber selbsttätig an und sendet jedem die Betriebsart START. Im Vergleich erhält der zweite Framegrabber die Betriebsart START schneller als mit BASPEED gemessen, da die Zeit für die Übertragung der Betriebsart START vom Auswerterechner zum root-Transputer entfällt.

In der Strömungsmessung laufen auf dem root-Transputer, auf T2 und auf T1 wesentlich kompliziertere Programme als bei der Messung mit BASPEED. Die Übermittlung der Betriebsarten wird sich dadurch verzögern. Die Zeit für die Betriebsartenübermittlung darf nach /M. Hiltermann/ 1,29 Millisekunden nicht überschreiten. Der Sicherheitsbereich zwischen der mit BASPEED gemessenen Übertragungszeit und der maximal zulässigen Übertragungszeit ist groß genug: selbst komplizierte Programme werden die Übermittlung nicht mehr als vierzigfach verzögern.

3 Das System NETSPEED

Sämtliche für eine Aufnahme aktiven Framegrabber werden in der Zeit eines Halbbildes (20 Millisekunden) abgefragt. Die Zeitscheibe wird durch die Zahl der Framegrabber geteilt. Der Zeitanteil eines Framegrabbers wird in eine Datenblockgröße umgerechnet. Jeder Framegrabber erhält die Datenblockgröße mitgeteilt, die er in dieser Zeitscheibe senden darf. Um aus einem Zeitscheibenanteil eine in diese Zeit passende Datenblockgröße zu berechnen, wird die Über-

tragungsgeschwindigkeit zwischen Framegrabber und root-Transputer mit dem System NETSPEED gemessen.

Das System NETSPEED wird aus mehreren Dateien erzeugt:

PROTOCOL.inc	- Headerdatei mit Protokollvereinbarungen (in occam2)
NETSPEED.pgm	- Konfigurationsdatei für die occam2-Programme
ROOT.occ	- Programm für den root-Transputer (in occam2)
T2.occ	- Programm für T2 (in occam2)
T1.occ	- Programm für T1 (in occam2)

In der Konfigurationsdatei wird die Anzahl der am Test beteiligten Framegrabber mit der Konstanten anz.FG eingestellt (Voreinstellung ist zwei). Mit Hilfe des Makefiles wird aus diesen Dateien NETSPEED.BTL generiert.

Für diese Messung wird der gleiche Hardwareaufbau wie für die Strömungsmessung verwendet (vgl. Abschnitt 2.1). Ein Umbau ist nicht erforderlich.

NETSPEED wird mit folgendem Kommando ausgeführt:

```
Prompt> ISERVER /sb NETSPEED.BTL
```

Mit der Option /sb wird die *.BTL-Datei in das Transputernetzwerk geladen. Anschließend wird automatisch ein Dateiserver auf dem Auswerterechner gestartet, der die Ausgabe des occam2-Programmes auf dem Bildschirm des Auswerterechners darstellt /INMOS/.

Transputer T1 wird für die Messung nicht verwendet. Auf T1 läuft ein Dummy-Prozeß, der nichts tut.

Auf Transputer T2 läuft ein Prozeß, der auf ein Zeichen vom root-Transputer wartet. Der T2 antwortet auf das Zeichen mit einem Datenblock mit einer für die Strömungsmessung maximalen Größe von 201 Lauflängen (100 Teilchen im Halbbild -> 200 Lauflängen + Halbbildendekennung). Der Datenblock wird in 32-Bit-Worten gesendet. Jede Lauflänge belegt 32 Bit.

Der Prozeß auf dem root-Transputer wählt die T2 über das Koppelfeld an und fordert Datenblöcke an. Die Zeit, die zum Wählen und zur Übertragung eines Datenblockes benötigt wird, wird vom root-Transputer gemessen. Jeder Framegrabber wird zehnmal angewählt und zum Senden aufgefordert. Die größte aller Übertragungszeiten wird ausgegeben.

210 Lauflängen werden von einem T2 zum root-Transputer in maximal 0,645 Millisekunden übermittelt (das Transputersystem ist auf eine Übertragungsrate von 20 MBit/s eingestellt).

Die Datenblöcke werden mit dem gleichen Protokoll gesendet, das für die Datenübertragung der Strömungsmessung benutzt wird. Die 0,645 Millisekunden enthalten auch die Zeit zum Wählen und die Zeit für die Protokollsteuerung (Übertragung der Länge des Datenblockes usw.).

Teilt man die Zeitscheibe von 20 Millisekunden unter 31 Framegrabber auf, so bleiben jedem Framegrabber 0,645 Millisekunden. Jeder Framegrabber kann die maximale Datenblockgröße in jedem Fall übermitteln.

Werden viele kleine Datenblöcke übermittelt, so ist der Overhead durch den häufigen Wählvorgang und durch die Protokollsteuerung größer als bei wenigen großen Datenblöcken. Die Übertragungsbandbreite wird schlecht ausgenutzt. Deshalb wird die Blockgröße an die Zahl der Framegrabber angepaßt. Bei wenigen Framegrabbern kann jeder Framegrabber größere Blöcke senden.

Aus den 0,645 Millisekunden für 201 Lauflängen wird nach dem Dreisatz die ungefähre Anzahl Lauflängen berechnet, die in einer Millisekunde übertragen werden können. In einer Millisekunde können rund 312 Lauflängen übertragen werden. Die Berechnung der maximalen Blockgröße für einen Framegrabber ist unkritisch, da die Übertragungsbandbreite auch bei der maximalen Zahl von Framegrabbern ausreicht. Bei der Berechnung der Lauflängen pro Millisekunde werden die (von der Blockgröße unabhängigen) Einflüsse der Wählvorgänge und der Protokollsteuerung vernachlässigt.

Die Zahl der Lauflängen pro Millisekunde wird in folgende Formel für die Blockgröße eingesetzt:

$$Blockgröße = TRUNC\left(\frac{20ms}{Anzahl\ \ Framegrabber} * Lauflängen\ \ pro\ \ ms\right) \qquad \textbf{(2)}$$

Neben der Netzgeschwindigkeit wird von NETSPEED auch die Übertragungsqualität zwischen root und T2 gemessen. Der Datenblock von T2 hat einen definierten Inhalt. Der root-Transputer überprüft den Datenblock und gibt die Zahl der fehlerhaften 32-Bit-Worte aus. Am Ende der Messung wird die Gesamtfehlerzahl ausgegeben. In den Tests traten keine Übertragungsfehler auf.

Abschließend wird von NETSPEED die Zeit gemessen, die das Koppelfeld zum Schalten einer Verbindung benötigt. Eine Verbindung wird in acht Mikrosekunden aufgebaut.

4 Das System LNKSPEED

Die Übertragungsgeschwindigkeit zwischen Linkadapterkarte und Auswerterechner wird mit dem System LNKSPEED gemessen. Für diese Messung ist das Transputernetzwerk nicht erforderlich, nur der root-Transputer auf der Linkadapterkarte ist beteiligt.

Das System LNKSPEED wird aus mehreren Dateien erzeugt:

LNKSPEED.c	- Programm für den Auswerterechner (in C)
LINK.o und LINK.h	- Modul und Headerdatei für die Bedienung der Linkadapterkarte
PROTOCOL.inc	- Headerdatei für Protokollvereinbarungen (in occam2)
LNKSPEED.pgm	- Konfigurationsdatei für die occam2-Programme
ROOT.occ	- Programm für den root-Transputer (in occam2)

Mit Hilfe des Makefiles werden aus diesen Dateien LNKSPEED.EXE (für den Auswerterechner) und LNKSPEED.BTL (für das Transputernetzwerk) generiert.

LNKSPEED wird mit folgenden Kommandos ausgeführt:

```
Prompt> ISERVER /sr /sc LNKSPEED.BTL
Prompt> LNKSPEED
```

Ein Prozeß auf dem root-Transputer sendet ständig Datenblöcke mit 4096 Bytes an den Auswerterechner. Ein Programm auf dem Auswerterechner empfängt die Datenblöcke. Die Übertragungszeit wird mit Hilfe des Timerbausteins 8253 des Auswerterechners gemessen.

Das Programm auf dem Auswerterechner mißt die Übertragungszeiten von zehn Datenblöcken. Aus den gemittelten Zeiten wird die Datenübertragungsrate berechnet.

Ohne das FastPCLink-Modul beträgt die Datenübertragungsrate ca. 1,4 MBit/s. Mit dem FastPCLink-Modul - ohne DMA-Übertragung - beträgt die Datenübertragungsrate ca. 1,8 MBit/s. Das ist keine wesentliche Verbesserung bei Kosten von etwa 800 DM. Der Einsatz eines schnelleren Rechners bringt keine spürbare Verbesserung der Datenübertragungsrate.

Die schnelle Datenübertragung des FastPCLink-Moduls wird nur im DMA-Betrieb ausgenutzt. Es stand keine DMA-fähige Linkadapterkarte zur Verfügung. Es wurde nicht getestet, wie hoch die Datenübertragungsrate im DMA-Betrieb ist.

Das Übertragen vom root-Transputer in den Dateibaum ist vorerst eine Schwachstelle.

5 Das System FRGRAB/FRAMExx

Für die Strömungsmessung werden das Programm FRGRAB.EXE auf dem Auswerterechner und eine der Dateien FRAMExx.BTL auf dem Transputernetz eingesetzt.

Das System wird aus mehreren Dateien erzeugt:

FRGRAB.c	- Programm für den Auswerterechner (in C)
LINK.o und LINK.h	- Modul und Headerdatei für die Bedienung der Linkadapterkarte
PROTOCOL.inc	- Headerdatei für Protokollvereinbarungen (in occam2)
GLOBAL.inc	- Headerdatei für globale Konstanten (in occam2)
FRAME.pgm	- unvollständige Konfigurationsdatei für die occam2-Programme
ROOT.occ	- Programm für den root-Transputer (in occam2)
T2.occ	- Programm für T2 (in occam2, für alle T2 gleich)
T1.occ	- Programm für T1 (in occam2, für alle T1 gleich)

Mit Hilfe des Makefiles wird FRGRAB.EXE (für den Auswerterechner) generiert. Die Konfigurationsdatei FRAME.pgm ist unvollständig. Die Konstante für die Anzahl der Framegrabber fehlt in FRAME.pgm. Soll eine der Dateien FRAMExx.BTL erzeugt werden, so wird eine Datei FRAMExx.pgm angelegt, die vorerst nur eine Zeile mit der Deklaration der Konstanten für die Anzahl der Framegrabber enthält. FRAME.pgm wird an die Datei FRAMExx.pgm angehängt. So entsteht eine vollständige Konfigurationsdatei FRAMExx.pgm. Mit dieser vollständigen Konfigurationsdatei wird FRAMExx.BTL erzeugt.

Die Anweisungen zum Generieren der vollständigen Konfigurationsdatei sind im Makefile enthalten. Alle FRAMExx.BTL-Dateien werden automatisch erzeugt.

Abschnitt 2.2 beschreibt das Laden des Transputernetzwerkes mit der passenden Datei FRAMExx.BTL. Abschnitt 3.2.1 beschreibt die Kommandozeilensyntax von FRGRAB.

FIFO2 (auf T2) und FIFO3 (auf dem root-Transputer) werden als Arrays implementiert. Auf diese Arrays wird von zwei Stellen aus parallel zugegriffen (Fifo auslesen/Fifo füllen). Normalerweise wird dieser parallele Zugriff auf Variablen vom occam2-Übersetzer verhindert. Mit der Übersetzeroption /N wird der occam2-Compiler angewiesen, den parallelen Zugriff zuzulassen.

Der Schreibzeiger eines Fifos wird vom Schreibprozeß verändert, der Lesezeiger eines Fifos wird vom Leseprozeß verändert. Hier gibt es keine Probleme bezüglich eines gleichzeitigen Schreibzugriffs auf eine Zustandsvariable des Fifos.

In der Betriebsart RESET werden Schreib- und Lesezeiger auf Null gesetzt. Um Konflikte mit Schreib- und Leseprozeß zu vermeiden, darf die Betriebsart RESET nur auftreten, wenn keine Daten für das Fifo eintreffen (Schreibprozeß steht still) und keine Daten abgefragt werden (Leseprozeß steht still). Die Synchronisation der Betriebsart RESET mit den Schreib- und Lesevorgängen ist durch die festgelegte Reihenfolge der Betriebsarten garantiert (vgl. Abschnitt 3.1.2).

6 Das System J3TEST

Mit dem System J3TEST wird die Kommunikation zwischen T1 und LCA getestet. Der Transputer T1 wird am FIFO1-Testboard mit dem Anschluß J3 angeschlossen (deshalb J3TEST). Betriebsarten für eine Einzelbildaufnahme werden vom T1 an das LCA gesendet; die Bild- und Statusdaten, die das FIFO1 an den T1 übergibt, werden weitergereicht und auf dem Bildschirm des Auswerterechners ausgegeben.

Das System J3TEST wird aus mehreren Dateien erzeugt:

J3TEST.pgm	- Konfigurationsdatei für die occam2-Programme
ROOT.occ	- Programm für den root-Transputer (in occam2)
T1.occ	- Programm für T1 (in occam2)

Mit Hilfe des Makefiles wird aus diesen Dateien J3TEST.BTL generiert.

Für J3TEST wird ein Transputer T1 mit Link 0 an Link 1 der Linkadapterkarte angeschlossen (Verbindung von LinkIn <-> LinkOut, Error, Reset und Analyse).

J3TEST wird mit folgendem Kommando ausgeführt:

```
Prompt> ISERVER /sb J3TEST.BTL
```

Mit der Option /sb wird die Datei J3TEST.BTL in das Transputernetzwerk geladen. Anschließend wird automatisch ein Dateiserver auf dem Auswerterechner gestartet, der Ein- und Ausgabeanweisungen des occam2-Programms auf dem root-Transputer auf das Dateisystem des Auswerterechners überträgt.

Der Prozeß auf T1 wartet auf Signale vom root-Transputer. Ein Signal besteht aus einem Byte beliebigen Inhaltes, das der root-Transputer an T1 überträgt. Nach dem ersten Signal wird die Betriebsart RESET an das LCA gesendet, nach dem zweiten Signal wird die Betriebsart STEP gesendet. Die Schwelle ist in T1.occ vorgegeben. Nach dem dritten Signal wird die Betriebsart START und nach einer Pause von 10 Millisekunden die Betriebsart STOP an das LCA gesendet. Das LCA legt ein Einzelbild im FIFO1 ab. Nach einem vierten Signal vom root-Transputer liest der T1 in einer Endlosschleife das Statuswort des LCAs und ein LCK-Datum aus FIFO1. Statuswort und LCK-Datum werden zum root-Transputer geschickt. Der root-Transputer zeigt die Daten auf dem Bildschirm des Auswerterechners an.

Der Prozeß auf dem root-Transputer schickt Signale an den T1 und zeigt Status- und LCK-Daten auf dem Bildschirm des Auswerterechners an. Das erste Signal wird nach Programmstart an T1 gesendet. Dieses Signal löst die Betriebsart RESET aus. Vor dem Senden der weiteren Signale (für STEP, START+STOP) wartet der Prozeß auf dem root-Transputer jeweils auf einen Tastendruck. Nach zweimaligem Tastendruck liegt also ein LCK-Bild in FIFO1. Nach einem

dritten Tastendruck empfängt der Prozeß auf dem root-Transputer in einer Schleife die Status- und LCK-Daten des LCAs. Mit einem letzten Tastendruck wird die Schleife verlassen. Der Prozeß auf dem root-Transputer wird beendet.

Der T1 ist mit seinem gemultiplexten Daten/Adreßbus an das LCA angeschlossen. Der T1 steuert die Kommunikation mit Bitkombinationen, die er als Adresse auf den Bus legt /W. Saathoff/. Eine Bitkombination wird als Konstante vereinbart. Diese Bitkombination wird einer Variablen als Adresse zugewiesen. Bei einem Schreib- oder Lesezugriff auf diese Variable legt der Transputer die Bitkombination auf den Bus.

Der Transputer verändert eine zugewiesene Variablenadresse, bevor er sie auf den Bus legt. Der Transputer invertiert Bit 29, anschließend wird die Adresse um zwei Bit nach links geschoben.

Die Bitkombinationen für die Betriebsarten werden verändert, bevor sie als Variablenadressen zugewiesen werden. Die Bitkombinationen werden so verändert, daß bei einem Variablenzugriff die richtigen Signale auf dem Bus erscheinen (vorab nach rechts schieben und Bit 29 invertieren, siehe Listing T1.occ von J3TEST).

Anhang B - Weiteres Vorgehen

1. Die Anbindung von T1 an FIFO1 wird mit dem Testprogramm J3TEST auf dem T1 getestet. Die Daten von FIFO1 und das Statuswort werden ununterbrochen an den root-Transputer gesendet und auf dem Bildschirm des Auswerterechners ausgegeben. Die Betriebsarten RESET, STEP, START und STOP werden menügesteuert von dem Programm an das LCA geschickt.

2. Die Prozesse auf T1 des Testsystems (T1/Testbild erzeugen, T1/BA verarbeiten und T1/LCK senden) werden durch Prozesse ersetzt, die Daten aus FIFO1 abholen und Betriebsarten an das LCA weitergeben. Sind diese Prozesse implementiert, ist die Programmierung des Transputernetzwerkes einschließlich des Koppelfeldes abgeschlossen.

3. Die Prozesse der Strömungsmessung werden um den Zugriff auf das Transputernetzwerk (über FRGRAB) erweitert. Der Prozeß EM wird implementiert.

Anhang C - Verwendete Komponenten

Hardware:

Auswerterechner:	PC 486DX2/66
Linkadapterkarte:	hema TR1 SMD-Karte mit root-Transputer T800, 8 MB RAM, erweitert mit einem MSC FastPCLink-Modul
Koppelfeld:	Neurotech LS32 Link-Switch-Karte
Transputer T2:	hema TR3C-Karte mit Transputer T800, 4 MB RAM
Transputer T1:	WEDA Transputer-Modulträger, bestückt mit IMS B403-Board
Pulsgenerator:	WAVETEK Pulse/Function Generator Model 191

Software:

C-Tools:	DJGPP-Distribution GNU C Version 2.7.2 (Hinweis: Die Strömungsmessung SM 2.4 wurde mit dem GNU-C-Übersetzer Version 2.6.0 übersetzt)
occam2-Tools:	INMOS occam2 toolset TDS 7205

Anhang D - Literaturverzeichnis

/H. Bernstein/ Herbert Bernstein
Hardware-Handbuch PC-XT-AT und Kompatible
2. Auflage
Markt & Technik Verlag 1988

/C. Brunzema/ Claus Brunzema
Bedienungsanleitung für die Strömungsmessung Version 2.3
Praxissemesterbericht, FH Ostfriesland, Fachbereich ET+I
1995

/H. Greiber/ Heiko Greiber
Verarbeitung einer Bildfolge zu einem Vektorfeld von Strömungsgeschwindigkeiten und Beschleunigungen mit Hilfe eines Systems paralleler Rechner
Diplomarbeit, FH Ostfriesland, Fachbereich ET+I
1991

/J. Heyen/ Jens Heyen
Simulation der Verarbeitung einer Bildfolge zu einem Vektorfeld von Strömungsgeschwindigkeiten
Diplomarbeit, FH Ostfriesland, Fachbereich ET+I
1991

/M. Hiltermann/ Matthias Hiltermann
(Titel steht noch nicht fest)
Diplomarbeit, FH Ostfriesland, Fachbereich ET+I
1996

/INMOS/ INMOS
occam2 toolset user manual
1991

/W. Rülander/ Wilfried Rülander
Messen von Oberflächenströmungen
Diplomarbeit, FH Ostfriesland, Fachbereich ET+I
1995

/W. Saathoff/ Werner Saathoff
Betriebsarten zum Steuern von netzwerkfähigen Framegrabbern
Diplomarbeit, FH Ostfriesland, Fachbereich ET+I
1995

/D. Siemens/ Dieter Siemens
Messung von Oberflächenströmungen (SM)
Dokumentation 1.0, FH Ostfriesland, Fachbereich ET+I
1992

Anhang E - Das System SMVIDEO für die Strömungsmessung

Bisher wurden Bildfolgen der Strömungsmessung von Hand gesteuert von einem Videorecorder auf den Auswerterechner übertragen. Der Videorecorder wird in den Standbildmodus geschaltet, das Grabben des Bildes mit der ScreenMachine II wird durch einen Tastendruck ausgelöst. Anschließend wird der Videorecorder mit Hilfe des Jog/Shuttle-Rades auf das nächste Bild gestellt, das Grabben wird per Tastendruck ausgelöst usw. Dieses Verfahren ist für eine größere Bildfolge anstrengend und zeitraubend. Die Bilder wurden im Format PCX abgelegt. Dieses Format ist im Vergleich zum Format LCK sehr speicherplatzaufwendig.

Mit Hilfe der COMO-ControlX lite-Box kann der Videorecorder vom Auswerterechner angesteuert werden (vgl. die Beschreibung in /M. Hiltermann/). Die Batchdatei SMVIDEO holt beliebig viele Bilder von einem Videoband auf den Auswerterechner, indem in einer Schleife die ScreenMachine II zum Grabben eines Bildes angestoßen wird. Nachdem das Bild zum Auswerterechner übertragen ist, wird der Videorecorder automatisch um ein Bild weiterbewegt. Anschließend wird das nächste Bild gegrabbt usw. Die Bilder werden von SMVIDEO.BAT im Format LCK abgelegt. Die platzraubenden Zwischenformate PCX und EBM werden nicht auf Platte gehalten.

Installation

Die ControlX-Box wird mit ihrer RS232-Schnittstelle mit der zweiten seriellen Schnittstelle (COM2) des Auswerterechners verbunden. Die PLAYER-Schnittstelle der ControlX-Box wird mit der LANC-Schnittstelle des Videorecorders verbunden. Die ControlX-Box wird mit dem Steckernetzteil über die POWER-Buchse mit Strom versorgt.

Der Videorecorder wird als LANC-Slave konfiguriert. Mit der Taste MENU wird das Hauptmenü des Videorecorders angewählt. Mit den Cursortasten wird SET UP MENU angewählt, mit der Taste EXECUTE gelangt man in das Setup-Menü. Im Setup-Menü (Page 1) wird mit den Cursortasten der LANC MODE S angewählt. Mit der Taste EXECUTE wird das Menü verlassen.

Die ControlX-Box wird vor dem Aufruf von SMVIDEO mit dem passenden Treiber für den Videorecorder SONY EV-9000E PAL geladen. Dies übernimmt das Windows-Programm ControlX, das mit der ControlX-Box mitgeliefert wird. Das Programm ControlX wird gemäß der Installationsanleitung installiert. Der Menüpunkt “Datei/Hardware konfigurieren” wird aufgerufen. Für die Bauart von Player 1 wird ControlX S007 eingestellt. Für die Schnittstelle von Player 1 wird COM2 eingestellt.

SMVIDEO

Bevor SMVIDEO benutzt werden kann, wird die ControlX-Box mit dem passenden Treiber geladen. Windows und das Programm ControlX werden gestartet. Nach dem Start von ControlX zeigt ein Balken den Ladevorgang an. Ist der Ladevorgang beendet, werden das ControlX-

Programm und Windows verlassen. Der Treiber bleibt bis zum Abschalten der Stromversorgung in der ControlX-Box, das Programm ControlX muß selbst nach einem Reset des Auswerterechners nicht erneut aufgerufen werden.

Zum automatischen Grabben der Videobilder wird die Strömungsmessung gestartet und ein Modell vorbereitet (evtl. Maskenbild erstellen, siehe /C. Brunzema/). Das Videoband wird an die Position des ersten Bildes der Bildfolge gespult, der Videorecorder wird in den Standbildmodus geschaltet (PAUSE). Anschließend wird SMVIDEO aufgerufen:

Prompt> SMVIDEO <Kameranummer> <Bildanzahl>

Die (zweistellige) Kameranummer beschreibt die Kamera, in deren LCK-Verzeichnis die Bildfolge abgelegt wird. Die Bildanzahl gibt an, wieviele Bilder fortlaufend automatisch gegrabbt werden sollen. In einer Sekunde fallen 25, in einer Minute 1500 und in den maximal 15 Minuten der Strömungsmessung 22500 Bilder an. SMVIDEO legt die Bilder im Format LCK ab. Ist der Schalter maskiere_ebm in der Datei <Modell>.SM auf on geschaltet und liegt ein Maskenbild im Format PCX in DATA_SRC\<Modell>\KAMERA<Kameranummer>\PCX vor, so werden alle Bilder mit der Maske maskiert. Steht der Schalter append_lck in der Datei <Modell>.SM auf on, so hängt SMVIDEO die neuen Bilder an eine bereits bestehende LCK-Bildfolge an. Steht append_lck auf off, so werden evtl. vorhandene Bildfolgen überschrieben. Zum Grabben von beispielsweise 25 Bildern braucht SMVIDEO etwa 2 Minuten und 55 Sekunden. Bildfolgen können “über Nacht” gegrabbt und abgelegt werden. Sind alle Bilder gegrabbt, schaltet SMVIDEO den Videorecorder automatisch auf STOP.

Der Videorecorder braucht nur kurze Zeit (weniger als eine halbe Sekunde), um das Videoband um ein Bild weiterzuschalten. In der jetzigen Version von SMVIDEO wird direkt nach dem Grabben des Bildes der Befehl zum Weiterschalten an den Videorecorder geschickt. Während der Videorecorder den Vorlauf um ein Bild betätigt, wird das vorausgehende Bild, das die ScreenMachine II im Format FLM liefert, über die Formate PCX und EBM in das Format LCK umgerechnet. Die Bilder werden bildweise zu einem LCK-Film zusammengefügt. Ist die Umrechnung beendet, wird sofort das nächste Bild gegrabbt. Die Umrechnung dauert wesentlich länger (ca. vier bis sieben Sekunden, je nach Typ des PC, Cache und Ramdisk) als das Weiterschalten des Videorecorders. Wird ein wesentlich schnellerer Rechner für den Auswerterechner eingesetzt, kann es zu Synchronisationsproblemen kommen. Bei einem PCI-586-100 (Asus TP4-Board) ist das bei weitem nicht der Fall. Falls das Umrechnen der Bilder schneller abläuft als das Weiterschalten der Bilder, wird ein Bild zweimal gegrabbt. Dies würde die Messung verfälschen. Bei neuen, sehr schnellen Rechnern läßt sich das an wenigen Bildern prüfen. Der Editor SMLCH bewegt die Bildfolgen mit jedem Schritt sichtbar vorwärts. Zwei gleiche Bilder fallen sofort auf.

Mit der ControlX-Box kann der Timecode des Videobandes gelesen werden. Mit dem Timecode kann evtl. kontrolliert werden, ob der Videorecorder das nächste Bild bereits angefahren hat (dieses Verfahren ist nicht getestet). Der Timecode muß für diesen Bedarfsfall auf allen Videobändern vollständig aufgespielt sein. Die Timecode-Synchronisation ist vorerst nicht implementiert.

Anhang F - Programmlistings

1 Programmlistings für das Modul LINK

- MAKEFILE
- LINK.h
- LINK.c

```
#-------------------------------------------------------------------------
#- MAKEFILE
#-
#- Modul zur Bedienung der Linkadpterkarte (LINK)
#- von Claus Brunzema
#- 28.3.96  Version 1.0
#- Makefile
#-------------------------------------------------------------------------

.PHONY:         all
all :           LINK.o

LINK.o :        LINK.c LINK.h
                gcc -c -O3 LINK.c -o LINK.o
```

```
/*-------------------------------------------------------------------------
  - LINK.h
  -
  - Modul zur Bedienung der Linkadapterkarte (LINK)
  - von Claus Brunzema
  - 28.3.96  Version 1.0
  - Headerfile
  -------------------------------------------------------------------------*/

#ifndef LINK_H_INCLUDED
#define LINK_H_INCLUDED

#define ADAPTER_C012          0            /* Konstanten für */
#define ADAPTER_FAST_PC_LINK  1            /* den Adaptertyp */

/*-------------------------------------------------------------------------
  - Funktion: InitLink(int Adapter)
  - Aufgabe:  Schnittstelle initialisieren
  - Hin:      int Adapter  -  Typ des Adapters, ADAPTER_C012 oder
  -                           ADAPTER_FAST_PC_LINK
  -------------------------------------------------------------------------*/
void InitLink(int Adapter);

/*-------------------------------------------------------------------------
  - Funktion: LinkOutChar(unsigned char value)
  - Aufgabe:  das Byte value an die Linkschnittstelle ausgeben
  - Hin:      unsigned char value  -  zu sendendes Byte
  -------------------------------------------------------------------------*/
void LinkOutChar(unsigned char value);

/*-------------------------------------------------------------------------
  - Funktion: LinkInInt(int *err)
  - Aufgabe:  ein Int (32 Bit) einlesen
  - Zurück:   unsigned int  -  eingelesenes Int
  -           int *err      -  0 -> OK, -1 -> Fehler
  -------------------------------------------------------------------------*/
unsigned int LinkInInt(int *err);

/*-------------------------------------------------------------------------
  - Funktion: LinkIn(void *buffer,unsigned int anz)
  - Aufgabe:  Datenblock der Länge anz Bytes nach buffer einlesen
  - Hin:      void *buffer      -  Bereich für den Datenblock
  -           unsigned int anz  -  Größe des Datenblockes in Bytes
  - Zurück:   int               -  0 -> OK, -1 -> Fehler
  -------------------------------------------------------------------------*/
int LinkIn(void *buffer,unsigned int anz);

/*-------------------------------------------------------------------------
  - Funktion: ExitLink(void)
  - Aufgabe:  Linkschnittstelle schliessen
  -------------------------------------------------------------------------*/
void ExitLink(void);

#endif
```

```
/*-------------------------------------------------------------------------
  - LINK.c
  -
  - Modul zur Bedienung der Linkadapterkarte (LINK)
  - Beschreibung der Funktionen in LINK.H
  - von Claus Brunzema
  - 28.3.96  Version 1.0
  - C-Funktionen
  -------------------------------------------------------------------------*/

#include <stdlib.h>
#include <pc.h>
#include "LINK.h"

int LinkBaseAddr = 0x150;           /* Default-Portadresse */

/*-------------------------------------------------------------------------*/
/* Linkschnittstelle initialisieren */
void InitLink(int Adapter)
{  char *p;
   int tmp;

   p=getenv("TRANSPUTER");
   if(p)
   {  if(p[0]=='#')
      {  if(sscanf(&p[1],"%x",&tmp)==1)
         {  LinkBaseAddr=tmp;
         }
      }
   }
   outportb(LinkBaseAddr+0x13,0);
   if(Adapter==ADAPTER_FAST_PC_LINK)
   {  outportb(LinkBaseAddr,1);
   }
}

/*-------------------------------------------------------------------------*/
/* ein Byte ausgeben */
void LinkOutChar(unsigned char value)
{  while(!(inportb(LinkBaseAddr+0x03) & 0x01));
   outportb(LinkBaseAddr+0x01,value);
   while(!(inportb(LinkBaseAddr+0x03) & 0x01));
}

/*-------------------------------------------------------------------------*/
/* ein GNU-Int (32 Bit) einlesen */
unsigned int LinkInInt(int *err)
{  static unsigned int ret;

   if(LinkIn(&ret,sizeof(unsigned int)))
   {  *err=-1;
   }
   else
   {  *err=0;
   }
   return ret;
}

/*-------------------------------------------------------------------------*/
/* Schnittstelle schliessen */
void ExitLink(void)
{  while(!(inportb(LinkBaseAddr+0x03) & 0x01)); /* Ende der Ausgabe abwarten */
}

/*-------------------------------------------------------------------------*/
```

```
/* Datenblock einlesen */
asm ("
   .align 2
   .globl _LinkIn

_LinkIn:
   pushl %ebp
   movl  %esp,%ebp
   pushl %edi                    /* edi -> Zielpuffer */
   pushl %ecx                    /* ecx -> Datenzaehler */
   pushl %esi                    /* si  -> LinkBaseAddr */
   pushl %edx                    /* dx  -> aktuelle Portadresse */
   pushl %ebx                    /* ebx -> Anzahl der Zugriffsversuche */
   movl  8(%ebp),%edi
   movl  12(%ebp),%ecx
   cmpl  $0,%ecx
   je    ok
   movl  _LinkBaseAddr,%esi
   movw  %si,%dx

schleife:
   addw  $2,%dx
   movl  $2000000,%ebx
wait:
   dec   %ebx
   jz    error
   inb   %dx,%al
   testb $1,%al
   jz    wait
   movw  %si,%dx
   inb   %dx,%al
   movb  %al,(%edi)
   inc   %edi
   loop  schleife
ok:
   movl  $0,%eax
   jmp   ex

error:
   movl  $-1,%eax

ex:
   popl  %ebx
   popl  %edx
   popl  %esi
   popl  %ecx
   popl  %edi
   leave
   ret
");
```

2 Programmlistings für das System BASPEED

- MAKEFILE
- BASPEED.c
- PROTOCOL.inc
- BASPEED.pgm
- ROOT.occ
- T2.occ
- T1.occ

Das Modul LINK (Objektdatei LINK.o und Headerfile LINK.h) wird von dem System BASPEED eingebunden.

```
#-------------------------------------------------------------------------
#- MAKEFILE
#-
#- Geschwindigkeitsmessung der Betriebsartenverteilung (BASPEED)
#- von Claus Brunzema
#- 28.3.96  Version 1.0
#- Makefile
#-------------------------------------------------------------------------

.PHONY:         all
all :           BASPEED.EXE BASPEED.BTL

ROOT.tco :      ROOT.occ PROTOCOL.inc
                oc ROOT /T800
ROOT.lku :      ROOT.tco
                ilink ROOT.tco /T800 /f occam8.lnk
T2.tco :        T2.occ
                oc T2 /T800
T2.lku :        T2.tco
                ilink T2.tco /T800 /f occam8.lnk
T1.tco :        T1.occ
                oc T1 /T414
T1.lku :        T1.tco
                ilink T1.tco /T414 /f occama.lnk
BASPEED.cfb :   BASPEED.pgm PROTOCOL.inc ROOT.lku T1.lku T2.lku
                occonf BASPEED.pgm
BASPEED.BTL :   BASPEED.cfb
                icollect BASPEED.cfb

BASPEED.EXE :   BASPEED.c link.h link.o
                gcc BASPEED.c link.o -O3 -o BASPEED.out -lpc -lm
                strip BASPEED.out
                coff2exe BASPEED.out
                del BASPEED.out
```

```
/*-------------------------------------------------------------------------
  - BASPEED.c
  -
  - Geschwindigkeitsmessung der Betriebsartenverteilung (BASPEED)
  - von Claus Brunzema
  - 28.3.96  Version 1.0
  - Programm für den Auswerterechner
  -------------------------------------------------------------------------*/

#include <stdio.h>
#include <stdlib.h>
#include <pc.h>
#include "LINK.h"

#define ANZ_VERSUCHE 10                     /* Anzahl Versuche */
#define TICKS_PER_SEC 1193180.0             /* Anzahl Ticks pro Sekunde */

/*=========================================================================
  = Hauptprogramm
  =========================================================================*/
int main(void)
{  unsigned char antwort;                   /* Rückmeldung */
   int i;                                   /* Zählvariable */
   int lo,hi;                               /* Timerstand LO- und HI-Byte */
   double zeit;                             /* Zeitbedarf eines Versuches */
   double max;                              /* maximaler Zeitbedarf */

   printf("BASPEED: Geschwindigkeitsmessung der Betriebsartenverteilung\n\n");

   outportb(0x61,inportb(0x61) | 0x01);     /* Timer enable */
   InitLink(ADAPTER_FAST_PC_LINK);
   max=0.0;
   for(i=0;i<ANZ_VERSUCHE;i++)
   {  outportb(0x43,0xb0);                  /* Timer starten */
      outportb(0x42,0);
      outportb(0x42,0);
      LinkOutChar(1);
      LinkIn(&antwort,1);
      outportb(0x43,0x80);                  /* Timer lesen */
      lo=inportb(0x42);
      hi=inportb(0x42);
      zeit=(((double)(65536-(lo+256*hi)))*TICKS_PER_SEC)/1000000.0;
      printf("Versuch %2d: %gµs\n",i+1,zeit);
      if(zeit>max) max=zeit;
   }
   printf("maximale Zeit: %gµs\n",max);
   ExitLink();
   exit(EXIT_SUCCESS);
}
```

```
------------------------------------------------------------------------------
-- PROTOCOL.inc
--
-- Geschwindigkeitsmessung der Betriebsartenverteilung (BASPEED)
-- von Claus Brunzema
-- 28.3.96  Version 1.0
-- Headerfile mit Protokollvereinbarung
------------------------------------------------------------------------------

PROTOCOL KF.protocol IS [5]BYTE :
```

```
------------------------------------------------------------------------------
-- BASPEED.pgm
--
-- Geschwindigkeitsmessung der Betriebsartenverteilung (BASPEED)
-- von Claus Brunzema
-- 28.3.96  Version 1.0
-- Konfigurationsfile
------------------------------------------------------------------------------

#INCLUDE "PROTOCOL.inc"

-- Konstanten
VAL K IS 1024 :          -- ein Kilobyte
VAL M IS K * K :         -- ein Megabyte

-- Hardwarebeschreibung
NODE root.h :
NODE T2.h :
NODE T1.h :

EDGE KF.config.e :
EDGE KF.root.e :
EDGE KF.T2.e :

ARC hostlink.a :
ARC root.KF.config.a :
ARC root.KF.a :
ARC T2.KF.a :
ARC T2.T1.a :

ARC root.T2.boot.a :

NETWORK BASPEED.NET
  DO
    SET root.h (type, memsize := "T800",8 * M)
    SET T2.h (type, memsize := "T800",4 * M)
    SET T1.h (type, memsize := "T414",1 * M)
    CONNECT root.h[link][0] TO HOST WITH hostlink.a
    CONNECT root.h[link][1] TO T2.h[link][0] WITH root.T2.boot.a
    CONNECT root.h[link][2] TO KF.root.e WITH root.KF.a
    CONNECT root.h[link][3] TO KF.config.e WITH root.KF.config.a

    CONNECT T2.h[link][1] TO T1.h[link][0] WITH T2.T1.a
    CONNECT T2.h[link][3] TO KF.T2.e WITH T2.KF.a
:

-- Softwarebeschreibung
#USE "root.lku"
#USE "t2.lku"
#USE "t1.lku"

CHAN OF ANY host.to.root :
CHAN OF ANY root.to.host :
CHAN OF BYTE KF.to.root :
CHAN OF BYTE root.to.KF :
CHAN OF KF.protocol root.to.config :
CHAN OF BYTE T2.to.KF :
CHAN OF BYTE KF.to.T2 :
CHAN OF BYTE T1.to.T2 :
CHAN OF BYTE T2.to.T1 :

CONFIG
  PAR
    PROCESSOR root.h
      root.process (host.to.root, root.to.host, KF.to.root,
```

```
                    root.to.KF, root.to.config)
    PROCESSOR T2.h
      T2.process(T2.to.KF, KF.to.T2, T2.to.T1, T1.to.T2)
    PROCESSOR T1.h
      T1.process(T1.to.T2, T2.to.T1)
:

-- Mapping
MAPPING BASPEED.MAP
  DO
    MAP host.to.root, root.to.host ONTO hostlink.a
    MAP KF.to.root, root.to.KF ONTO root.KF.a
    MAP root.to.config ONTO root.KF.config.a
    MAP T2.to.KF, KF.to.T2 ONTO T2.KF.a
    MAP T2.to.T1, T1.to.T2 ONTO T2.T1.a
:
```

```
--------------------------------------------------------------------------
-- ROOT.occ
--
-- Geschwindigkeitsmessung der Betriebsartenverteilung (BASPEED)
-- von Claus Brunzema
-- 28.3.96  Version 1.0
-- Programm fuer den root-Transputer
--------------------------------------------------------------------------

#INCLUDE "PROTOCOL.inc"

--------------------------------------------------------------------------
-- Prozess: root.process
-- Aufgabe: wartet auf Byte vom AR, waehlt FG eins, schickt ein Byte zum
--          FG, wartet auf Antwort vom FG und schickt ein Byte zum AR
-- Hin:     host.to.root    -  Kanal von AR zum root (BA)
--          root.to.host    -  Kanal vom root zum AR (LCK)
--          KF.to.root      -  Kanal vom Link 0 des KF zum root (LCK)
--          root.to.KF      -  Kanal vom root zum Link 0 des KF (BA)
--          root.to.config  -  Kanal vom root zum ConfigLink des KF (Conf)
--------------------------------------------------------------------------
PROC root.process (CHAN OF ANY host.to.root,
                   CHAN OF ANY root.to.host,
                   CHAN OF BYTE KF.to.root,
                   CHAN OF BYTE root.to.KF,
                   CHAN OF KF.protocol root.to.config)

  [5]BYTE config.msg :                 -- Nachricht fuer C004
  BYTE dummy :                         -- Variable fuer Byte-Empfang

  SEQ
    config.msg[0] := #04 (BYTE)    -- Reset C004
    config.msg[1] := #01 (BYTE)    -- Link schalten
    config.msg[2] := #00 (BYTE)    -- von Link 0 nach ...
    config.msg[4] := #03 (BYTE)    -- Endezeichen fuer C004
    WHILE TRUE
      SEQ
        host.to.root ? dummy
        config.msg[3] := #01 (BYTE)
        root.to.config ! config.msg
        root.to.KF ! 1 (BYTE)
        KF.to.root ? dummy
        root.to.host ! 1 (BYTE)
:
```

```
------------------------------------------------------------------------------
-- T2.occ
--
-- Geschwindigkeitsmessung der Betriebsartenverteilung (BASPEED)
-- von Claus Brunzema
-- 28.3.96  Version 1.0
-- Programm fuer T2
------------------------------------------------------------------------------

------------------------------------------------------------------------------
-- Prozess: T2.process
-- Aufgabe: leitet Signale vom KF an T1 und umgekehrt
-- Hin:     T2.to.KF  -  Kanal von T2 zum KF (LCK)
--          KF.to.T2  -  Kanal vom KF zum T2 (BA)
--          T2.to.T1  -  Kanal vom T2 zum T1 (BA)
--          T1.to.T2  -  Kanal vom T1 zum T2 (LCK)
------------------------------------------------------------------------------
PROC T2.process (CHAN OF BYTE T2.to.KF,
                 CHAN OF BYTE KF.to.T2,
                 CHAN OF BYTE T2.to.T1,
                 CHAN OF BYTE T1.to.T2)

  BYTE dummy1, dummy2 :                  -- Variablen fuer Byte-Empfang

  PAR
    WHILE TRUE
      SEQ
        KF.to.T2 ? dummy1
        T2.to.T1 ! 1 (BYTE)
    WHILE TRUE
      SEQ
        T1.to.T2 ? dummy2
        T2.to.KF ! 1 (BYTE)
:
```

```
--------------------------------------------------------------------------
-- T1.occ
--
-- Geschwindigkeitsmessung der Betriebsartenverteilung (BASPEED)
-- von Claus Brunzema
-- 28.3.96  Version 1.0
-- Programm fuer T1
--------------------------------------------------------------------------

--------------------------------------------------------------------------
-- Prozess: T1.process
-- Aufgabe: Wartet auf ein Byte von T2, schicht ein Byte an T2
-- Hin:     T1.to.T2  -  Kanal von T1 nach T2 (LCK)
--          T2.to.T1  -  Kanal von T2 nach T1 (BA)
--------------------------------------------------------------------------
PROC T1.process (CHAN OF BYTE T1.to.T2,
                 CHAN OF BYTE T2.to.T1)
  BYTE dummy :                              -- Variable fuer Byte-Empfang

  WHILE TRUE
    SEQ
      T2.to.T1 ? dummy
      T1.to.T2 ! dummy
:
```

3 Programmlistings für das System NETSPEED

- MAKEFILE
- PROTOCOL.inc
- NETSPEED.pgm
- ROOT.occ
- T2.occ
- T1.occ

```
#-------------------------------------------------------------------------
#- MAKEFILE
#-
#- Netz-Geschwindigkeitsmessung (NETSPEED)
#- von Claus Brunzema
#- 28.3.96  Version 1.0
#- Makefile
#-------------------------------------------------------------------------

.PHONY:         all
all :           NETSPEED.BTL

ROOT.tco :      ROOT.occ PROTOCOL.inc
                oc ROOT /T800
ROOT.lku :      ROOT.tco
                ilink ROOT.tco hostio.lib convert.lib /T800 /f occam8.lnk
T2.tco :        T2.occ PROTOCOL.inc
                oc T2 /T800
T2.lku :        T2.tco
                ilink T2.tco /T800 /f occam8.lnk
T1.tco :        T1.occ
                oc T1 /T414
T1.lku :        T1.tco
                ilink T1.tco /T414 /f occama.lnk
NETSPEED.cfb :  NETSPEED.pgm PROTOCOL.inc ROOT.lku T1.lku T2.lku
                occonf NETSPEED.pgm
NETSPEED.BTL :  NETSPEED.cfb
                icollect NETSPEED.cfb
```

```
--------------------------------------------------------------------------------
-- PROTOCOL.inc
--
-- Netz-Geschwindigkeitsmessung (NETSPEED)
-- von Claus Brunzema
-- 28.3.96  Version 1.0
-- Headerfile mit Protokollvereinbarung
--------------------------------------------------------------------------------

PROTOCOL LCK.protocol
  CASE
    DATA; INT16::[]INT32       -- Datenblock
    NODATA                     -- keine Daten vorhanden
:

PROTOCOL KF.protocol IS [5]BYTE :
```

```
--------------------------------------------------------------------------
-- NETSPEED.pgm
--
-- Netz-Geschwindigkeitsmessung (NETSPEED)
-- von Claus Brunzema
-- 28.3.96  Version 1.0
-- Konfigurationsfile
--------------------------------------------------------------------------

#INCLUDE "hostio.inc"
#INCLUDE "PROTOCOL.inc"

-- Konstanten
VAL anz.FG IS 3 :        -- Anzahl der Framegrabber im Netzwerk
VAL K IS 1024 :          -- ein Kilobyte
VAL M IS K * K :         -- ein Megabyte

-- Hardwarebeschreibung
NODE root.h :
[anz.FG]NODE T2.h :
[anz.FG]NODE T1.h :

EDGE KF.config.e :
EDGE KF.root.e :
[anz.FG]EDGE KF.T2.e :

ARC hostlink.a :
ARC root.KF.config.a :
ARC root.KF.a :
[anz.FG]ARC T2.KF.a :
[anz.FG]ARC T2.T1.a :

ARC root.T2.boot.a :
[anz.FG-1]ARC T1.T2.boot.a :

NETWORK NETSPEED.NET
  DO
    SET root.h (type, memsize := "T800",8 * M)
    DO i=0 FOR anz.FG
      DO
        SET T2.h[i] (type, memsize := "T800",4 * M)
        SET T1.h[i] (type, memsize := "T414",1 * M)
    CONNECT root.h[link][0] TO HOST WITH hostlink.a
    CONNECT root.h[link][1] TO T2.h[0][link][0] WITH root.T2.boot.a
    CONNECT root.h[link][2] TO KF.root.e WITH root.KF.a
    CONNECT root.h[link][3] TO KF.config.e WITH root.KF.config.a
    DO i=0 FOR anz.FG
      DO
        CONNECT T2.h[i][link][1] TO T1.h[i][link][0] WITH T2.T1.a[i]
        CONNECT T2.h[i][link][3] TO KF.T2.e[i] WITH T2.KF.a[i]
    DO i=0 FOR (anz.FG-1)
      CONNECT T1.h[i][link][1] TO T2.h[i+1][link][0] WITH T1.T2.boot.a[i]
:

-- Softwarebeschreibung
#USE "root.lku"
#USE "t2.lku"
#USE "t1.lku"

CHAN OF SP app.in, app.out :
CHAN OF LCK.protocol KF.to.root :
CHAN OF ANY root.to.KF :
CHAN OF KF.protocol root.to.config :
[anz.FG]CHAN OF LCK.protocol T2.to.KF :
[anz.FG]CHAN OF ANY KF.to.T2 :
```

```
CONFIG
  PAR
    PROCESSOR root.h
      INT anz :
      SEQ
        anz:=anz.FG
        PRI PAR
          root.process (app.in, app.out, anz,KF.to.root,
                        root.to.KF, root.to.config)
          SKIP
    PAR i=0 FOR anz.FG
      PAR
        PROCESSOR T2.h[i]
          T2.process(T2.to.KF[i], KF.to.T2[i])
        PROCESSOR T1.h[i]
          T1.process()
:

-- Mapping
MAPPING NETSPEED.MAP
  DO
    MAP app.in, app.out ONTO hostlink.a
    MAP KF.to.root, root.to.KF ONTO root.KF.a
    MAP root.to.config ONTO root.KF.config.a
    DO i=0 FOR anz.FG
      DO
        MAP T2.to.KF[i], KF.to.T2[i] ONTO T2.KF.a[i]
:
```

```
--------------------------------------------------------------------------
-- ROOT.occ
--
-- Netz-Geschwindigkeitsmessung (NETSPEED)
-- von Claus Brunzema
-- 28.3.96  Version 1.0
-- Programm fuer den root-Transputer
--------------------------------------------------------------------------

#INCLUDE "hostio.inc"
#INCLUDE "PROTOCOL.inc"

VAL anz.versuche IS 10 :              -- Anzahl der Versuche

--------------------------------------------------------------------------
-- Prozess: root.process
-- Aufgabe: waehlt Framegrabber, empfaengt Datenbloecke. Die
--          Uebertragungszeit wird gemesen, die Uebertragungsfehler werden
--          gezaehlt. Die Zeit fuer eine Koppelfeldschaltung wird
--          gemessen.
-- Hin:     fs, ts          -  Kanaele zum iserver
--          anz.FG          -  Anzahl der FG
--          KF.to.root      -  Kanal von Link 0 des KF nach root
--          root.to.KF      -  Kanal von root zum Link 0 des KF
--          root.to.config  -  Kanal zum ConfigLink des KF
--------------------------------------------------------------------------
PROC root.process (CHAN OF SP fs, ts, VAL INT anz.FG,
                   CHAN OF LCK.protocol KF.to.root,
                   CHAN OF ANY root.to.KF,
                   CHAN OF KF.protocol root.to.config)
  #USE "hostio.lib"
  [5]BYTE config.msg :                -- Nachricht fuer C004
  INT start, ende :                   -- Start- und Endezeit einer Messung
  INT max.zeit :                      -- maximale Zeit
  TIMER uhr :                         -- Uhr fuer Messung
  INT16 groesse.block :                -- Blockgroesse fuer Empfang
  [201]INT32 buffer :                 -- Speicher fuer Empfang
  INT error :                         -- Fehlerzahl einer Messung
  INT ges.error :                     -- Gesamtfehlerzahl

  SEQ
    so.write.string.nl(fs,ts,"NETSPEED: Netz-Geschwindigkeitsmessung")
    so.write.nl(fs,ts)
    config.msg[0] := #04 (BYTE)     -- Reset C004
    config.msg[1] := #01 (BYTE)     -- Link schalten
    config.msg[2] := #00 (BYTE)     -- von Link 0 nach ...
    config.msg[4] := #03 (BYTE)     -- Endezeichen fuer C004
    ges.error := 0
    max.zeit := 0
    SEQ j=0 FOR anz.versuche
      SEQ i=0 FOR anz.FG
        SEQ
          config.msg[3] := BYTE (i + 1)
          uhr ? start
          root.to.config ! config.msg
          root.to.KF ! BYTE 1
          KF.to.root ? CASE
            DATA; groesse.block :: buffer
              SKIP
            NODATA
              SKIP
          uhr ? ende
          IF
            (ende MINUS start) > max.zeit
              max.zeit := (ende MINUS start)
```

```
            TRUE
              SKIP
          so.write.string(fs,ts,"Framegrabber ")
          so.write.int(fs,ts,INT (i+1),2)
          so.write.string(fs,ts,": ")
          so.write.int(fs,ts,ende MINUS start,2)
          so.write.string(fs,ts," Mikrosekunden  ")
          error := 0
          SEQ i=0 FOR 201
            IF
              buffer[i] <> (INT32 i)
                error := error + 1
              TRUE
                SKIP
          so.write.int(fs,ts,error,3)
          so.write.string.nl(fs,ts," Fehler")
          ges.error := ges.error+error
    so.write.string(fs,ts,"Gesamtfehlerzahl: ")
    so.write.int(fs,ts,ges.error,3)
    so.write.nl(fs,ts)
    so.write.string(fs,ts,"Maximale Zeit: ")
    so.write.int(fs,ts,max.zeit,5)
    so.write.string.nl(fs,ts," Mikrosekunden")
    uhr ? start
    config.msg[3] := 1 (BYTE)
    root.to.config ! config.msg
    uhr ? ende
    so.write.string(fs,ts,"Koppelfeldschaltzeit: ")
    so.write.int(fs,ts,(ende MINUS start),3)
    so.write.string.nl(fs,ts," Mikrosekunden")
    so.exit(fs,ts,sps.success)
:
```

```
------------------------------------------------------------------------------
-- T2.occ
--
-- Netz-Geschwindigkeitsmessung (NETSPEED)
-- von Claus Brunzema
-- 28.3.96  Version 1.0
-- Programm fuer T2
------------------------------------------------------------------------------

#INCLUDE "PROTOCOL.inc"

------------------------------------------------------------------------------
-- Prozess: T2.process
-- Aufgabe: schickt Datenbloecke an das Koppelfeld
-- Hin:     T2.to.KF  -  Kanal vom T2 zum KF
--          KF.to.T2  -  Kanal vom KF zum T2
------------------------------------------------------------------------------
PROC T2.process (CHAN OF LCK.protocol T2.to.KF, CHAN OF ANY KF.to.T2)
  BYTE trigger :                     -- Variable zum Warten auf Signal
  [201]INT32 testdaten :             -- Testdatenblock

  SEQ
    PAR i=0 FOR 201
      testdaten[i]:= INT32 i
    WHILE TRUE
      SEQ
        KF.to.T2 ? trigger
        T2.to.KF ! DATA; 201 (INT16) :: testdaten
:
```

```
--------------------------------------------------------------------------
-- T1.occ
--
-- Netz-Geschwindigkeitsmessung (NETSPEED)
-- von Claus Brunzema
-- 28.3.96  Version 1.0
-- Programm fuer T1
--------------------------------------------------------------------------

--------------------------------------------------------------------------
-- Prozess: T1.process
-- Aufgabe: Tut nichts. Ist nur vorhanden, um einen Hardwareumbau zu
--          vermeiden.
--------------------------------------------------------------------------
PROC T1.process ()
  SKIP
:
```

4 Programmlistings für das System LNKSPEED

- MAKEFILE
- LNKSPEED.c
- PROTOCOL.inc
- LNKSPEED.pgm
- ROOT.occ

Das Modul LINK (Objektdatei LINK.o und Headerfile LINK.h) wird von dem System LNKSPEED eingebunden.

```
#-------------------------------------------------------------------------
#- MAKEFILE
#-
#- Link-Geschwindigkeitsmessung zwischen AR und root (LNKSPEED)
#- von Claus Brunzema
#- 28.3.96  Version 1.0
#- Makefile
#-------------------------------------------------------------------------

.PHONY:         all
all :           LNKSPEED.EXE LNKSPEED.BTL

ROOT.tco :      ROOT.occ PROTOCOL.inc
                oc ROOT /T800
ROOT.lku :      ROOT.tco
                ilink ROOT.tco /T800 /f occam8.lnk
LNKSPEED.cfb :  LNKSPEED.pgm PROTOCOL.inc ROOT.lku
                occonf LNKSPEED.pgm
LNKSPEED.BTL :  LNKSPEED.cfb
                icollect LNKSPEED.cfb

LNKSPEED.EXE :  LNKSPEED.c link.h link.o
                gcc LNKSPEED.c link.o -O3 -o LNKSPEED.out -lpc -lm
                strip LNKSPEED.out
                coff2exe LNKSPEED.out
                del LNKSPEED.out
```

```
/*-------------------------------------------------------------------------
  - LNKSPEED.c
  -
  - Link-Geschwindigkeitsmessung zwischen AR und root (LNKSPEED)
  - von Claus Brunzema
  - 29.3.96  Version 1.0
  - Programm für den Auswerterechner
  -------------------------------------------------------------------------*/

#include <stdio.h>
#include <stdlib.h>
#include <pc.h>
#include "LINK.h"

#define ANZ_VERSUCHE 10                     /* Anzahl Versuche */
#define ANZ_BYTES 4096                      /* Datenblockgröße */
#define TICKS_PER_SEC 1193180.0             /* Anzahl Ticks pro Sekunde */

/*=========================================================================
  = Hauptprogramm
  =========================================================================*/
int main(void)
{  unsigned char buffer[ANZ_BYTES];         /* empfangene Daten */
   double sum;                              /* Zeitsumme aller Versuche */
   int i;                                   /* Zählvariable */
   int lo,hi;                               /* Timerstand LO- und HI-Byte */
   double zeit;                             /* Zeitbedarf eines Versuches */
   double durchschnitt;                     /* durchschnittlicher Zeitbedarf */

   printf("LNKSPEED: Messung der Datenübertragungsrate zwischen\n");
   printf("root-Transputer und Auswerterechner\n\n");

   outportb(0x61,inportb(0x61) | 0x01);     /* Timer enable */
   InitLink(ADAPTER_FAST_PC_LINK);
   sum=0;
   for(i=0;i<ANZ_VERSUCHE;i++)
   {  outportb(0x43,0xb0);                  /* Timer starten */
      outportb(0x42,0);
      outportb(0x42,0);
      LinkIn(buffer,ANZ_BYTES);
      outportb(0x43,0x80);                  /* Timer lesen */
      lo=inportb(0x42);
      hi=inportb(0x42);
      zeit=(((double)(65536-(lo+256*hi)))*TICKS_PER_SEC)/1000000.0;
      printf("Versuch %2d: %gµs\n",i+1,zeit);
      sum+=zeit;
   }
   durchschnitt=(double)sum/(double)ANZ_VERSUCHE;
   printf("Durchschnitt: %gµs für %d Bytes\n",durchschnitt,ANZ_BYTES);
   printf("Übertragungsrate: %g MBit/s\n",1.0/(durchschnitt/(ANZ_BYTES*8)));
   ExitLink();
   exit(EXIT_SUCCESS);
}
```

```
------------------------------------------------------------------------------
-- PROTOCOL.inc
--
-- Link-Geschwindigkeitsmessung zwischen AR und root (LNKSPEED)
-- von Claus Brunzema
-- 28.3.96  Version 1.0
-- Headerfile mit Protokollvereinbarung
------------------------------------------------------------------------------

PROTOCOL test.protocol IS [1024]INT32 :
```

```
--------------------------------------------------------------------------
-- LNKSPEED.pgm
--
-- Link-Geschwindigkeitsmessung zwischen AR und root (LNKSPEED)
-- von Claus Brunzema
-- 28.3.96  Version 1.0
-- Konfigurationsfile
--------------------------------------------------------------------------

#INCLUDE "PROTOCOL.inc"

-- Konstanten
VAL K IS 1024 :           -- ein Kilobyte
VAL M IS K * K :          -- ein Megabyte

-- Hardwarebeschreibung
NODE root.p :
ARC hostlink :

NETWORK LNKSPEED.NET
  DO
    SET root.p(type, memsize := "T800",8 * M)
    CONNECT root.p[link][0] TO HOST WITH hostlink
:

-- Softwarebeschreibung
#USE "root.lku"
NODE root.l :
CHAN OF test.protocol root.to.host :
CONFIG
  PROCESSOR root.l
    root.process (root.to.host)
:

-- Mapping
MAPPING LNKSPEED.MAP
  DO
    MAP root.l ONTO root.p
    MAP root.to.host ONTO hostlink
:
```

```
------------------------------------------------------------------------------
-- ROOT.occ
--
-- Link-Geschwindigkeitsmessung zwischen AR und root (LNKSPEED)
-- von Claus Brunzema
-- 28.3.96  Version 1.0
-- Programm fuer den root-Transputer
------------------------------------------------------------------------------

#INCLUDE "PROTOCOL.inc"

------------------------------------------------------------------------------
-- Prozess: root.process
-- Aufgabe: schickt Datenbloecke an den Auswerterechner
-- Hin:     root.to.host  -  Kanal zum AR
------------------------------------------------------------------------------
PROC root.process (CHAN OF test.protocol root.to.host)
  [1024]INT32 data:                          -- Dummy-Daten
  WHILE TRUE
    root.to.host ! data
:
```

5 Programmlistings für das System FRGRAB/FRAMExx

- MAKEFILE
- FRGRAB.c
- PROTOCOL.inc
- GLOBAL.inc
- FRAME.pgm
- ROOT.occ
- T2.occ
- T1.occ

Das Modul LINK (Objektdatei LINK.o und Headerfile LINK.h) wird von dem System FRGRAB/FRAMExx eingebunden.

```
#-------------------------------------------------------------------------
#- MAKEFILE
#-
#- Framegrabber-Programme (FRGRAB/FRAMExx)
#- von Claus Brunzema
#- 28.3.96  Version 1.0
#- Makefile
#-------------------------------------------------------------------------

BTLS = FRAME01.BTL FRAME02.BTL FRAME03.BTL FRAME04.BTL FRAME05.BTL \
       FRAME06.BTL FRAME07.BTL FRAME08.BTL FRAME09.BTL FRAME10.BTL \
       FRAME11.BTL FRAME12.BTL FRAME13.BTL FRAME14.BTL FRAME15.BTL \
       FRAME16.BTL FRAME17.BTL FRAME18.BTL FRAME19.BTL FRAME20.BTL \
       FRAME21.BTL FRAME22.BTL FRAME23.BTL FRAME24.BTL FRAME25.BTL \
       FRAME26.BTL FRAME27.BTL FRAME28.BTL FRAME29.BTL FRAME30.BTL \
       FRAME31.BTL


.PHONY:         all
all :           FRGRAB.EXE $(BTLS)

ROOT.tco :      ROOT.occ PROTOCOL.inc global.inc
                oc ROOT /T800 /N
ROOT.lku :      ROOT.tco
                ilink ROOT.tco /T800 /f occam8.lnk
T2.tco :        T2.occ PROTOCOL.inc global.inc
                oc T2 /T800 /N
T2.lku :        T2.tco
                ilink T2.tco /T800 /f occam8.lnk
T1.tco :        T1.occ PROTOCOL.inc global.inc
                oc T1 /T414
T1.lku :        T1.tco
                ilink T1.tco /T414 /f occama.lnk
FRAME%.cfb :    FRAME.pgm PROTOCOL.inc ROOT.lku T1.lku T2.lku
                echo VAL anz.FG IS $*: > FRAME$*.pgm
                type FRAME.pgm >> FRAME$*.pgm
                occonf FRAME$*.pgm
                del FRAME$*.pgm
FRAME%.BTL :    FRAME%.cfb
                icollect FRAME$*.cfb

FRGRAB.EXE :    FRGRAB.c link.h link.o
                gcc FRGRAB.c link.o -O3 -o FRGRAB.out -lpc
                strip FRGRAB.out
                coff2exe FRGRAB.out
                del FRGRAB.out
```

```
/*-------------------------------------------------------------------------
  - FRGRAB.c
  -
  - Programm zur Framegrabbersteuerung (FRGRAB)
  - von Claus Brunzema
  - 28.3.96  Version 1.0
  - Programm für den Auswerterechner
  -------------------------------------------------------------------------*/

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <pc.h>
#include "LINK.h"

#define MAX_KAMERAS 31        /* Konstante aus der Strömungsmessung */

/*--------------- Konstanten für das Transputernetzwerk -----------------*/
#define BA_RESET           1   /* Betriebsartencodes */
#define BA_START           2
#define BA_STOP            3
#define BA_STEP            4
#define BA_BF              5
#define MAX_BLOCK_ROOT 10000   /* max. Blockgröße vom root-Transputer */
/*-------------------------------------------------------------------------*/

#define PRN_BASE 0x378         /* Basisadresse von LPT1 */

struct { int nr;
         int schwelle;
         FILE *lck_file;
       } FG[MAX_KAMERAS];      /* FG-Daten */

FILE *FG_Information=NULL;     /* Datei mit FG-Information */
char *lck_name=NULL;           /* Dateiname der LCK-Dateien */
char mode=' ';                 /* Modus: Einzelbild / Bildfolge */
int append=0;                  /* Daten anhängen / überschreiben */
int no_sync=0;                 /* SYNC nicht abwarten */
int anz_FG=0;                  /* Anzahl Framegrabber */
int FG_index[MAX_KAMERAS+1];   /* Zeiger: FG_index[FG-Nr.] -> FG-Daten */


/*-------------------------------------------------------------------------
  - Funktion: bildfolge(int *overflow, int *datenende)
  - Aufgabe:  Holt Bilddaten mit der Betriebsart BF in die LCK-Dateien
  - Zurück:   int *overflow   -  0 -> OK, 1 -> FIFO2-OF
  -           int *datenende  -  0 -> OK, 1 -> Datenende
  -------------------------------------------------------------------------*/
void bildfolge(int *overflow, int *datenende)
{  static int akt_FG=0;                          /* aktuelle FG-Nr. */
   static int symbol_count=0;                    /* aktuelles Symbol */
   static char symbol[]={'|','/','-','\\'};      /* Symbole für Aktivität */
   static unsigned int buffer[MAX_BLOCK_ROOT];   /* Speicher für Datenblöcke */
   unsigned int size;                            /* Blockgröße */
   int i;                                        /* Zählvariable */
   int err;                                      /* Variable für Fehler */

   *overflow=0;
   *datenende=0;
   LinkOutChar(BA_BF);
   size=LinkInInt(&err);
   if(err)
   {  printf("Protokollfehler bei der Transputerkommunikation\n");
      exit(EXIT_FAILURE);
   }
```

```
   printf("%c Blockgröße: %d            \r",symbol[symbol_count],size);
   fflush(stdout);
   symbol_count=(symbol_count+1) % 4;

   if(LinkIn(buffer,size*sizeof(unsigned int)))
   {  fprintf(stderr,"Fehler bei der Transputerkommunikation\n");
      exit(EXIT_FAILURE);
   }
   for(i=0;i<size;i++)
   {  if((buffer[i] & 0x3ff00000) || (buffer[i] & 0xc0000000))
      {  /* L!=0 || HB || OF */
         if(fwrite(&buffer[i],sizeof(unsigned int),1,FG[akt_FG].lck_file)!=1)
         {  fprintf(stderr,"Fehler beim Dateischreiben\n");
            exit(EXIT_FAILURE);
         }
      }
      else
      {  if(buffer[i]>0 && buffer[i]<=MAX_KAMERAS)  /* neue FG-Nr. */
         {  akt_FG=FG_index[buffer[i]];
            if(akt_FG<0)
            {  fprintf(stderr,"ungültige Framegrabbernummer empfangen\n");
               exit(EXIT_FAILURE);
            }
         }
         else
         {  if(buffer[i]==0)                        /* FIFO2-OF */
            {  *overflow=1;
            }
            else
            {  if(buffer[i]==0x000fffff)            /* Datenende */
               {  *datenende=1;
                  break;
               }
               else
               {  fprintf(stderr,"fehlerhafte Daten empfangen\n");
                  exit(EXIT_FAILURE);
               }
            }
         }
      }
   }
}

/*-------------------------------------------------------------------------
  - Funktion: start_aufnahme(void)
  - Aufgabe:  Sendet Betriebsarten zum Starten einer Aufnahme (RESET,
  -           STEP, START)
  -------------------------------------------------------------------------*/
void start_aufnahme(void)
{  int i;

   LinkOutChar(BA_RESET);
   LinkOutChar(anz_FG);
   for(i=0;i<anz_FG;i++)
   {  LinkOutChar(FG[i].nr);
   }
   LinkOutChar(BA_STEP);
   for(i=0;i<anz_FG;i++)
   {  LinkOutChar(FG[i].schwelle);
   }
   if(!no_sync)
   {  while(inportb(PRN_BASE+1) & 0x40);   /* START-Synchronisierung */
   }
   LinkOutChar(BA_START);
}
```

```
/*=============================================================================
  = Hauptprogramm  FRGRAB
  =============================================================================*/
int main(int argc, char *argv[])
{  int i;                        /* Zählvariable */
   unsigned char pfad[255];      /* Buffer für Pfade der LCK-Dateien */
   int overflow;                 /* FIFO2-OF-Flagge */
   int datenende;                /* Datendende-Flagge */

   /* Parameter einlesen und LCK-Dateien öffnen */
   if(argc<4 || argc>6)
   {  fprintf(stderr,"Anwendung: frgrab s|m <FG-Information> ");
      fprintf(stderr,"<Dateiname> [-a] [-n]\n");
      fprintf(stderr,"s -> Single\nm -> Multi\n-a -> LCK-Daten anhängen\n");
      fprintf(stderr,"-n -> START-Synchronisierung ausschalten\n");
      exit(EXIT_FAILURE);
   }
   mode=argv[1][0];
   if(mode!='s' && mode!='m')
   {  fprintf(stderr,"Unbekannter Modus\n");
      exit(EXIT_FAILURE);
   }
   FG_Information=fopen(argv[2],"r");
   if(!FG_Information)
   {  fprintf(stderr,"Fehler beim Öffnen der FG-Information\n");
      exit(EXIT_FAILURE);
   }
   lck_name=argv[3];
   append=0;
   no_sync=0;
   for(i=4;i<argc;i++)
   {  if(argv[i][0]=='-')
      {  switch(argv[i][1])
         {  case 'a': append=1; break;
            case 'n': no_sync=1; break;
            default:  fprintf(stderr,"Unbekannte Option %s\n",argv[i]);
         }
      }
   }

   /* FG-Information einlesen */
   anz_FG=0;
   for(i=0;i<=MAX_KAMERAS;i++)
   {  FG_index[i]=-1;
   }
   while(fscanf(FG_Information,"%d",&FG[anz_FG].nr)==1)
   {  FG_index[FG[anz_FG].nr]=anz_FG;
      sprintf(pfad,"KAMERA%02d\\LCK\\%s",FG[anz_FG].nr,lck_name);
      if(append)
      {  FG[anz_FG].lck_file=fopen(pfad,"ab");
      }
      else
      {  FG[anz_FG].lck_file=fopen(pfad,"wb");
      }
      if(!FG[anz_FG].lck_file)
      {  fprintf(stderr,"Fehler beim Öffnen von %s\n",pfad);
         exit(EXIT_FAILURE);
      }
      if(fscanf(FG_Information,"%d",&FG[anz_FG].schwelle)!=1)
      {  fprintf(stderr,"Format-Fehler im Kameradatenfile\n");
         exit(EXIT_FAILURE);
      }
      if(FG[anz_FG].schwelle<16 || FG[anz_FG].schwelle>220)
      {  fprintf(stderr,"Schwelle ausserhalb des zulässigen Bereiches\n");
         exit(EXIT_FAILURE);
```

```
      }
      anz_FG++;
   }
   fclose(FG_Information);

   /* Aufnahme starten */
   InitLink(ADAPTER_FAST_PC_LINK);
   if(mode=='m')
   {  printf("Stop der Aufnahme durch einen Tastendruck\n");
   }
   start_aufnahme();
   overflow=0;
   if(mode=='s')
   {  LinkOutChar(BA_STOP);
   }
   else
   {  while(!kbhit())
      {  if(!overflow)
         {  bildfolge(&overflow,&datenende);
         }
         else
         {  /* FIFO2-Überlauf verarbeiten */
            LinkOutChar(BA_STOP);
            printf("Überlauf in FIFO2 !\nAufnahme wird gestoppt\n");
            do
            {  bildfolge(&overflow,&datenende);
            }while(!datenende);
            overflow=0;
            printf("Aufnahme wird gestartet\n");
            start_aufnahme();
         }
      }
      LinkOutChar(BA_STOP);
   }

   /* Restdaten holen */
   do
   {  bildfolge(&overflow,&datenende);
   }while(!datenende);

   ExitLink();
   for(i=0;i<anz_FG;i++) fclose(FG[i].lck_file);
   printf("Daten vollständig empfangen\n");
   exit(EXIT_SUCCESS);
}
```

```
------------------------------------------------------------------------------
-- PROTOCOL.inc
--
-- Programm zur Framegrabbersteuerung (FRAMExx)
-- von Claus Brunzema
-- 28.3.96  Version 1.0
-- Headerfile mit Protokollvereinbarung
------------------------------------------------------------------------------

PROTOCOL LCK.protocol
  CASE
    DATA; INT16::[]INT32 -- Datenblock
    NODATA               -- keine Daten vorhanden
:

PROTOCOL KF.protocol IS [5]BYTE :
```

```
--------------------------------------------------------------------------
-- GLOBAL.inc
--
-- Programm zur Framegrabbersteuerung (FRAMExx)
-- von Claus Brunzema
-- 28.3.96  Version 1.0
-- Headerfile mit globalen Konstanten
--------------------------------------------------------------------------

VAL BA.RESET IS   1 (BYTE):              -- Betriebsartencodes
VAL BA.START IS   2 (BYTE):
VAL BA.STOP  IS   3 (BYTE):
VAL BA.STEP  IS   4 (BYTE):
VAL BA.BF    IS   5 (BYTE):

VAL LCK.HB    IS #80000000 (INT32)   :   -- LCK-Format-Konstanten
VAL LCK.OF    IS #40000000 (INT32)   :
VAL LCK.SQ    IS LCK.HB BITOR LCK.OF :
VAL LCK.HI    IS #00080000 (INT32)   :
VAL FIFO2.OF  IS #00000000 (INT32)   :
VAL DATENENDE IS #000FFFFF (INT32)   :
```

```
--------------------------------------------------------------------------
-- FRAME.pgm
--
-- Programm zur Framegrabbersteuerung (FRAMExx)
-- von Claus Brunzema
-- 28.3.96  Version 1.0
-- Konfigurationsfile ohne Angabe der Framegrabberanzahl anz.FG
-- (wird von MAKE eingefuegt)
--------------------------------------------------------------------------

#INCLUDE "PROTOCOL.inc"

-- Konstanten
VAL K IS 1024 :          -- ein Kilobyte
VAL M IS K * K :         -- ein Megabyte

-- Hardwarebeschreibung
NODE root.h :
[anz.FG]NODE T2.h :
[anz.FG]NODE T1.h :

EDGE KF.config.e :
EDGE KF.root.e :
[anz.FG]EDGE KF.T2.e :

ARC hostlink.a :
ARC root.KF.config.a :
ARC root.KF.a :
[anz.FG]ARC T2.KF.a :
[anz.FG]ARC T2.T1.a :

NETWORK FRAME.NET
  ARC root.T2.boot.a :
  [anz.FG]ARC T1.T2.boot.a :       -- eigentlich werden nur anz.FG-1 gebraucht,
  DO                               -- das gibt aber Probleme bei anz.FG=1
    SET root.h (type, memsize := "T800",8 * M)
    DO i=0 FOR anz.FG
      DO
        SET T2.h[i] (type, memsize := "T800",4 * M)
        SET T1.h[i] (type, memsize := "T414",1 * M)
    CONNECT root.h[link][0] TO HOST WITH hostlink.a
    CONNECT root.h[link][1] TO T2.h[0][link][0] WITH root.T2.boot.a
    CONNECT root.h[link][2] TO KF.root.e WITH root.KF.a
    CONNECT root.h[link][3] TO KF.config.e WITH root.KF.config.a
    DO i=0 FOR anz.FG
      DO
        CONNECT T2.h[i][link][1] TO T1.h[i][link][0] WITH T2.T1.a[i]
        CONNECT T2.h[i][link][3] TO KF.T2.e[i] WITH T2.KF.a[i]
    IF
      anz.FG > 1
        DO i=0 FOR (anz.FG-1)
          CONNECT T1.h[i][link][1] TO T2.h[i+1][link][0] WITH T1.T2.boot.a[i]
      TRUE
        SKIP
:

-- Softwarebeschreibung
#USE "root.lku"
#USE "t2.lku"
#USE "t1.lku"

CHAN OF ANY host.to.root :
CHAN OF ANY root.to.host :
CHAN OF LCK.protocol KF.to.root :
CHAN OF ANY root.to.KF :
```

```
CHAN OF KF.protocol root.to.config :
[anz.FG]CHAN OF LCK.protocol T2.to.KF :
[anz.FG]CHAN OF ANY KF.to.T2 :
[anz.FG]CHAN OF INT32 T1.to.T2 :
[anz.FG]CHAN OF BYTE T2.to.T1 :

CONFIG
  PAR
    PROCESSOR root.h
      root.process (host.to.root, root.to.host, KF.to.root, root.to.KF,
                    root.to.config)
    PAR i=0 FOR anz.FG
      PAR
        PROCESSOR T2.h[i]
          T2.process(T2.to.KF[i], KF.to.T2[i],
                     T2.to.T1[i], T1.to.T2[i])
        PROCESSOR T1.h[i]
          T1.process(i+1, T1.to.T2[i], T2.to.T1[i])
:

-- Mapping
MAPPING FRAME.PGM
  DO
    MAP host.to.root, root.to.host ONTO hostlink.a
    MAP KF.to.root, root.to.KF ONTO root.KF.a
    MAP root.to.config ONTO root.KF.config.a
    DO i=0 FOR anz.FG
      DO
        MAP T2.to.KF[i], KF.to.T2[i] ONTO T2.KF.a[i]
        MAP T2.to.T1[i], T1.to.T2[i] ONTO T2.T1.a[i]
:
```

```
------------------------------------------------------------------------
-- ROOT.occ
--
-- Programm zur Framegrabbersteuerung (FRAMExx)
-- von Claus Brunzema
-- 28.3.96  Version 1.0
-- Programm fuer den root-Transputer
------------------------------------------------------------------------

#INCLUDE "PROTOCOL.inc"
#INCLUDE "GLOBAL.inc"

VAL MAX.KAMERAS IS 31 :   -- Konstante aus der Stroemungsmessung

------------------------------------------------------------------------
-- Prozess root.process
-- Aufgabe: Hauptprozess fuer den root-Transputer
-- Hin:     host.to.root    -  Kanal von AR nach root (BA)
--          root.to.host    -  Kanal von root zum AR (LCK)
--          KF.to.root      -  Kanal vom KF nach root (LCK)
--          root.to.KF      -  Kanal von root zum Koppelfeld (BA)
--          root.to.config  -  Kanal von root zum ConfigLink des KF (Conf)
------------------------------------------------------------------------
PROC root.process (CHAN OF ANY host.to.root,
                   CHAN OF ANY root.to.host,
                   CHAN OF LCK.protocol KF.to.root,
                   CHAN OF ANY root.to.KF,
                   CHAN OF KF.protocol root.to.config)

  VAL MAX.BLOCK.T2 IS 6000 :    -- absolute max. Blockgroesse von T2
  VAL MAX.BLOCK.HOST IS 10000 : -- max. Blockgroesse an AR
  VAL fifo.size IS 2050000 :    -- Groesse des FIFO3
  [fifo.size]INT32 FIFO :       -- FIFO3
  INT fifo.lesen :              -- Lesezeiger: naechstes zu lesendes Element
  INT fifo.schreiben :          -- Schreibzeiger: naechstes zu schr. Element
  BYTE anz.FG :                 -- Anzahl FG
  [MAX.KAMERAS]BYTE FG.nr :     -- Nummern der FG
  INT16 max.blockgroesse :      -- berechnete maximale Blockgroesse
  INT anz.SQ :                  -- Anz. empfangene SQ

  ----------------------------------------------------------------------
  -- Prozess: BA.manager
  -- Aufgabe: empfaengt Betriebsarten vom AR, verarbeitet BF
  -- Hin:     BA.to.KF  -  Kanal zum KF-Manager
  ----------------------------------------------------------------------
  PROC BA.manager(CHAN OF BYTE BA.to.KF)
    INT local.fifo.schreiben :   -- Zwischenspeicher fuer fifo.schreiben
    BYTE schwelle :              -- Schwelle
    BYTE BA :                    -- aktuelle Betriebsart
    BYTE command :               -- Kommando fuer neue Betriebsart
    INT anz.to.host :            -- Blockgroesse zum Auswerterechner
    WHILE TRUE
      SEQ
        host.to.root ? command
        CASE command
          BA.RESET
            SEQ
              #USE "snglmath.lib"
              BA := BA.RESET
              anz.SQ := 0
              fifo.schreiben, fifo.lesen := 0, 0
              host.to.root ? anz.FG
              max.blockgroesse := INT16 TRUNC (((20.0 (REAL32)) /
                                  (REAL32 TRUNC (INT anz.FG))) *
                                  (312.0 (REAL32)))
```

```
            IF
              max.blockgroesse > (INT16 MAX.BLOCK.T2)
                max.blockgroesse := INT16 MAX.BLOCK.T2
              TRUE
                SKIP
            SEQ i=0 FOR INT anz.FG
              host.to.root ? FG.nr[i]
            BA.to.KF ! BA.RESET
        BA.STEP
          SEQ
            BA.to.KF ! BA.STEP
            SEQ i=0 FOR INT anz.FG
              SEQ
                host.to.root ? schwelle
                BA.to.KF ! schwelle
        BA.START
          SEQ
            BA := BA.START
            BA.to.KF ! BA.START
        BA.BF
          IF
            ((BA <> BA.START) AND (BA <> BA.STOP))
              root.to.host ! 0 (INT32)
            TRUE
              SEQ
                local.fifo.schreiben := fifo.schreiben
                IF
                  fifo.lesen = local.fifo.schreiben
                    IF
                      ((BA = BA.STOP) AND (anz.SQ = (INT anz.FG)))
                        SEQ
                          root.to.host ! 1 (INT32)
                          root.to.host ! DATENENDE
                      TRUE
                        root.to.host ! 0 (INT32)
                  fifo.lesen <> local.fifo.schreiben
                    SEQ
                      IF
                        fifo.lesen < local.fifo.schreiben
                          anz.to.host := local.fifo.schreiben - fifo.lesen
                        fifo.lesen > local.fifo.schreiben
                          anz.to.host := fifo.size - fifo.lesen
                      IF
                        anz.to.host > MAX.BLOCK.HOST
                          anz.to.host := MAX.BLOCK.HOST
                        TRUE
                          SKIP
                      root.to.host ! (INT32 anz.to.host); [FIFO FROM
                                          fifo.lesen FOR anz.to.host]
                      fifo.lesen := (fifo.lesen + anz.to.host) REM fifo.size
        BA.STOP
          SEQ
            BA := BA.STOP
            BA.to.KF ! BA.STOP
        ELSE
          SKIP
:

--------------------------------------------------------------------------
-- Prozess: KF.manager
-- Aufgabe: verwaltet das KF, leitet Betriebsarten weiter, empfaengt
--          Daten und traegt sie in FIFO3 ein
-- Hin:     BA.in  -  Kanal fuer Betriebsarten vom BA-Manager
--------------------------------------------------------------------------
PROC KF.manager(CHAN OF BYTE BA.in)
```

```
INT akt.FG:                           -- aktueller FG
[5]BYTE config.msg :                  -- Nachricht fuer C004
BYTE schwelle :                       -- Schwelle
BYTE BA :                             -- aktuelle BA
INT fifo.frei :                       -- freier Platz im FIFO3
INT local.fifo.lesen :                -- Zwischenspeicher fuer fifo.lesen
INT16 anz :                           -- Blockgroesse von T2
[MAX.BLOCK.T2]INT32 buffer :          -- Buffer fuer Daten von T2

SEQ
  config.msg[0] := #04 (BYTE)     -- Reset C004
  config.msg[1] := #01 (BYTE)     -- Link schalten
  config.msg[2] := #00 (BYTE)     -- von Link 0 nach ...
  config.msg[4] := #03 (BYTE)     -- Endezeichen fuer C004
  BA := BA.RESET
  WHILE TRUE
    PRI ALT
      ---------------- Betriebsarten verarbeiten ----------------------
      BA.in ? BA
        CASE BA
          BA.RESET
            SEQ
              akt.FG := 0
              SEQ i=0 FOR INT anz.FG
                SEQ
                  config.msg[3] := FG.nr[i]
                  root.to.config ! config.msg
                  root.to.KF ! BA.RESET
                  root.to.KF ! max.blockgroesse
          BA.STEP
            SEQ i=0 FOR INT anz.FG
              SEQ
                config.msg[3] := FG.nr[i]
                root.to.config ! config.msg
                root.to.KF ! BA.STEP
                BA.in ? schwelle
                root.to.KF ! schwelle
          BA.START, BA.STOP
            SEQ i=0 FOR INT anz.FG
              SEQ
                config.msg[3] := FG.nr[i]
                root.to.config ! config.msg
                root.to.KF ! BA
          ELSE
            SKIP
      ------------------- Daten von T2 in FIFO3 ---------------------
      ((BA = BA.START) OR (BA = BA.STOP)) & SKIP
        SEQ
          local.fifo.lesen := fifo.lesen
          IF
            local.fifo.lesen < fifo.schreiben
              fifo.frei := (fifo.size - fifo.schreiben) + local.fifo.lesen
            local.fifo.lesen > fifo.schreiben
              fifo.frei := local.fifo.lesen - fifo.schreiben
            local.fifo.lesen = fifo.schreiben
              fifo.frei := fifo.size
          IF
            ((INT max.blockgroesse) + 1) < fifo.frei
              SEQ
                config.msg[3] := FG.nr[akt.FG]
                root.to.config ! config.msg
                root.to.KF ! BA.BF
                KF.to.root ? CASE
                  DATA; anz :: buffer
                    SEQ
```

```
                          FIFO[fifo.schreiben] := INT32 FG.nr[akt.FG]
                          fifo.schreiben := (fifo.schreiben + 1) REM fifo.size
                          PAR
                            SEQ i=0 FOR INT anz
                              SEQ
                                FIFO[fifo.schreiben] := buffer[i]
                                fifo.schreiben := (fifo.schreiben + 1) REM
                                                  fifo.size
                            SEQ j=0 FOR INT anz
                              IF
                                (buffer[j] BITAND LCK.SQ) = LCK.SQ
                                  anz.SQ := anz.SQ + 1
                                TRUE
                                  SKIP
                      NODATA
                        SKIP
                TRUE
                  SKIP
              akt.FG := (akt.FG + 1) REM (INT anz.FG)
  :

  ------------------------------------------------------------------------
  --------------------- Hauptprogramm root.process -----------------------
  ------------------------------------------------------------------------
  CHAN OF BYTE BA.manager.to.KF.manager: -- Kanal BA.manager -> KF.manager

  SEQ
    fifo.lesen, fifo.schreiben := 0, 0
    PAR
      BA.manager(BA.manager.to.KF.manager)
      KF.manager(BA.manager.to.KF.manager)
:
```

```
--------------------------------------------------------------------------
-- T2.occ
--
-- Programm zur Framegrabbersteuerung (FRAMExx)
-- von Claus Brunzema
-- 28.3.96  Version 1.0
-- Programm fuer T2
--------------------------------------------------------------------------

#INCLUDE "PROTOCOL.inc"
#INCLUDE "GLOBAL.inc"

--------------------------------------------------------------------------
-- Prozess T2.process
-- Aufgabe: BA weiterleiten, FIFO2 verwalten
-- Hin:     T2.to.KF  -  Kanal vom T2 zum KF (LCK)
--          KF.to.T2  -  Kanal vom KF zum T2 (BA)
--          T2.to.T1  -  Kanal vom T2 zum T1 (BA)
--          T1.to.T2  -  Kanal vom T1 zum T2 (LCK)
--------------------------------------------------------------------------
PROC T2.process (CHAN OF LCK.protocol T2.to.KF,
                 CHAN OF ANY KF.to.T2,
                 CHAN OF BYTE T2.to.T1,
                 CHAN OF INT32 T1.to.T2)

  VAL fifo.size IS 1000000 : -- Groesse von FIFO2
  [fifo.size]INT32 FIFO :    -- FIFO2
  INT fifo.lesen :           -- Lesezeiger: naechstes zu lesendes Element
  INT fifo.schreiben :       -- Schreibzeiger: naechstes zu schreibendes Elem.

  BOOL reset.flag :          -- Reset-Flagge
  BOOL overflow :            -- Ueberlauf-Flagge
  INT32 HI :                 -- HI-Bit des laufenden Bildes

  --------------------------------------------------------------------------
  -- Prozess BA.manager
  -- Aufgabe: Betriebsarten verarbeiten
  --------------------------------------------------------------------------
  PROC BA.manager()
    BYTE BA :                    -- empfangene Betriebsart
    BYTE schwelle :              -- empfangene Schwelle
    INT16 max.blockgroesse :     -- Maximale Blockgroesse an root
    INT anz.to.root :            -- Blockgroesse an root
    INT local.fifo.schreiben :   -- Zwischenspeicher fuer fifo.schreiben
    WHILE TRUE
      SEQ
        KF.to.T2 ? BA
        CASE BA
          BA.RESET
            SEQ
              reset.flag := TRUE
              overflow := FALSE
              HI := LCK.HI
              fifo.lesen, fifo.schreiben := 0, 0
              T2.to.T1 ! BA.RESET
              KF.to.T2 ? max.blockgroesse
          BA.STEP
            SEQ
              KF.to.T2 ? schwelle
              T2.to.T1 ! BA.STEP
              T2.to.T1 ! schwelle
          BA.START
            SEQ
              reset.flag := FALSE
              T2.to.T1 ! BA.START
```

```
        BA.BF
          IF
            reset.flag = TRUE
              T2.to.KF ! NODATA
            reset.flag = FALSE
              SEQ
                local.fifo.schreiben := fifo.schreiben
                IF
                  local.fifo.schreiben <> fifo.lesen
                    SEQ
                      IF
                        fifo.lesen < local.fifo.schreiben
                          anz.to.root := (local.fifo.schreiben - fifo.lesen)
                        fifo.lesen > local.fifo.schreiben
                          anz.to.root := (fifo.size - fifo.lesen)
                      IF
                        anz.to.root > (INT max.blockgroesse)
                          anz.to.root := (INT max.blockgroesse)
                        TRUE
                          SKIP
                      T2.to.KF ! DATA; (INT16 anz.to.root) :: [FIFO FROM
                                                 fifo.lesen FOR anz.to.root]
                      fifo.lesen := (fifo.lesen + anz.to.root) REM
                                    fifo.size
                  local.fifo.schreiben = fifo.lesen
                    T2.to.KF ! NODATA
            TRUE
              SKIP
        BA.STOP
          T2.to.T1 ! BA.STOP
        ELSE
          SKIP
:

--------------------------------------------------------------------------
-- Prozess T1.manager
-- Aufgabe: Daten vom T1 empfangen und in FIFO2 eintragen
--------------------------------------------------------------------------
PROC T1.manager()
  INT local.fifo.lesen :  -- Zwischenspeicher fuer fifo.lesen
  INT fifo.frei :         -- freier Platz im FIFO2
  INT32 data :            -- LCK-Datum von T1
  WHILE TRUE
    SEQ
      T1.to.T2 ? data
      IF
        reset.flag = FALSE
          SEQ
            local.fifo.lesen := fifo.lesen
            IF
              local.fifo.lesen < fifo.schreiben
                fifo.frei := (fifo.size - fifo.schreiben) + local.fifo.lesen
              local.fifo.lesen > fifo.schreiben
                fifo.frei := local.fifo.lesen - fifo.schreiben
              TRUE
                fifo.frei := fifo.size
            IF
              (fifo.frei > 5) AND (overflow = FALSE)
                SEQ
                  IF
                    fifo.frei = 6
                      SEQ
                        FIFO[fifo.schreiben] := LCK.OF
                        fifo.schreiben := (fifo.schreiben + 1) REM fifo.size
                        FIFO[fifo.schreiben] := LCK.HB BITOR HI
```

```
                        fifo.schreiben := (fifo.schreiben + 1) REM fifo.size
                        IF
                          HI <> 0 (INT32)
                            SEQ
                              FIFO[fifo.schreiben] := LCK.HB      -- HB,HI=0
                              fifo.schreiben := (fifo.schreiben + 1) REM
                                                fifo.size
                          TRUE
                            SKIP
                        FIFO[fifo.schreiben] := LCK.SQ
                        fifo.schreiben := (fifo.schreiben + 1) REM fifo.size
                        FIFO[fifo.schreiben] := FIFO2.OF
                        fifo.schreiben := (fifo.schreiben + 1) REM fifo.size
                        overflow := TRUE
                    TRUE
                      SEQ
                        FIFO[fifo.schreiben] := data
                        fifo.schreiben := (fifo.schreiben + 1) REM fifo.size
                        IF
                          ((data BITAND LCK.SQ) >< LCK.HB) = 0 (INT32)
                            HI := (data BITAND LCK.HI) >< LCK.HI
                          TRUE
                            SKIP
            TRUE
              SKIP
      TRUE
        SKIP
:

--------------------------------------------------------------------------
------------------- Hauptprogramm T2.process -----------------------------
--------------------------------------------------------------------------
SEQ
  reset.flag := TRUE
  overflow := FALSE
  fifo.lesen, fifo.schreiben := 0, 0
  PAR
    BA.manager()
    T1.manager()
:
```

```
--------------------------------------------------------------------------
-- T1.occ
--
-- Programm zur Framegrabbersteuerung (FRAMExx)
-- von Claus Brunzema
-- 28.3.96  Version 1.0
-- Programm fuer T1
--------------------------------------------------------------------------

#INCLUDE "PROTOCOL.inc"
#INCLUDE "GLOBAL.inc"

--------------------------------------------------------------------------
-- Prozess: T1.process
-- Aufgabe: bildet Testbild, sendet Testbild
-- Hin:     nr        -  Nummer des FG
--          T1.to.T2  -  Kanal von T1 nach T2 (LCK-Daten)
--          T2.to.T2  -  Kanal von T2 nach T1 (BA)
--------------------------------------------------------------------------
PROC T1.process (VAL INT nr, CHAN OF INT32 T1.to.T2, CHAN OF BYTE T2.to.T1)
  VAL lck.size IS 402 :                 -- Groesse des Bildes
  VAL bildbreite IS 760 (INT32):        -- Breite des Bildes in Pixeln
  VAL ZEIT.VOLLBILD IS 625 :            -- Zeit fuer ein Vollbild
                                        --   Einheit: 64 Mikrosekunden

  BYTE BA :                             -- aktuelle Betriebsart
  BYTE schwelle :                       -- Schwelle
  [lck.size]INT32 LCK.bild:             -- Testbild
  INT zeile.a,zeile.b,block.a,block.b:  -- Zaehler zur Tesbilderzeugung
  INT akt.lck :                         -- Zeiger auf naechstes LCK
  TIMER uhr :                           -- Uhr zum Abwarten Bildende
  INT start.bild :                      -- Startzeit des Bildes
  INT jetzt :                           -- momentane Zeit

  SEQ
    -- Testbild erzeugen
    zeile.a,zeile.b,block.a,block.b := 0, 0, 0, 0
    PAR
      SEQ i=0 FOR ((lck.size/2)-1)
        SEQ
          LCK.bild[i] := INT32 (block.a * 30)
          LCK.bild[i] := LCK.bild[i] BITOR ((INT32 zeile.a) << 10)
          LCK.bild[i] := LCK.bild[i] BITOR ((10 (INT32)) << 20)
          block.a := block.a + 1
          IF
            block.a >= nr
              PAR
                block.a := 0
                zeile.a := zeile.a + 1
            TRUE
              SKIP
      SEQ j=(lck.size/2) FOR ((lck.size/2)-1)
        SEQ
          LCK.bild[j] := INT32 (block.b * 30)
          LCK.bild[j] := LCK.bild[j] BITOR ((INT32 (zeile.b+100)) << 10)
          LCK.bild[j] := LCK.bild[j] BITOR ((10 (INT32)) << 20)
          block.b := block.b + 1
          IF
            block.b >= nr
              PAR
                block.b := 0
                zeile.b := zeile.b + 1
            TRUE
              SKIP
      LCK.bild[(lck.size/2)-1] := LCK.HB BITOR LCK.HI
```

```
      LCK.bild[lck.size-1] := LCK.HB

    WHILE TRUE
      PRI ALT
        T2.to.T1 ? BA
          CASE BA                                -- BA verarbeiten
            BA.STEP
              T2.to.T1 ? schwelle
            BA.START
              SEQ
                uhr ? start.bild
                T1.to.T2 ! LCK.bild[0]
                akt.lck := 1
            ELSE
              SKIP
        TRUE & SKIP                              -- LCK senden
          CASE BA
            BA.START
              SEQ
                T1.to.T2 ! LCK.bild[akt.lck]
                akt.lck := (akt.lck + 1) REM lck.size
                IF
                  akt.lck = 0
                    PAR
                      PAR i=0 FOR lck.size
                        LCK.bild[i] := (LCK.bild[i] BITAND
                                        #FFFFFC00 (INT32)) BITOR
                                        (((LCK.bild[i] BITAND
                                        #000003FF (INT32)) +
                                        3 (INT32)) REM bildbreite)
                      SEQ
                        uhr ? jetzt
                        uhr ? AFTER jetzt PLUS (ZEIT.VOLLBILD -
                                                (jetzt MINUS start.bild))
                        uhr ? start.bild
                  TRUE
                    SKIP
            BA.STOP
              IF
                akt.lck > 0
                  SEQ
                    T1.to.T2 ! LCK.bild[akt.lck]
                    akt.lck := (akt.lck + 1) REM lck.size
                akt.lck = 0
                  SEQ
                    T1.to.T2 ! LCK.SQ
                    akt.lck := -1
                TRUE
                  SKIP
            ELSE
              SKIP
:
```

6 Programmlistings für das System J3TEST

- MAKEFILE
- J3TEST.pgm
- ROOT.occ
- T1.occ

```
#-------------------------------------------------------------------------
#- MAKEFILE
#-
#- Test der Kommunikation LCA - T1 (J3TEST)
#- von Claus Brunzema
#- 30.4.96 Version 1.0
#- Makefile
#-------------------------------------------------------------------------

.PHONY:          all
all :           J3TEST.BTL

ROOT.tco :      ROOT.occ
                oc ROOT /T800
ROOT.lku :      ROOT.tco
                ilink ROOT.tco hostio.lib convert.lib /T800 /f occam8.lnk
T1.tco :        T1.occ
                oc T1 /T800
T1.lku :        T1.tco
                ilink T1.tco /T800 /f occam8.lnk
J3TEST.cfb :    J3TEST.pgm ROOT.lku T1.lku
                occonf J3TEST.pgm
J3TEST.BTL :    J3TEST.cfb
                icollect J3TEST.cfb
```

```
--------------------------------------------------------------------------
-- J3TEST.pgm
--
-- Test der Kommunikation LCA - T1 (J3TEST)
-- von Claus Brunzema
-- 30.4.96 Version 1.0
-- Konfigurationsfile
--------------------------------------------------------------------------

#INCLUDE "hostio.inc"

-- Konstanten
VAL K IS 1024 :         -- ein Kilobyte
VAL M IS K * K :        -- ein Megabyte

-- Hardwarebeschreibung
NODE root.h :
NODE T1.h :

ARC hostlink.a :
ARC root.T1.boot.a :

NETWORK J3TEST.net
  DO
    SET root.h (type, memsize := "T800",8 * M)
    SET T1.h (type, memsize := "T800", 3584)
    CONNECT root.h[link][0] TO HOST WITH hostlink.a
    CONNECT root.h[link][1] TO T1.h[link][0] WITH root.T1.boot.a
:

-- Softwarebeschreibung
#USE "root.lku"
#USE "T1.lku"

CHAN OF SP app.in, app.out :
CHAN OF INT32 T1.to.root :
CHAN OF BYTE root.to.T1 :

CONFIG
  PAR
    PROCESSOR root.h
      root.process (app.in,app.out,T1.to.root,root.to.T1)
    PROCESSOR T1.h
      PRI PAR
        T1.process(T1.to.root, root.to.T1)
        SKIP
:

-- Mapping
MAPPING J3TEST.map
  DO
    MAP app.in, app.out ONTO hostlink.a
    MAP T1.to.root, root.to.T1 ONTO root.T1.boot.a
:
```

```
--------------------------------------------------------------------------
-- ROOT.occ
--
-- Test der Kommunikation LCA - T1 (J3TEST)
-- von Claus Brunzema
-- 30.4.96 Version 1.0
-- Programm fuer den root-Transputer
--------------------------------------------------------------------------

#INCLUDE "hostio.inc"

--------------------------------------------------------------------------
-- Prozess: root.process
-- Aufgabe: sendet Signale fuer die Betriebsarten an T1, empfaengt Status-
--          und Fifo-Daten.
-- Hin:     fs, ts          -  Kanaele zum iserver
--          T1.to.root      -  Kanal von T1 an root (Status- und
--                             Fifo-Daten)
--          root.to.T1      -  Kanal von root zum T1 (fuer Signale)
--------------------------------------------------------------------------


PROC root.process (CHAN OF SP fs, ts,
                   CHAN OF INT32 T1.to.root,
                   CHAN OF BYTE root.to.T1)
  #USE "hostio.lib"

  INT32 status,lck:      -- Daten von T1
  BOOL ende:             -- Ende-Flagge
  BYTE key,result:       -- fuer Warten auf Taste

  SEQ
    root.to.T1 ! 1 (BYTE)
    so.write.string.nl(fs,ts,"RESET gesendet")
    so.getkey(fs,ts,key,result)
    root.to.T1 ! 1 (BYTE)
    so.write.string.nl(fs,ts,"STEP gesendet")
    so.getkey(fs,ts,key,result)
    root.to.T1 ! 1 (BYTE)
    so.write.string.nl(fs,ts,"START+STOP gesendet")
    so.getkey(fs,ts,key,result)
    root.to.T1 ! 1 (BYTE)
    ende := FALSE
    WHILE (NOT ende)          -- Schleife fuer Anzeige von Status- und
      SEQ                     -- Fifo-Daten
        T1.to.root ? status
        T1.to.root ? lck
        so.write.string(fs,ts,"Status: ")
        so.write.hex.int32(fs,ts,status,9)
        so.write.string(fs,ts,"   LCK-Datum: x=")
        so.write.int32(fs,ts,lck BITAND (#000003FF (INT32)),3)
        so.write.string(fs,ts,"  y=")
        so.write.int32(fs,ts,((lck BITAND (#0007FC00 (INT32))) >> 10),3)
        so.write.string(fs,ts,"  l=")
        so.write.int32(fs,ts,((lck BITAND (#3FF00000 (INT32))) >> 20),3)
        so.write.string(fs,ts,"   HB=")
        IF
          (lck BITAND (#80000000 (INT32))) <> (0 (INT32))
            so.write.string(fs,ts,"1 OF=")
          TRUE
            so.write.string(fs,ts,"0 OF=")
        IF
          (lck BITAND (#40000000 (INT32))) <> (0 (INT32))
            so.write.string(fs,ts,"1 HI=")
          TRUE
```

```
            so.write.string(fs,ts,"0 HI=")
          IF
            (lck BITAND (#00080000 (INT32))) <> (0 (INT32))
              so.write.string.nl(fs,ts,"1")
            TRUE
              so.write.string.nl(fs,ts,"0")
          so.pollkey(fs,ts,key,result)
          IF
            result = spr.ok
              ende := TRUE
            TRUE
              SKIP
      so.exit(fs,ts,sps.success)
:
```

```
------------------------------------------------------------------------------
-- T1.occ
--
-- Test der Kommunikation LCA - T1 (J3TEST)
-- von Claus Brunzema
-- 30.4.96 Version 1.0
-- Programm fuer T1
------------------------------------------------------------------------------

------------------------------------------------------------------------------
-- Prozess: T1.process
-- Aufgabe: Schickt signalgesteuert Betriebsarten an das LCA,
--          empfaengt Status- und Fifo-Daten. Schickt Status- und
--          Fifo-Daten an root.
-- Hin:     T1.to.root      -  Kanal von T1 an root (Status- und
--                             Fifo-Daten)
--          root.to.T1      -  Kanal von root zum T1 (fuer Signale)
------------------------------------------------------------------------------

PROC T1.process (CHAN OF INT32 T1.to.root, CHAN OF BYTE root.to.T1)

  VAL fifo.adr    IS (#70000000 >> 2) >< #20000000:  -- noch nicht getestet
  VAL status.adr  IS (#8FFFFFFF >> 2) >< #20000000:  -- noch nicht getestet

  VAL start.adr   IS (#30000000 >> 2) >< #20000000:  -- Transputer veraendert
  VAL stop.adr    IS (#20000000 >> 2) >< #20000000:  -- Adressen: zwei nach
  VAL reset.adr   IS (#10000000 >> 2) >< #20000000:  -- rechts, Bit 29
  VAL step.adr    IS (#40000000 >> 2) >< #20000000:  -- invertieren

  INT32 fifo.data:      -- Variable fuer Fifo-Daten
  INT32 status.data:    -- Variable fuer Statusdaten
  INT32 start:          -- fuer Senden der BA START
  INT32 stop:           -- fuer Senden der BA STOP
  INT32 reset:          -- fuer Senden der BA RESET
  INT32 step:           -- fuer Senden der BA STEP

  PLACE status.data AT status.adr:
  PLACE fifo.data   AT fifo.adr:
  PLACE start       AT start.adr:
  PLACE stop        AT stop.adr:
  PLACE reset       AT reset.adr:
  PLACE step        AT step.adr:

  BYTE trigger:        -- fuer Empfang der Signale
  INT32 dummy:         -- fuer Lesezugriff auf Variable
  INT32 status:        -- Zwischenspeicher fuer Status
  INT32 fifo:          -- Zwischenspeicher fuer Fifo-Daten
  INT jetzt :          -- Zeitpunkt jetzt
  TIMER uhr :          -- zum Warten

  SEQ
    root.to.T1 ? trigger
    dummy:=reset
    root.to.T1 ? trigger
    step := #30000000 (INT32)        -- Schwelle als Konstante
    root.to.T1 ? trigger
    dummy := start
    uhr ? jetzt
    uhr ? AFTER jetzt PLUS 10000     -- Warten zwischen START und STOP:
    dummy := stop                    -- Einzelbild wird aufgenommen
    root.to.T1 ? trigger
    WHILE TRUE
      SEQ
        uhr ? jetzt
        uhr ? AFTER jetzt PLUS 50000
```

```
        status := status.data
        fifo := fifo.data
        T1.to.root ! status
        T1.to.root ! fifo
:
```