Technische Universität München
Heinz Nixdorf-Lehrstuhl für Medizinische Elektronik

Echtzeitumgebung (Hard- und Firmware-Plattform) für ein Mikroskop-basiertes „Machine-Vision“ System

Thomas Geisler

Master of Science in Electrical Engineering
(University of Tulsa, USA)

Vollständiger Abdruck der von der Fakultät für Elektrotechnik und Informationstechnik der Technischen Universität München zur Erlangung des akademischen Grades eines Doktor-Ingenieurs genehmigten Dissertation.

Vorsitzender:		Univ.-Prof. Dr.-Ing. habil. Alexander W. Koch
Prüfer der Dissertation:		
	1.	Univ.-Prof. Dr. rer. nat. habil. Bernhard Wolf
	2.	Univ.-Prof. Dr.-Ing. Georg Färber
	3.	Univ.-Prof. Dr. rer. nat. Rainer Uhl, Ludwig-Maximilians-Universität München

Die Dissertation wurde am 24.10.2006 bei der Technischen Universität München eingereicht und durch die Fakultät für Elektrotechnik und Informationstechnik am 20.02.2007 angenommen.

Danksagung

Diese Dissertation entstand als Ergebnis meiner Tätigkeit als wissenschaftlicher Mitarbeiter am BioImaging Zentrum der Ludwigs-Maximilians-Universität München bzw. als Doktorand am Heinz Nixdorf-Lehrstuhl für Medizinische Elektronik der Technischen Universität München. Teile der Arbeit wurden vom Bundesministerium für Bildung und Forschung sowie dem Freistaat Bayern im Rahmen zweier Verbundprojekte unter den Förderkennzeichen IuK 054/001 (*Konzeption und Entwicklung einer Software-Plattform für fluoreszenzmikroskopische Diagnose- und Screening-Applikationen - ESDiSA*) bzw. 13N8780 (*LiveCellScreening*) gefördert. Ich möchte allen Personen danken, die mir diese Tätigkeit ermöglichten.

Besonderen Dank schulde ich Herrn Professor Wolf, der diese Arbeit betreut hat. Ebenso danke ich Herrn Professor Färber für seine Bereitschaft, als Gutachter zu wirken. Mein besonderer Dank geht an Herrn Professor Rainer Uhl, der diese Arbeit angeregt und vor Ort betreut hat.

Allen Mitarbeitern des BioImaging Zentrums danke ich für die sehr gute Zusammenarbeit und Unterstützung sowie für das herzlich kollegiale Arbeitsklima. Weiterhin danke ich den Mitarbeitern des Heinz Nixdorf-Lehrstuhl für Medizinische Elektronik, dass sie mich als externen Doktoranden aufgenommen und in die Arbeiten des Lehrstuhls integriert haben.

Ich möchte allen Kollegen und Freunden danken, die durch viele wertvolle Diskussionen zum Gelingen dieser Arbeit beigetragen haben.

Nicht zuletzt möchte ich mich auch bei meiner Familie und speziell bei meiner Schwester Stefanie bedanken, dass sie mich auf den bisherigen Weg gebracht und diesen so eben wie möglich gemacht haben und ich jederzeit auf ihre Unterstützung zählen kann. Ein besonderer Dank gilt meiner Verlobten Daniela, die mich stets voll unterstützt und vor allem in der Schlussphase immer wieder angespornt hat.

München, im Oktober 2006

Vorwort

Abbildungen, Tabellen

Die Quellen der in Abbildungen präsentierten Graphiken bzw. der in Tabellen aufgelisteten Daten sind jeweils gegeben, falls die Inhalte aus externer Literatur übernommen wurden. Abbildungen oder Tabellen ohne Quellenangabe wurden als Teil der hier vorgestellten Arbeiten erstellt und/oder ermittelt.

Glossar

Im Anhang dieser Dissertation befindet sich u. a. ein Glossar, in dem erklärungsbedürftige Wörter aufgelistet und kurz beschrieben werden. Kenntnisse von der Bedeutung dieser Wörter ist für das Verständnis der entsprechenden Textstellen stellenweise unabdingbar. Wörter, die im Glossar aufgelistet sind, werden beim ersten Vorkommen im Text durch eine spezielle Formatierung gekennzeichnet: BEISPIELFORMATIERUNG. Um den Lesefluss nicht unnötig zu stören sind alle weiteren Vorkommen innerhalb des verbleibenden Textes nicht gesondert formatiert.

PDF-Version dieses Dokuments

Bei der Erstellung des Dokuments wurden *interne* Verknüpfungen für etwaige Querverweise vorgesehen, über die bequem zu den entsprechend verwiesenen Stellen im Dokument gewechselt werden kann. Die Beschreibung der Navigation bezieht sich auf die Verwendung des Acrobat® Readers® in der aktuellen Version (Version 7; erhältlich unter:
http://www.adobe.de/products/acrobat/readstep2.html).
Externe Verknüpfungen, wie z. B. zur soeben genannten Webseite, sollten zudem automatisch einen Web-Browser und darin die entsprechende Webseite öffnen.
Hinweis: Einige Verknüpfungen funktionieren nur ordnungsgemäß, wenn im Dialogfeld „Bearbeiten“ → “Grundeinstellungen“ im Bereich „Allgemein“ die Option „Automatisch URLs im Text erkennen“ aktiviert wurde.
Verknüpfung werden folgendermaßen aufgerufen:

- Wählen Sie das Hand-Werkzeug aus  .
- Zeigen Sie mit der Maus auf die Verknüpfung, bis der Zeiger sich in eine Hand mit einem ausgestreckten Zeigefinger verwandelt. (Die Hand ist zusätzlich mit einem „W“ gekennzeichnet, wenn es sich bei der Verknüpfung um eine Internet-Verknüpfung handelt.)
- Klicken Sie auf die Verknüpfung.

Zum Aktivieren der Verknüpfung muss jeweils direkt auf die verwiesenen Zahlen, Abkürzung, etc. geklickt werden. Für folgende Querverweise wurden interne Links vorgesehen.

- Inhaltsverzeichnis: durch Anklicken des gewünschten Abschnitts im Inhaltsverzeichnis wird automatisch zu diesem gewechselt.
- Verweise auf Abschnitte: z. B. (vgl. Abschnitt 1)
- Verweise auf Seiten: z. B. (s. Seite 1)
- Verweise auf Abbildungen: z. B. (s. Abbildung 1.1)

- Literaturangaben: z. B. [50]
- Abkürzungen: z. B. **T**echnische **U**niversität **M**ünchen (TUM) oder LMU

Interne Verknüpfungen führen unmittelbar *unter* die verknüpfte Stelle im Dokument. Je nachdem von welcher Position innerhalb des Texts zu der verknüpften Stelle gewechselt wird, kann diese unmittelbar *oberhalb* des sichtbaren Textbereichs sein, so dass der Bereich durch „scrollen“ angepasst werden muss.

Ähnlich der Navigation in einem Web-Browser kann durch Klicken auf die Schaltfläche  (befindet sich in der Statusleiste im unteren Teil des Fensters) wieder zu Ansicht des Ausgangspunkts zurück gewechselt werden. In der Statusleiste befinden sich zudem Navigationsschaltflächen zum Wechseln zur „Erste Seite“, „Vorherige Seite“, „Nächste Seite“, „Letzte Seite“, die bereits beschriebene „Vorherige Ansicht“ und eine Schaltfläche zum Wechseln zur „Nächste Ansicht“.

Inhaltsverzeichnis

Zusammenfassung

Zentrales Thema dieser Arbeit ist eine Echtzeitumgebung für vollautomatisierte Mikroskopieanwendungen im Bereich der Life Sciences, in denen es u. a. darum geht, die in lebenden Zellen und Zellverbänden (wie z. B. Gewebe) ablaufenden komplexen Vorgänge zu erfassen und zu modellieren. Um dafür einen möglichst ganzheitlichen Einblick in die intra- und interzellulären Vorgänge zu erhalten, müssen eine Vielzahl unterschiedlichster Parameter parallel sowie räumlich und zeitlich aufgelöst betrachtet werden. Neben sensorbasierten stehen hier vor allem die vielseitigen lichtmikroskopischen Messtechniken im Vordergrund.

In der vorliegenden Arbeit wird ein allgemeines elektronisches Steuerungskonzept vorgestellt, das die modulare Integration verschiedener, an sich autarker Komponenten in ein ganzheitliches System ermöglicht und gleichzeitig den zeitgenauen Betrieb aller Komponenten im Systemverbund gewährleistet. Basierend auf einer neuartigen, speziell für vollautomatisierten Betrieb konzipierten Lichtmikroskopie-Plattform erlaubt das Konzept den Aufbau modularer Mikroskop-basierter Analysesysteme, in denen alle System-Komponenten (z. B. Mikroskop, Beleuchtungseinheiten, Kamera oder Laser-Scan-Einrichtungen) miteinander synchronisiert betrieben werden. Die Modularität des Konzepts erlaubt die Integration weiterer Komponenten z. B. zur automatisierten Bestückung und Beprobung oder zur sensorischen Charakterisierung zellulärer Proben.

Zentraler Bestandteil des Steuerungskonzepts ist die Hardwaresynchronisation aller Systemkomponenten mittels einer speziell konzipierten Echtzeit-Plattform. Auf einem Raster von 34 µs ermöglicht diese die zeitlich deterministische Abarbeitung benutzerdefinierter Ablaufprotokolle. Zudem kann über die Definition entsprechender Abhängigkeiten die Ausführung von Protokollkommandos nahezu beliebig an sowohl hard- als auch firmwarebasierte Systemzustände gekoppelt werden. Anhand des Ablaufprotokolls können somit während der Laufzeit unterschiedliche Synchronisationsgeber (Master) ausgewählt werden (Multi-Master-System). Hierüber wird zeitoptimales Verhalten des Gesamtsystems mit minimalen Totzeiten erreicht. Über Protokolleigenschaften wie Schleifen und bedingte Ausführung von Protokollblöcken ist zusätzlich zum Zeitverhalten auch der Ablauf selbst beeinflussbar. Das Systemverhalten kann hierdurch auf während der Laufzeit auftretende veränderte Anforderungen adaptiert werden. Durch die zeitrichtige Abarbeitung deterministischer Abläufe einerseits und durch mannigfaltige Möglichkeiten der Adaption während der Laufzeit andererseits, stellt die Echtzeit-Plattform die Grundlage für ein Mikroskop-basiertes *Machine-Vision*-System dar, nach unserem Kenntnisstand das erste seiner Art weltweit.

Die Validierung der Funktionalität und des Systemverhaltens im Rahmen von Beispielanwendungen bestätigen das Steuerungskonzept sowie das korrekte Verhalten von Hard- und Firmware der Echtzeit-Plattform. Die Messung der Ausführzeiten aller zeitkritischen Protokollkommandos verifiziert das Leistungsverhalten im angestrebten Rahmen und liefert zugleich die Grundlage zur Optimierung benutzerdefinierter Ablaufprotokolle.

1 Einleitung

„Remember that time is money!“ [50] Als Benjamin Franklin diesen Satz 1748 in seiner Schrift „Advice to a Young Tradesman“ niederschrieb, waren die meisten Medikamente „Hausmittelchen“, die auf der Kenntnis der heilenden Wirkung verschiedener alltäglicher Substanzen basierten und eine über Jahrhunderte hinweg gesammelte und jeweils von einer an die nächste Generation weitergegebene Erfahrung widerspiegelten. Franklins Aussage kann daher in keiner Weise in einer direkten Beziehung zu der „Pharmakologie“ seiner Zeit gestanden haben.

Gut 250 Jahre später hat sich die Pharmakologie unter Verwendung von Erkenntnissen aus der Biologie, der Chemie, der Veterinär- und Humanmedizin sowie angrenzender Gebiete und durch den Einsatz modernster Technologien zu einem rasant wachsenden, weltweiten Multi-Billionen Dollar-Geschäft entwickelt [32, 37, 39]. Neben dem Kampf gegen Seuchen (wie z. B. Aids, Vogelgrippe) und Zivilisationskrankheiten (wie z. B. Herzkreislauferkrankungen oder Diabetes) unserer Zeit spielt bei der Entwicklung neuer Medikamente auch der Konkurrenzkampf unter den Pharmakonzernen eine bedeutende Rolle. Nach Jahren der Forschung und dem Durchlaufen der Prozesse zur Zulassung auf dem Markt, muss ein Medikament - durch Generika bedrängt - die vorfinanzierten Entwicklungskosten innerhalb von wenigen Jahren wieder einspielen. Um dabei Kosten zu senken und sich einen zeitlichen Vorsprung vor der Konkurrenz zu sichern, muss die zur Entwicklung benötigte Zeit stetig reduzieren werden. Vor diesem Hintergrund gewinnt Benjamin Franklins Aussage „time is money“ nun auch in Hinsicht auf die Forschung und Entwicklung neuer Medikamente eine aktuelle Bedeutung!

Die Prozesse von der zugrunde liegenden Entwicklungsidee bis hin zum verkaufsfähigen Medikament sind dabei sehr komplex, zeit- sowie kostenintensiv und benötigen Ressourcen aus unterschiedlichsten naturwissenschaftlichen, technischen, betriebswirtschaftlichen sowie juristischen Fachgebieten. Der Ablauf der Neuentwicklung eines Medikaments wird in Abbildung 1.1 anhand eines Flussdiagramms veranschaulicht und bietet mit kurzen Beschreibungen der einzelnen Phasen einen Überblick über die involvierten Vorgänge. Nach einer Umfrage von DiMasi et al. [37] dauert allein die Forschung in der vorklinischen Phase im Durchschnitt 4½ Jahre und kostet 335 Millionen US-Dollar. Sie wird gefolgt von klinischen Testphasen, die durchschnittlich weitere 6 Jahre dauern und mit Kosten von 467 Millionen US-Dollar zu Buche schlagen. Bis das Medikament letztendlich für den Markt zugelassen und dort erhältlich ist, vergehen weitere 1½ Jahre, mit denen zusätzliche Kosten von 98 Millionen US-Dollar einhergehen.

Ein wesentlicher Ansatz zur Kostenminimierung besteht darin, durch Miniaturisierung und Automatisierung der Prozesse in den frühen vorklinischen Phasen[1] eine steigende Zahl von Wirksubstanzen (Compounds) auf die „Targets“ zu testen um damit eine zunehmende Anzahl von „Leads“ in die späteren vorklinischen Phasen einbringen zu können [148, 49, 63, 67, 155, 106, 46]. Diese Trends der letzten zehn Jahre bedeuten eine zunehmende Verschiebung des „Flaschenhalses“ der vorklinischen Medikamentenentwicklung zu aufwändigen und damit zeit- und kostenintensiveren **Z**ell-**b**asierten (ZB)-Screening-Verfahren [99], die den Übergang vom Screening auf rein molekularer Ebene (**I**solated-**T**arget (IT)) hin zu Versuchsreihen an Tieren und letztlich am Menschen bilden [107].

[1] Die Miniaturisierung und die Automatisierung im Bereich der Medikamentenentwicklung werden z. B. durch die Techniken des **H**IGH **T**HROUGHPUT **S**CREENING (HTS), mikro (**µ**) **H**igh **T**hroughput **S**creening (µHTS) und „virtual Screening“ (IN SILICO) [169, 18, 91, 133] beschrieben.

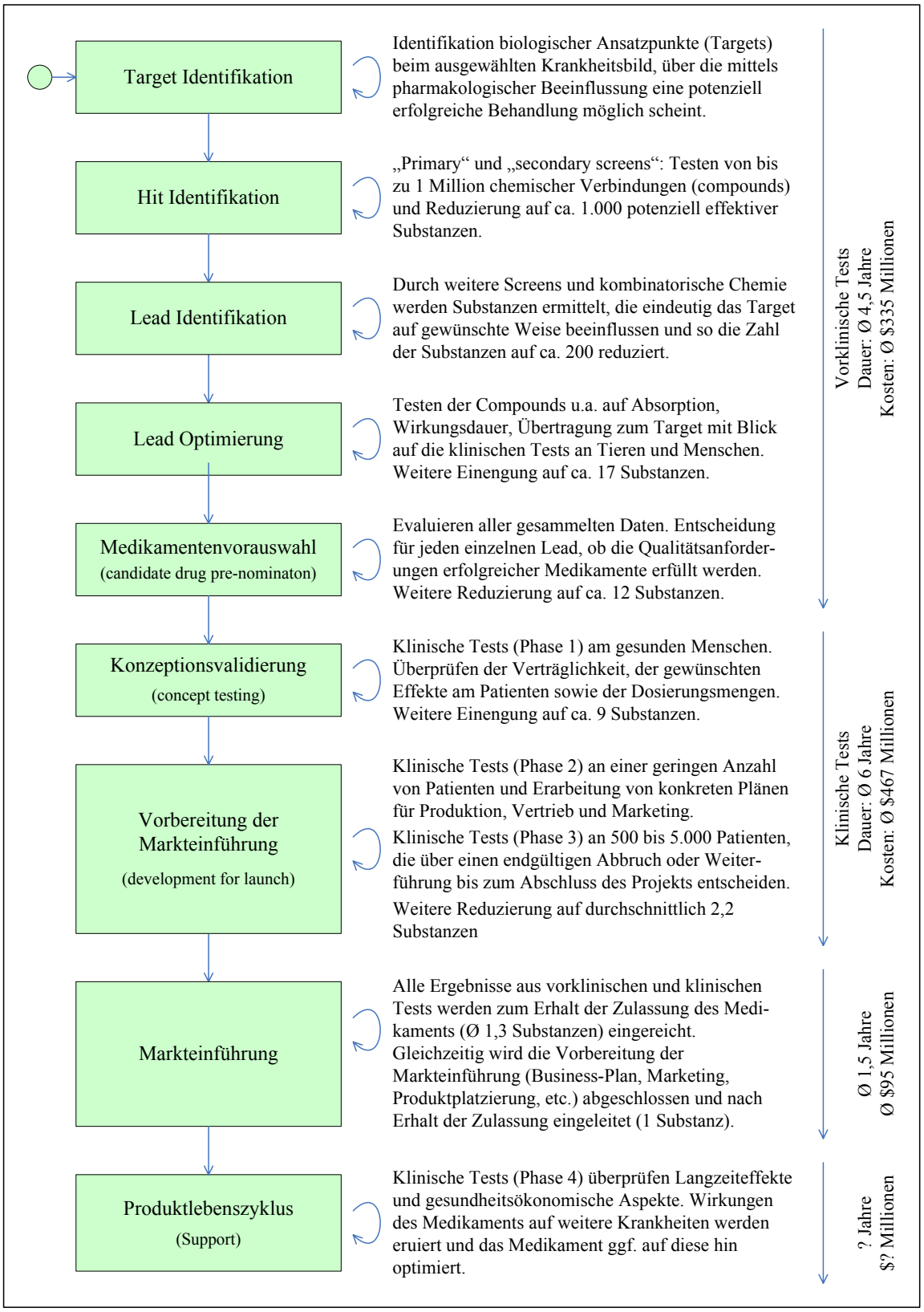


Abbildung 1.1: Beschreibung der einzelnen Phasen der Entwicklung eines neuen Medikaments. Die angegebenen Durchschnittswerte der Anzahl der in den einzelnen Phasen getesteten Substanzen, der Dauer sowie der Kosten von vorklinischen und klinischen Tests sowie der Markteinführung wurden der Veröffentlichung von DiMasi et al. entnommen. Quelle: [37]

Die in Abbildung 1.1 beschriebenen Prozesse der Medikamentenentwicklung haben sich in den letzten Jahrzehnten durch steigende Anforderungen an die Qualität der Produkte einerseits und durch höhere Auflagen der Behörden andererseits immer weiter verteuert. Wie die Graphik in Abbildung 1.2 zeigt, stehen einer lediglich schwach ansteigenden Zahl von „Blockbuster-Medikamenten“ ca. 60-fach gestiegene Kosten für Forschung und Entwicklung gegenüber [112, 125, 120]. Vor diesem Hintergrund müssen die „Erfahrungszeiträume“ dramatisch verkürzt werden und es müssen Modellsysteme eingeführt werden, die bereits lange vor den letztendlichen Validierungen durch Tierversuche bzw. klinische Studien am eigentlichen „Target Mensch“ eine Einengung von Millionen auf einige wenige, erfolgversprechende Medikament-Kandidaten gestatten.

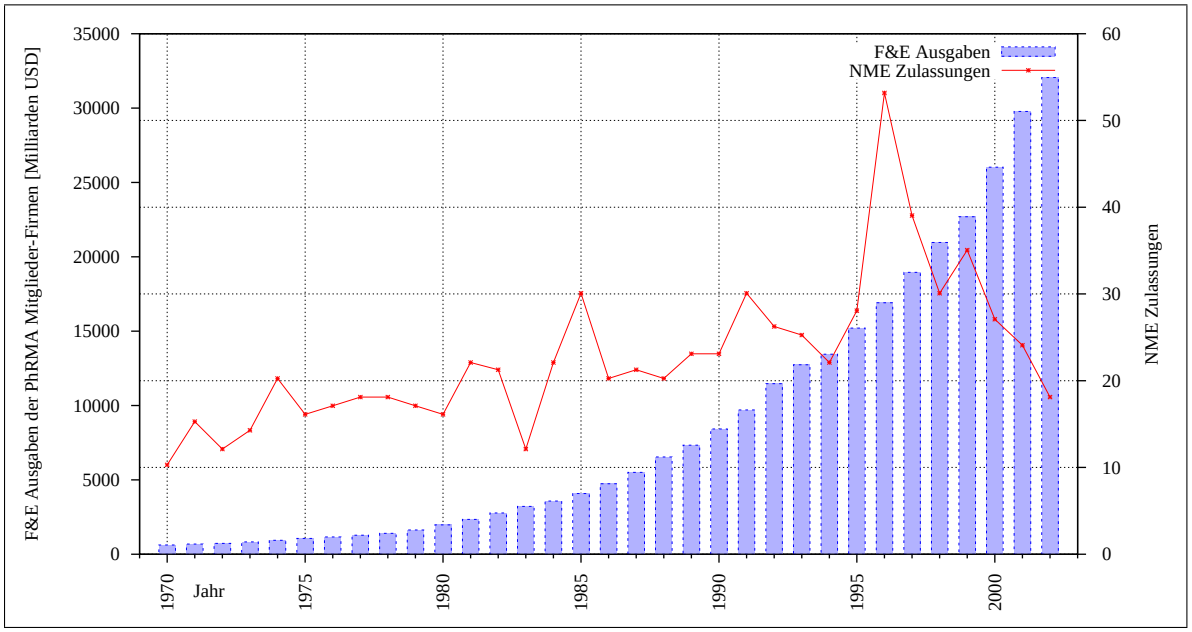


Abbildung 1.2: Diskrepanz zwischen steigenden Ausgaben für die pharmakologische Forschung und Entwicklung einerseits und der Anzahl der daraus hervorgehenden Ergebnisse andererseits - hier am Beispiel der Zulassung von **N**ew **M**olecular **E**ntities (NME); Quellen: [125, 120]

Vor diesem Hintergrund erlangen Verfahren (z. B. **Z**ell-**B**asiert (ZB)-ASSAYS), die eine schnellere Einengung der Anzahl aktiver Compounds ermöglichen, eine zentrale Bedeutung [60]. Die bisher im großen Stil praktizierte Beschränkung auf die reine Chemie (s. IT-Assays) erlaubt zu wenig spezifische Aussagen und hat zu viele falsche Fährten gelegt [107, 11, 136, 63]. Als Beispiel seien hier Bindungsstudien an Rezeptoren genannt, die, aus ihren zellulären Umfeld gerissen, zu unspezifischen Bindungsverhalten neigten und somit zu falschen Hits geführt haben.

Dem genannten Problem lässt sich durch Studien an höher integrierten Modellsystemen begegnen. Damit sind zweidimensionale Zellverbände (Zellkulturen) und in zunehmendem Maße auch dreidimensionale gewebeartige Zellkonglomerate gemeint, von denen nicht nur einfache *ja/nein* Ergebnisse abgeleitet werden können, sondern differenzierte, vielparametrige Aussagen [149]. Für die Anwendung entsprechender, zumeist lichtmikroskopischer Verfahren, hat sich der Name „High Content Screening“ (HCS) eingebürgert.

Die Vergrößerung der Anzahl erfasster Messparameter geht naturgemäß mit einer Verringe-

rung des Durchsatzes einher. „High Content“ und „High Throughput“ sind somit quasi komplementär. Die Motivation zur vorliegenden Dissertation speist sich aus dem Willen, dazu beizutragen, diesen Automatismus zu durchbrechen und sehr differenzierte, vielparametrige Assays mit hohem Probendurchsatz durchführen zu können. Unmittelbarer Anknüpfungspunkt war eine mehrfach preisgekrönte Lichtmikroskopie-Plattform[2], das **i**maging **MIC**roscope (iMIC)[3], die speziell für die automatisierte *high-end* Mikroskopie konzipiert wurde und deren bereits existierender Opto-Mechanik ein adäquates elektronisches Steuerungskonzept gegenübergestellt werden sollte.

Anwendungsfelder für eine solche hochautomatisierte high-end Lichtmikroskopie beschränken sich nicht auf die pharmazeutische Wirkstoffsuche, sondern finden sich auch auf den Gebieten der medizinischen Diagnostik, der Pathologie und innerhalb der sog. Hochdurchsatzbiologie. Letztere trägt der Tatsache Rechnung, dass zur Beschreibung komplexer biologischer Zusammenhänge große, statistisch signifikante Datensätze gewonnen werden müssen. Der auf diesem Gebiet beschrittene Weg ist durch drei in ihrer Entstehung zeitlich aufeinander folgende Forschungsfelder vorgezeichnet:

- Die GENOMICS (beschränkt sich auf die Entschlüsselung des genetischen Codes; Analogie beim Film: die Besetzungsliste) [4, 90, 33, 19],
- die PROTEOMICS (beschränkt sich auf die Erkundung des zellulären Proteoms; Analogie beim Film: Wer tritt wann und mit wem zusammen auf) [164, 147, 94, 146],
- die CELLOMICS (auch „Systems Biology“ genannt, zielt auf das modellhafte Verstehen komplexer biologischer Systeme und das Zusammenspiel ihrer Komponenten; Analogie beim Film: Das Drehbuch) [86, 26, 69, 175].

Unter Berücksichtigung aller Anwendungsfelder richtet sich die vorliegende Arbeit jedoch überwiegend am Bereich des Pharma-Screenings aus, deren Leistungsansprüche an die hier benötigte Performance die aller anderen Gebiete mit abdecken. Die in dieser Arbeit vorgestellten Ansätze und Lösungen wurden daher aus den Anforderungen erarbeitet, die sich aus den Anliegen des Umfelds des Pharma-Screenings ergeben.

1.1 Zell-Basierte Assays/Life Sciences

Zellbasierte Assays kommen bei der langwierigen Entwicklung neuer Medikamente auf mehrfache Weise zum Einsatz: Im Rahmen von HCS gestatten sie, im Anschluss an das Screening auf isolierte Targets, die Wirkung der ermittelten Hits bzw. Leads auf der IN-VITRO Ebene an Zellen und Geweben auf Effektivität und Nebenwirkungen zu überprüfen. Damit erleichtern ZB-Assays den Übergang hin zu IN-VIVO Versuchen an Tieren und später ebenfalls an Menschen. ZB-Assays sind weiterhin beim Screening von Targets unabdingbar, die auf Basis von Zell-Zell Interaktionen (Membranfunktionen, Signalübertragung, etc.) ermittelt wurden (wie z. B. beschrieben von Sun et. al sowie von Totok-Storb et. al [161, 154]). Auch hier gilt, dass mit zunehmender Eingrenzung der ermittelten Wirksubstanzen die Anzahl der zu betrach-

[2] Das iMIC ist u. a. ausgezeichnet mit dem Circle of Excellence Award (Photonics Spectra - 2005) und dem Bayrischen Innovationspreis (Bayern Innovativ, 2004).

[3] Entwickler und Hersteller des iMIC ist die Firma TILL Photonics, Gräfelfing, Deutschland.

tenden Parameter zunimmt, wenn man die Wirkung der Compounds im lebenden Organismus möglichst präzise abschätzen bzw. evaluieren möchte.

Hinzu kommt, dass Medikamente bei verschiedenen Patienten oft ungleiche Wirkungen hervorrufen. Aus diesem Grund gewinnt in jüngster Zeit auch der Einsatz von patientenorientierten, funktionalen Assays zur Diagnostik und zur Ermittlung individueller Therapien an Bedeutung. Anhand von Proben des erkrankten Gewebes (z. B. innerhalb der Krebstherapie) kann dabei z. B. mit Hilfe effektiver ZB-Assays die Auswahl des Arzneimittels und dessen Dosierung optimal auf den Patienten eingestellt werden.

Auch im Zuge der Identifikation neuer Targets werden ZB-Assays verwendet [93] und sind somit unmittelbar mit der Grundlagenforschung in Biologie, Medizin, Chemie und anderen naturwissenschaftlichen Disziplinen verknüpft. Die unter dem Begriff „Life Sciences“ zusammengefassten Disziplinen beschäftigen sich primär mit der Erforschung zellulärer Funktionen sowie interzellulärer Zusammenhänge bzw. Wechselbeziehungen und zielen dabei auf einen möglichst ganzheitlichen Einblick in das „System Zelle“[4], auf dessen Signalvielfalt und innere Signalführung, wie Abbildung 1.3 exemplarisch veranschaulicht. Auch dort ist es essentiell, eine Vielzahl unterschiedlichster zellulärer Parameter gleichzeitig erfassen zu können.

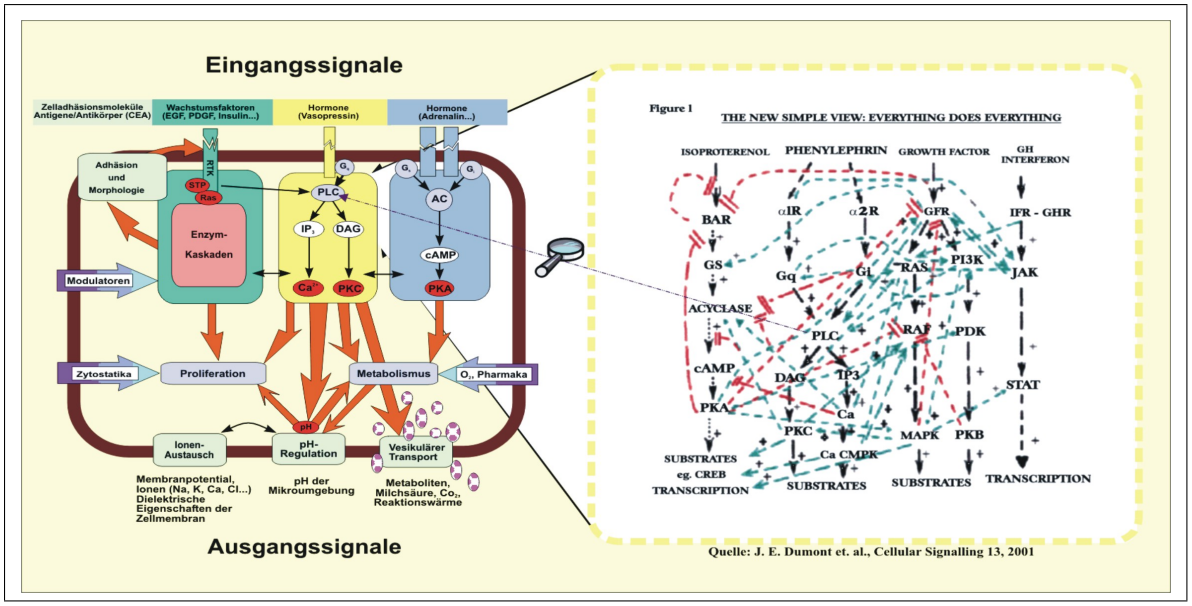


Abbildung 1.3: Inter- und intrazelluläre Signale und Signalwege einer Zelle - die zelluläre Signalverarbeitung ist hochgradig parallelisiert und vernetzt; Quellen: [89, 40].

Die Zelle als komplexes, lebendes, nanostrukturelles System [2] stellt im Vergleich zu isolierten Targets höchste Anforderungen an die Versuchsbedingungen und damit an die zur Durchführung notwendige Ausrüstung. Dabei ist es sowohl für die Reproduzierbarkeit als auch für die Qualität der Ergebnisse von großer Bedeutung, dass die in-vitro Gegebenheiten die in-vivo Verhältnisse durch konstante Versuchsbedingungen und Umgebungsparameter so gut wie möglich simulieren: Je nach Empfindlichkeit der untersuchten zellulären Proben müssen für die Durchführung einer Messung/eines Screens u. a. die Temperatur, die Luftfeuchtigkeit, der Sauerstoffgehalt und der pH-Wert geregelt werden. Zudem sollte die Belastung der Proben durch minimal-invasive Messverfahren möglichst gering gehalten werden (z. B. Reduzierung

[4] Beispiele hierfür führen Huang und Ingber [72] sowie Folch und Toner [48] auf.

der PHOTOTOXIZITÄTS-Effekte durch Ausblenden der Beleuchtung zwischen zwei Bildaufnahmen).

1.2 Ausgangspunkt und Zielsetzung

Die Arbeiten zur hier vorgelegten Dissertation wurden im Umfeld zweier universitärer Arbeitsgruppen und dreier Firmen durchgeführt. In einer aufeinander abgestimmten Vorgehensweise wurden dabei Synergien genutzt, die sich durch komplementäre Kompetenzen auf den Gebieten der Life Sciences und den sich ergänzenden Technologien ergaben:

- **B**io**I**maging **Z**entrum (BIZ) der **L**udwig-**M**aximilians-**U**niversität (LMU),
- **L**ehrstuhl für **M**edizinische **E**lektronik (LME) der **T**echnischen **U**niversität **M**ünchen (TUM),
- TILL Photonics GmbH,
- TILL I.D. GmbH,
- SmartMove GmbH.

Die Technologie-Entwicklung am BIZ konzentriert sich auf moderne Mikroskopietechniken, die auf der optischen Plattform iMIC [31, 166, 167, 116, 135] aufsetzen, sowie auf die Ausarbeitung und die hardwareseitige Umsetzung von Konzepten für voll automatisierte, interaktive und adaptive Steuerungen von modularen, Mikroskop-basierten Analysesystemen [54, 53, 52, 55]. Die Evaluierung der am BIZ erarbeiteten Konzepte und der daraus entstandenen Geräte geschieht durch den unmittelbaren Einsatz auf dem Gebiet der Grundlagenforschung anhand des Wachstumsverhaltens von Neuronen [110, 111].

Am LME wird u. a. an der Entwicklung bioelektrischer Sensorik gearbeitet, die über das Messen einer Vielzahl von Parametern unmittelbar an lebenden, zellulären Proben für das Beobachten der Zellaktivität bestimmt ist (s. LAB-ON-A-CHIP) [24, 95, 180, 178, 171]. Zur Erhöhung des Durchsatzes wird diese Technologie zudem in der Entwicklung von MIKROTITERPLATTEN mit integrierter Sensorik für MIKROSENSORARRAY-basiertes Screening eingesetzt [179, 129, 54]. Zur Durchführung von Langzeitanalysen werden modulare Erweiterungen wie ein automatisierbares Fluidiksystem zur Versorgung von lebenden, zellulären Proben mit Nähr- und Wirkstofflösungen und ein Klima-Kontrollsystem entwickelt. Zur Funktionsüberprüfung und zur Entwicklung neuer Assays, welche die neuen technischen Möglichkeiten nutzen, werden die entstandenen Geräte am LME u. a. innerhalb der Forschung an Krebs-Zelllinien eingesetzt [108, 122, 174].

Vorarbeiten und technischer Ausgangspunkt

Die am LME entwickelten Konzepte und Geräte zur sensorbasierten Beobachtung von lebenden, zellulären Proben bilden die Basis für ein Forschungsprojekt[5] zur Schaffung einer vollautomatisierten Analyseplattform - dem IMR (s. Abbildung 1.4) [95, 97, 180]. Der IMR soll die Durchführung zellulärer Assays im Rahmen von HCS mit HTS-üblichem Probendurchsatz ermöglichen. Im Zentrum dieser Analyseplattform steht eine mit Mikrosensorarrays bestückte

[5] Das Projekt wird durch die Bayrische Forschungsstiftung unter dem Namen „Intelligente Multiwellplatte“ (IMWP) gefördert. Das Ende der Förderzeit ist August 2007.

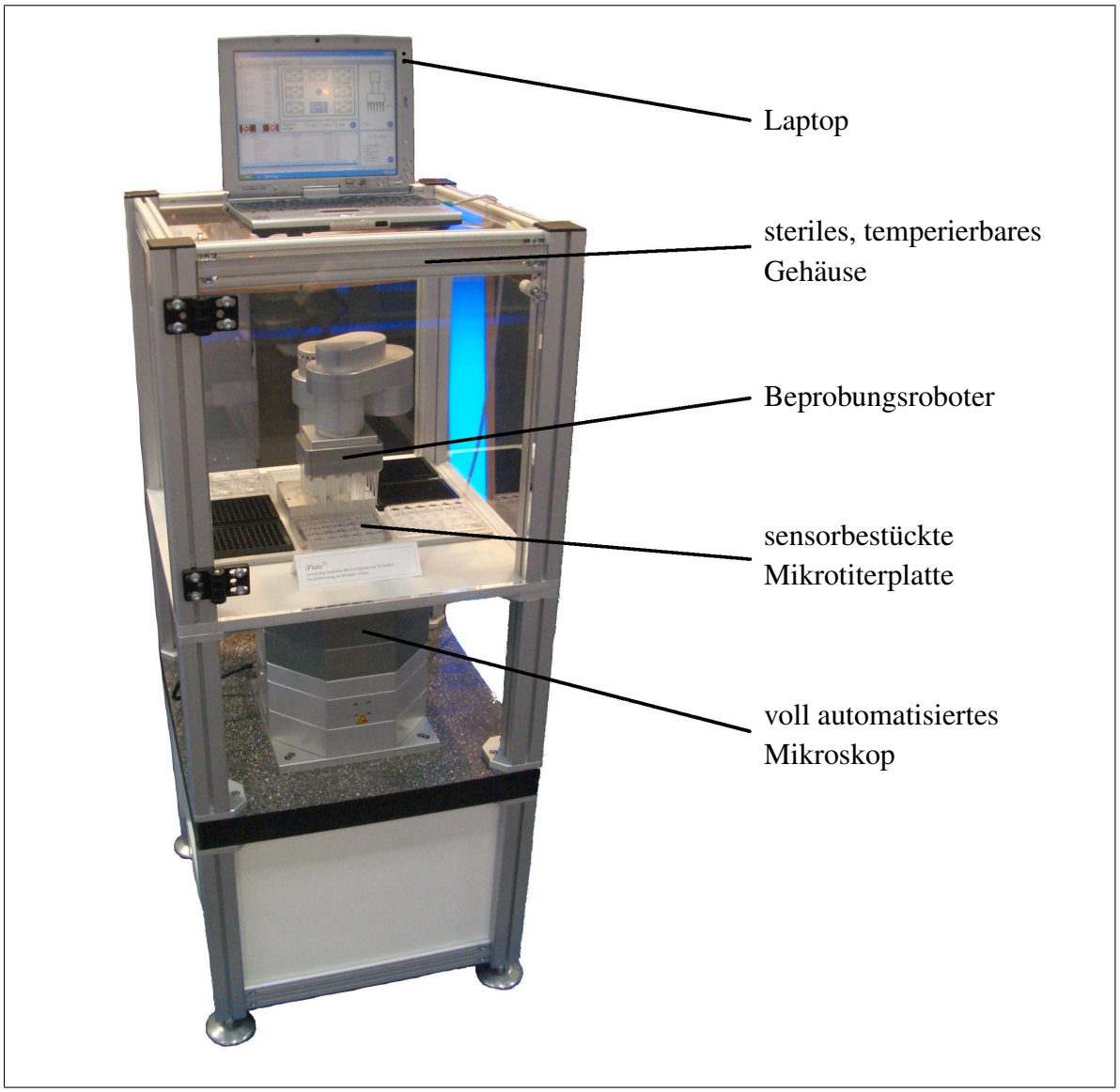


Abbildung 1.4: Prototyp des **I**ntelligent **M**icroplate **R**eader (IMR) [180]: voll automatisiertes Mikroskop (TILL-Photonics), sensorbestückte Mikrotiterplatte, Beprobungsroboter (Thermo Electron (Oberschleissheim) GmbH (ehemals H&P Labortechnik AG), Steinbeiszentrum Lab-on-Chip-Systeme, LME), steriles, temperierbares Gehäuse und Steuerrechner; Quelle: LME

Mikrotiterplatte mit integrierten Fluidikeinheiten. Das Fluidiksystem der Mikrotiterplatte wird von einem automatisierbarem Pipettierroboter mit Nährlösungen und Wirkstoffen versorgt. Die Umgebungsparameter aller Assay-relevanten Materialien und Geräte werden durch ein Temperierungs- und Klimakontrollsystem entsprechend den Versuchsanforderungen konstant gehalten. Die Verwendung durchsichtiger Trägermaterialien für den sensorbestückten Boden der Mikrotiterplatte ermöglicht zusätzlich zu der sensorischen Detektion von Parametern die Anwendung lichtmikroskopischer und damit optischer Methoden. Hierbei handelt sich sich neben der qualitativen Auswertung ganzheitlicher Probenabbilder (z. B. zur Analyse der Zellmorphologie) vor allem um die im Screeningbereich wohl etablierten fluoreszenzbasierten Methoden [143, 7, 140, 177, 20, 65]. Durch die Integration eines Mikroskopiesystems und der sensorgestützten Mikrotiterplatte vereint das im Rahmen des Projekts erarbeitete modulare Ge-

samtsystem damit erstmals die mannigfaltigen Möglichkeiten optischer Analyseverfahren mit denen bioelektrischer Sensortechnologie. Innerhalb des Projekts ergänzen sich die beteiligten Arbeitsgruppen und industriellen Projektpartner durch ihre langjährigen Erfahrungen und die hierbei erworbenen Kompetenzen auf den jeweiligen Fachgebieten.

Als Basis für die lichtmikroskopische Charakterisierung der Zellen wird die in Zusammenarbeit von TILL Photonics und dem BIZ entwickelte Mikroskop-Plattform iMIC verwendet, die speziell auf die Bedürfnisse der voll automatisierten und computergestützten Mikroskopie abgestimmt wurde. Alle bei herkömmlichen Mikroskopen manuell bedienten Funktionen, wie Fokussierung, Objektivwechsel, Filterwechsel[6] und das Verschieben des Präparats relativ zum Objektiv, wurden motorisiert und in einem kompakten, modular skalierbaren, achteckigen Baustein integriert (s. Abbildung 1.5).

Der für die Beleuchtung verwendete „Polychrome“ der TILL Photonics ist eine monochromatische Lichtquelle, die speziell zur Fluoreszenzanregung im Bereich der Life-Sciences entwickelt wurde [165]. Durch die Verwendung digitaler Scan-Technologie[7] ermöglicht der Polychrome (aktuelle Version V, s. Abbildung 1.6) ein schnelles Umschalten von der „Ruhewellenlänge“ (Blackout) auf die für die Anregung von Fluoreszenz benötigte Wellenlänge bzw. Wellenlängen bei Mehrfachfluoreszenz. Neben ausgezeichneten Ausleuchtungseigenschaften (stufenloses Einstellen der Wellenlänge von 320 nm bis 680 nm bei einer Verstellgeschwindigkeit von bis zu 400 nm/ ms, gleichmäßige Ausleuchtung, Beleuchtungsstabilität durch geregelte Temperaturkontrolle, etc.) verfügt der Polychrome V neben der traditionellen analogen auch über eine Standard-PC-Schnittstelle (USB oder EIA-232-Schnittstelle[8]). Über die PC-Schnittstelle können zusätzlich komplexe Abläufe, wie z. B. sich wiederholende Mehrfachbeleuchtung mit verschiedenen Wellenlängen und/oder Beleuchtungszeiten, in Form eines Protokolls vordefiniert werden. Diese Abläufe können anschließend wahlweise über den PC gestartet oder über ein externes Triggersignal angestoßen werden. In den frei konfigurierbaren Ablaufprotokollen besteht bei eigenem Timing der Lichtquelle auch die Möglichkeit, Triggersignale generieren zu lassen, die der Synchronisation von externen Geräten dienen.

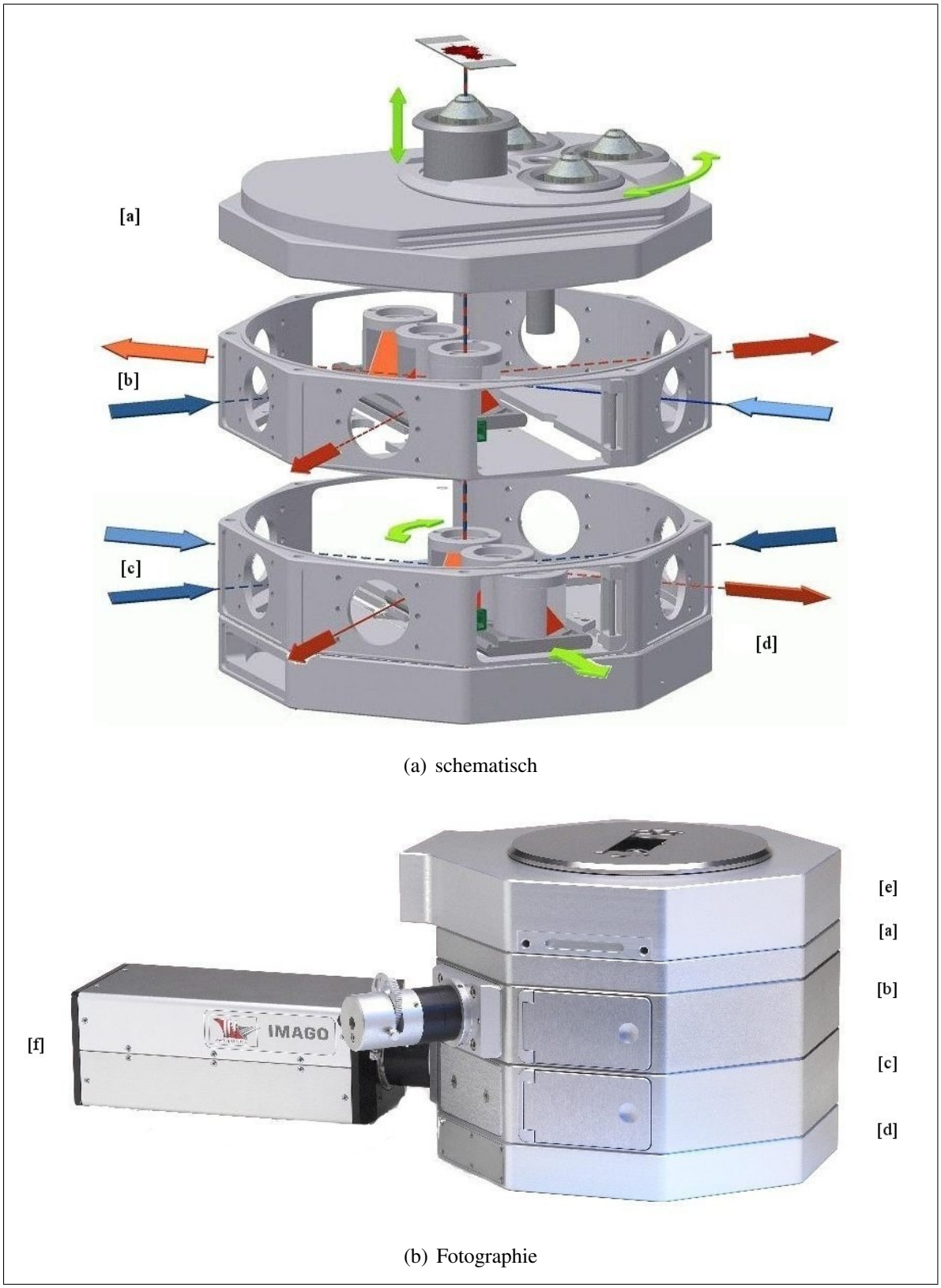
Die im Polychrome V eingesetzte Scan-Technologie dient in den am BIZ bearbeiteten Projekten als Basis für jegliche Funktionalität, die schnelle und präzise Positionierung optischer Komponenten wie z. B. Spiegel oder optische Gitter in Strahlengängen erfordert. Beispielanwendungen, die im Verlauf dieser Arbeit noch näher vorgestellt werden, befinden sich auf dem Gebiet spezieller Beleuchtungsmethoden, dem schnellen „Umschalten“ zwischen verschiedenen Beleuchtungen sowie der Laser-Scanning-Mikroskopie.

Die technische Basis für die Arbeiten zu dieser Dissertation bilden Geräte, die innerhalb des beschriebenen Arbeitsumfelds entwickelt wurden. Gliedert man die Arbeiten des IMR-Projekts grob in die Bereiche (i) Sensorik, (ii) Umgang mit Proben und (iii) Mikroskopie, so ist die vorliegende Arbeit vor allem im Bereich der Mikroskopie angesiedelt. Beim technischen Ausgangspunkt handelt es sich daher im Wesentlichen um die Lichtmikroskop-Plattform der TILL Photonics sowie um die digitalen Regelkonzepte zum Betreiben von galvanometrischen Moto-

[6] Filter bzw. Filterwechsler werden in der FLUORESZENZMIKROSKOPIE benötigt [131].

[7] Digitale Scan-Technologie basiert auf einem galvanometrischen Motor, der über einen digitalen Regler der Firma SmartMove angesteuert wird.

[8] Die EIA-232-Schnittstelle ist nach dem ursprünglichen Standard weitläufig als RS232-Schnittstelle bekannt.



(a) schematisch

(b) Fotographie

Abbildung 1.5: imaging **MIC**roscope: voll-automatisierbares Mikroskop für computergestützte Mikroskopie (TILL Photonics): Objektiv-Wechsler [a], Beam-Hub mit integriertem Filterwechsler zur Einkopplung der Beleuchtungsquelle(n) [b], Beam-Hub mit integriertem Filterwechsler zum Auswählen des/der Detektionsgerät(e)s [c], Mikroskopboden mit integrierten Schrittmotorsteuerungen [d]; motorisierter xy-Tisch [e] und CCD-Kamera [f] (beides nur in Bild (b)).

Abbildung 1.6: Polychrome V - Lichtquelle für Fluoreszenzmikroskopie (TILL Photonics): Stufenloses, programmierbares Einstellen der Wellenlänge des zur Anregung von Fluoreszenz benötigten Lichtes (320 nm bis 680 nm).

ren der Firma SmartMove, einer Ausgründung des BIZ. Als Bestandteil vieler Geräte der TILL Photonics eingesetzt, bilden letztere zudem die Basis für eine Reihe der am BIZ umgesetzten Mikroskopietechniken (z. B. LASER-SCANNING-basierte KONFOKALMIKROSKOPIE).

Die einzelnen Mikroskopie-Geräte bzw. -Komponenten eignen sich für den Einsatz innerhalb vieler verschiedener Anwendungsgebiete. Als Beispiele hierfür sind neben der computergestützten, automatisierten Mikroskopie, der vollautomatischen Mikroskop-basierten Hochdurchsatzanalyse (wie z. B. im Fall des IMR) vor allem die angestrebten Mikroskop-basierten Maschine-Vision Funktionalität zu nennen. Die einzelnen Anwendungsgebiete werden weiter unten detailliert vorgestellt (s. Abschnitt 2.2). Eine computergestützte, automatisierte Mikroskopie erfordert das Zusammenspiel einer Vielzahl optischer, mechanischer und elektronischer Einzelkomponenten. Um dabei einen maximalen Probendurchsatz realisieren zu können, müssen Totzeiten minimiert und das zeitliche Zusammenwirken optimiert werden. Daraus ergibt sich die Notwendigkeit einer zeitlich exakt aufeinander abgestimmten, geräteübergreifenden Synchronisation im Rahmen eines zuerst einmal streng deterministischen Systemverhaltens. Um darauf aufbauend auch ein interaktiv bzw. adaptiv funktionierendes Machine-Vision-System konzipieren zu können, bei dem zusätzlich zum deterministischen und zeitoptimalen Systemverhalten die Möglichkeit des adaptiven Eingriffs in vorher definierte Abläufe gewährleistet ist, müssen sich schnelle Kommunikationswege integrieren lassen, über die ergebnisabhängige Änderungen der Messparameter kommuniziert werden können.

Um das geforderte deterministische Verhalten bei gleichzeitigem zeitoptimalen Zusammenspiel aller Komponenten in einem modularen Gesamtsystems zu garantieren, bedarf es einer wirklichen Echtzeit-Steuerung. Der Begriff der Echtzeit bedeutet, dass ein System definitiv innerhalb einer festgelegten Zeitspanne reagiert. Im Interesse eines maximalen Probendurchsatzes kommt

es jedoch nicht nur darauf an, die Reaktionszeit definiert, sondern auch so gering wie möglich zu halten.

Zielsetzung

Ausgehend von der Motivation, den scheinbaren Widerspruch zwischen High-Content und High-Throughput zu relativieren, lag die Zielsetzung dieser Arbeit auf der Entwicklung eines modularen elektronischen Steuerungskonzepts, das als Basis für ein Mikroskop-basiertes Machine-Vision System dienen kann. Die Umsetzung dieses Anspruchs in Hard- und Firmware und die damit realisierbare Echtzeitplattform sollte durch Hardwaresynchronisation aller Komponenten der Mikroskop-basierten Systeme die Optimierung zeitlich deterministischer Abläufe gewährleisten und im Hinblick auf interaktive Bedienung und Machine-Vision-Funktionalität ein adaptives Eingreifen in vordefinierte Ablaufprotokolle ermöglichen.

Im Einzelnen ließen sich folgende Teilziele identifizieren:

- Anwendungsunabhängige Konzepterstellung im Hinblick auf modulare Integration an sich autonomer Geräte in Gesamtsysteme,
- Schaffung einer modularen Hardwareplattform zur Synchronisation aller Systemkomponenten in Echtzeit,
- Konzeption und Implementierung einer echtzeitfähigen Firmware, die durch freie Konfigurierbarkeit der zeitlichen Abläufe und Abhängigkeiten größtmögliche Flexibilität bietet.

Die detaillierte Spezifikation der Anforderungen an das Konzept, die Gestaltung der Hardware und die Funktionalität von sowohl Hard- als auch Firmware erfolgt im Weiteren anhand der genaueren Betrachtung der geplanten Anwendungsgebiete (s. Abschnitt 2.2).

2 Stand der Technik und Anwendungsgebiete

Ziel dieser Arbeit ist die Konzeption einer Echtzeitplattform und deren hardwareseitige Umsetzung, die als Basis für ein Mikroskop-basiertes Machine-Vision-Konzept dienen kann. Ihr Anwendungsfeld liegt in den Life Sciences. Die Anforderungen entspringen zum einen der modernen *high-end* Lichtmikroskopie und zum anderen der computergesteuerten, vollautomatisierten *Reader*-Technologie. Auf beiden Gebieten wurden in den vergangen Jahrzehnten immense Fortschritte erzielt. Bis zum heutigen Tag gibt es jedoch nur wenige technologische Ansätze, die das Beste dieser „beiden Welten“ in eine einheitliche Plattform integrieren. Die folgenden Abschnitte geben den diesbezüglichen Stand der Technik auf der Ebene der Systemintegration und Systemsteuerung wieder, beschränken sich dabei jedoch nicht nur auf das, was im Bereich der automatisierten Mikroskopie bisher möglich ist, sondern zeigen generell die Möglichkeiten der modernen Lichtmikroskopie und damit den Nachholbedarf auf, den eine automatisierte, auf eine Erfassung möglichst vieler zellulärer Parameter ausgerichtete Mikroskopie aufweist.

2.1 Stand der Technik auf den relevanten Gebieten

2.1.1 Mikroskopie

Die Ausbauvariationen kommerziell erhältlicher Mikroskope sind nahezu unüberschaubar. Je nach Anwendungsgebiet wird zwischen industrieller Nutzung und dem Einsatz im Bereich der Life Sciences differenziert. Da im Rahmen dieser Arbeit vorrangig die Geräte des letzteren Bereichs und hier insbesondere PC-unterstütze, automatisierbare Systeme von Interesse sind, wird der Stand der Technik im Folgenden anhand von high-end Mikroskopiesystemen renommierter Hersteller vorgestellt.

Cell Observer - Carl Zeiss

Der *Cell Observer* von Carl Zeiss kombiniert ein motorisiertes invertiertes Forschungsmikroskop (Axiovert 200M), eine digitale Kamera (AxioCam), die eine für Fluoreszenzaufnahmen ausreichende Empfindlichkeit aufweist, eine Quecksilber- oder Xenonlichtquelle mit wechselbaren Anregungswellenlängen und verschiedene Geräte zur Zellkultivierung mit einer Software (AxioVison) für automatisierte Bildaufnahme. Über eine im Mikroskop integrierte Elektronik werden die zur Bildaufnahme benötigten Komponenten miteinander synchronisiert. Der Cell Observer (s. Abbildung 2.1) ist primär für eine Zielgruppe konzipiert, die u. a. quantitative Langzeitanalysen von lebenden Zellen basierend auf Techniken der Fluoreszenzmikroskopie durchführt. Hierfür kann das computergestützte Imaging-System[1] um Peripheriegeräte zur Kontrolle der Temperatur, des pH-Werts, des CO_2-Gehalts sowie der Luftfeuchtigkeit im Reagenzraum (Klimakammer) erweitert werden. Die verschiedenen Parameter werden an den entsprechenden Geräten manuell eingestellt. Eine unmittelbare Verbindung zum Imaging-System und damit die Möglichkeit zur Automatisierung der Abläufe, wie z. B. dem Start eines Versuchs bei Erreichen der gewünschten Klimabedingungen oder dem Abbruch bei gravierenden klimatischen Veränderungen, besteht nicht.

[1] Der Begriff *Imaging-System* wird an späterer Stelle ausführlich erläutert (s. Abschnitt 2.2.1).

Abbildung 2.1: Live Cell Imaging-System *Cell Observer* von Carl Zeiss. Quelle: Carl Zeiss Internetauftritt: www.zeiss.de

cellˆR - Olympus

Das Imaging-System *cellˆR* (s. Abbildung 2.2) von Olympus kann sowohl mit aufrechten[2] als auch mit inversen Mikroskopen[3] betrieben werden. Neben einer CCD-Kamera ist das Herzstück des Systems eine Filterrad-basierte monochromatische Lichtquelle[4]. Die Ablaufkontrolle des Systems erfolgt als PC-basierte Steuerung über eine PCI-Echtzeiterweiterung von National Instruments. Alle zur Bildaufnahme benötigten Komponenten werden hierüber miteinander synchronisiert. Die zeitliche Auflösung beträgt 1 ms. Bis zu drei digitale Ausgänge[5] können zur Synchronisation von zusätzlichen Peripheriegeräten genutzt werden. Neben TIRF- und FRET-Erweiterungen bietet Olympus ebenfalls eine Klimakammer (Cellˆ Cubator) zur Regulierung der Luftfeuchtigkeit (Umgebungswert aufwärts), des CO_2-Gehalts (Umgebungswert bis Umgebungswert + 10%) sowie der Temperatur (Umgebungswert bis 42°C).

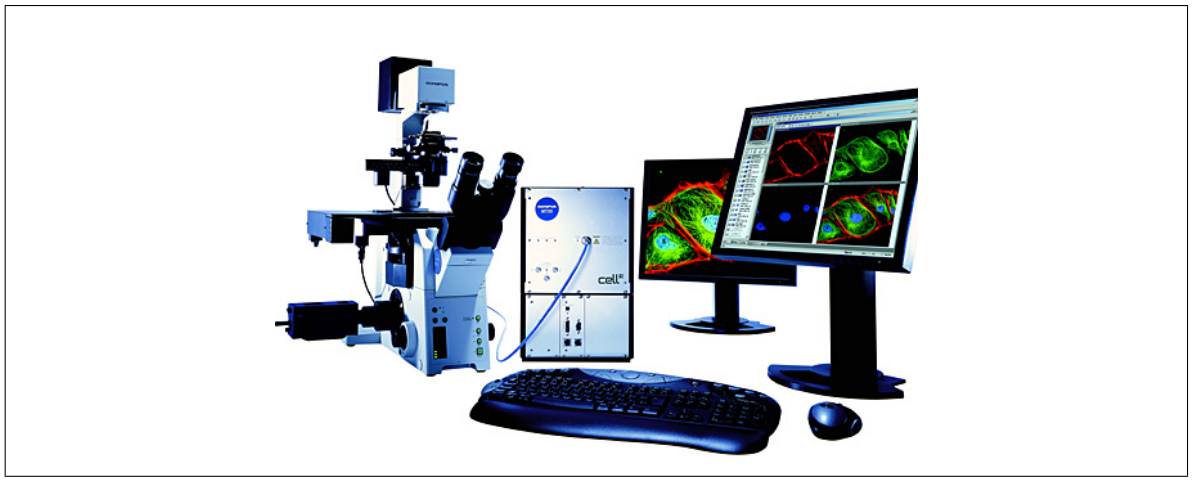


Abbildung 2.2: Live Cell Imaging-System *cellˆR* von Olympus. Quelle: Olympus Internetauftritt: www.olympus-europa.com/microscopy

[2]Gemeint sind aufrechte Olympus-Mikroskope der Baureihe BX.
[3]Gemeint sind inverse Olympus-Mikroskope der Baureihe IX.
[4]Olympus Lichtquelle für Fluoreszenzanregung *MT20*: Filterrad mit acht Filterpositionen, 14-stufige Abschwächung von 1 % bis 100 %, Filterwechsel in mindestens 58 ms bei benachbarten Filterpositionen.
[5]Die digitalen Ausgänge stehen über **B**ayonet **N**eill **C**oncelman (BNC)-Steckverbinder zur Verfügung.

AF6000 LX - Leica Microsystems

Leica bietet mit dem *AF6000 LX* (s. Abbildung 2.3) ein integriertes Komplettsystem vorwiegend für Anwendungen des Weitfeld-Fluoreszenz-Imaging. Über entsprechende Erweiterungen können ebenfalls TIRF- und FRET-Messungen durchgeführt werden. Auf die Analyse von lebenden Zellen in Langzeitmessungen ausgelegt, kann die Temperatur (Umgebungstemperatur + 3°C bis Umgebungstemperatur + 20°C) sowie der CO_2-Gehalt (0% bis 7,5%) innerhalb der Klimakammer stabilisiert werden. Laut Hersteller werden alle Komponenten des Imaging Systems inklusive der Fokussiereinheit und der motorisierten Wechsler für die in der Fluoreszenzmikroskopie benötigten Filter in Echtzeit bedient. Allerdings werden keine Informationen über das Steuerungsschema der Einzelkomponenten oder die Reaktionszeit (Echtzeit-Raster) des PC-basierten Systems preisgegeben.

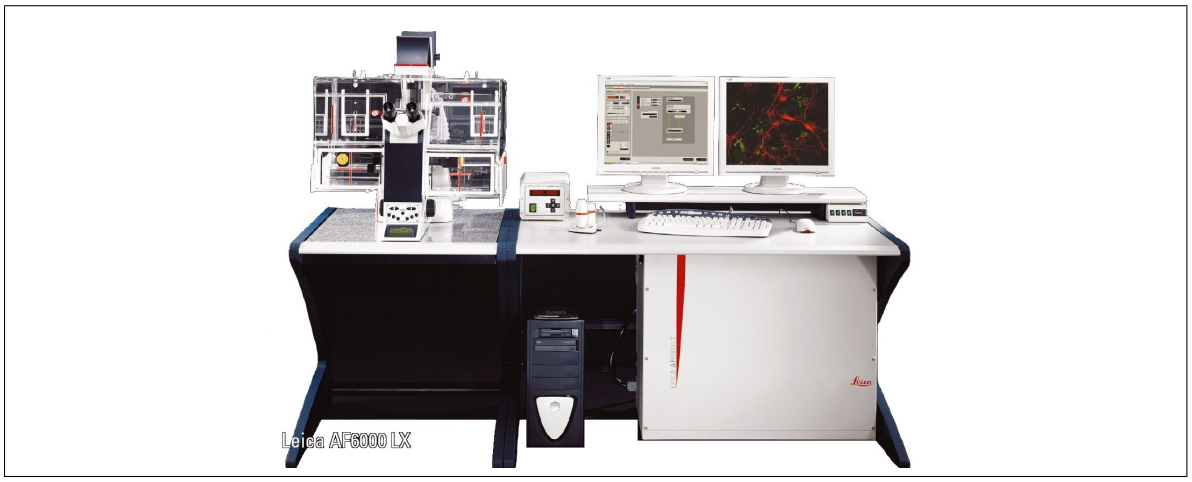


Abbildung 2.3: Live Cell Imaging-System *AF6000 LX* von Leica Microsystems. Quelle: Leica Microsystems Internetauftritt: www.leica-microsystems.de

Eclipse TE2000-E - Nikon

Das Mikroskop *Eclipse TE2000-E* von Nikon wird prinzipiell als rein manuell bedientes Forschungsmikroskop angeboten. In der high-end Version *TE2000-E* werden ein motorisierter Fokus sowie eine motorisierte Auswahl der verwendeten Beleuchtungsquelle integriert. Das System kann durch zusätzliche Komponenten wie u. a. eine externe Fokussierungseinheit, ein motorisierter Objektivwechsler oder Digitalkameras ergänzt werden. Die Bedienung der motorisierten Komponenten erfolgt mittels einer Fernbedienung (*Smart Touch*). Alle elektronischen Komponenten werden über eine Kommunikationseinheit (Communication HUB controller) miteinander verbunden, die über eine EIA-232 Verbindung zudem den Anschluss des Mikroskops an einen PC ermöglicht. Die Verbindung zwischen der Kommunikationseinheit und den daran angeschlossenen Geräten erfolgt über den standardisierten **C**ontroller **A**rea **N**etwork (CAN)-Bus.

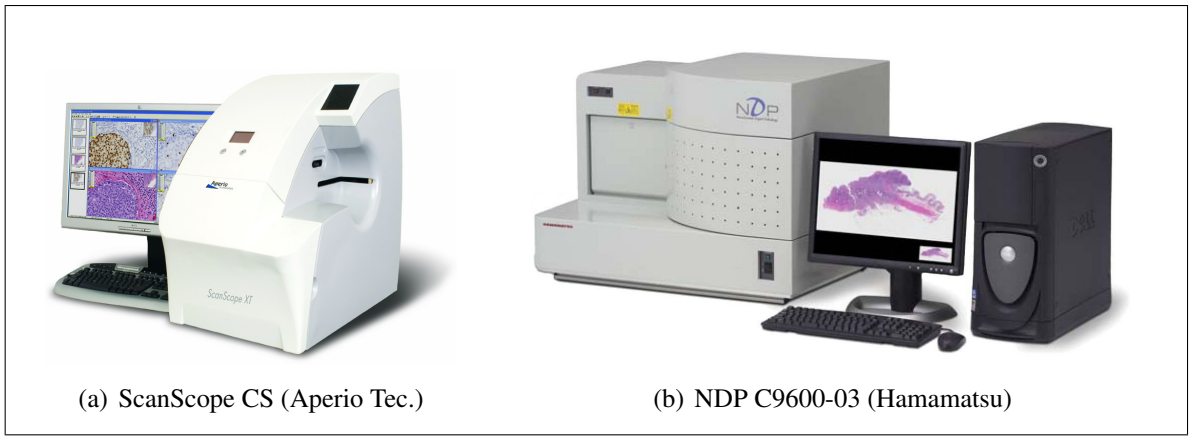
Das Nikon Mikroskop Eclipse TE2000-E kann mittels Drittanbietersoftware über einen PC bedient werden, ist entgegen den bisher vorgestellten Mikroskopie-Plattformen allerdings nicht speziell als Imaging-Systemen ausgelegt.

2.1.2 Vollautomatische Reader

Der Markt der vollautomatischen, optischen Reader war bisher primär auf Mikrotiterplatten-basiertes Hochdurchsatz-Screening molekularer Proben ausgelegt. Bei einem Screen werden

Abbildung 2.4: Invertiertes Forschungsmikroskop *Eclipse TE2000-E* von Nikon. Quelle: Nikon Internetauftritt: www.nikon-instruments.jp/eng

hierbei für jedes Well einer Mikrotiterplatte einfache *Ja/Nein*-Ergebnisse ermittelt, ob es zu einer Bindungsreaktion kommt oder nicht. Ein differenzierteres Bild, wie es nur zellbasierte Assays zu liefern vermögen, ist damit nicht zu erhalten. Auch für zellbasierte Assays gibt es bereits automatisierte Reader-Systeme, doch sind sie in der Regel monofunktional, d. h. sie beschränken sich lediglich auf eine Mikroskopietechnik, wie die nachfolgenden Beispiele zeigen.



(a) ScanScope CS (Aperio Tec.) (b) NDP C9600-03 (Hamamatsu)

Abbildung 2.5: Beispiele für automatische Reader zum Scannen vollständiger Objektträger (Slide): *ScanScope CS* von Aperio Technologies(Quelle: www.Aperio.com) und *NanoZoomer Digital Pathology C9600-03* von Hamamatsu (Quelle: www.Hamamatsu.com).

Ein Beispiel sind die in der Histo-Pathologie eingesetzten optischen Reader, die ausschließlich auf Durchlichtmikroskopie ausgelegt sind und mit denen auf Objektträger (Slides) aufgebrachte Gewebeschnitte automatisch „eingescanned“ werden können. Geräte wie die in Abbildung 2.5 gezeigten Slide-Scanner erzeugen in automatisierten Verfahren hoch aufgelöste digitale Abbilder der Objektträger[6], auch *virtual slide* genannt. Die digitalisierten Objektträger dienen z. B. der einfachen und platzsparenden Archivierung der Gewebeproben oder der Ferndiagnose durch Experten via Internet.

Für die Analyse lebender Proben werden gewöhnlich Mikrotiterplatten eingesetzt und die darin wachsenden Zellkulturen mittels Weitfeld-Epifluoreszenz-Techniken charakterisiert. Die dafür

[6]Im Beispiel des in Abbildung 2.5 gezeigten NanoZoomer von Hamamatsu wird ein 20 mm mal 20 mm großer Ausschnitt eines Objektträgers in ein 1,9 Gigapixel großes, digitales Bild konvertiert.

eingesetzten automatischen Reader-Plattformen reichen von kompakten Imagern (z. B. der *BD Pathway 415* der Firma Becton Dickinson, Franklin Lakes, USA; s. Abbildung 2.7(a)), über Mikrotiterplatten-Imaging-Systeme mit Erweiterungen zur Automatisierung (z. B. der *IN Cell Analyzer 1000* der Firma GE Healthcare, Little Chalfont, England; s. Abbildung 2.7(b)) bis zu komplexen, voll integrierten Screening *Workstations* (z. B. der *ImageXpress 5000A* der Firma Molecular Devices, Sunnyvale, USA; s. Abbildung 2.7(c)). Da hier nicht mehr mit fixierten - also toten - Proben gearbeitet wird, ermöglichen die größeren Geräte eine Kontrolle der Klimabedingungen, d. h. von Temperatur, Sauerstoff- und CO_2-Gehalt. Dazu bietet der *IN Cell Analyzer 1000* Zusatzmodule zur Temperierung des Probentisches und zur Versorgung der Proben mit Nähr- bzw. Wirksubstanzen (ein Kanal Pipette). Der *ImageXpress 5000A* kann ebenfalls optional mit Modulen zur Temperierung (30°C - 40°C, ±0,5°C), zur Stabilisierung des CO_2-Gehalts (bei 5 %) und zur Stabilisierung des pH-Werts (bei pH 7,4 ±pH 0,1) sowie um ein Modul zur automatischen Beprobung erweitert werden.

(a) BD Pathway 415 (Becton Dickinson)

(b) IN Cell Analyzer 1000 (GE Healthcare)

(c) ImageXpress 5000A (Molecular Devices)

Abbildung 2.6: Beispiele für Mikrotiterplatten-Reader: *BD Pathway 415* von Becton Dickinson (Quelle: www.bdbiosciences.com), *IN Cell Analyzer 1000* von GE Healthcare (Quelle: www5.amershambiosciences.com), *ImageXpress 5000A* von Molecular Devices (Quelle: www.moleculardevices.com).

Die derzeit leistungsfähigsten optischen Reader erweitern die Weitfeld-Epifluoreszenz-Technik um die Möglichkeit 3-D Schichtaufnahmen zu machen. Dazu wird auf eine Nipkow-Konfokal-Technologie[7] zurückgegriffen. Beispiele für solche konfokalen Reader sind der *Pathway*

[7] Das Prinzip der Nipkow-Konfokal-Mikroskopie wird in Abschnitt 3.2.4 vorgestellt.

Bioimager HT von Atto Biosciences[8] und der *IN Cell Analyzer 3000* der Firma GE Healthcare[9]. Dem Ziel, die Vorteile der Mikroskopie und der Reader-Technologie in einem System zu vereinen, kommt gegenwärtig die Hamburger Firma Evotec Technologies mit ihrem konfokalen Imaging-Reader für Mikrotiterplatten *Opera* am nächsten (s. Abbildung 2.7). Ihr ebenfalls auf dem Prinzip der Nipkow-Konfokal-Mikroskopie aufbauendes Opera-System verwendet, wie das in dieser Arbeit beschriebene System auch, den Mikroskopkörper des iMIC (TILL-Photonics) als interne optische Plattform. Um bei gewünschtem Durchsatz (100.000 Proben am Tag) die Anzahl der detektierten Parameter durch Vielfarben-Imaging zu erhöhen, setzt das Opera System auf redundante Hardware und verwendet mehrere Kameras zur gleichzeitigen Aufnahme der Bilddaten [77]. Das gesamte System mit seinen komplexen Funktionen wird über einen PC gesteuert, der ebenfalls die *online* Bildauswertung und Archivierung übernimmt. Die fehlende Echtzeitfähigkeit des Windows-Betriebssystems limitiert die Einsatzmöglichkeiten des Systems sowie seine Adaptationsfähigkeit jedoch beträchtlich.

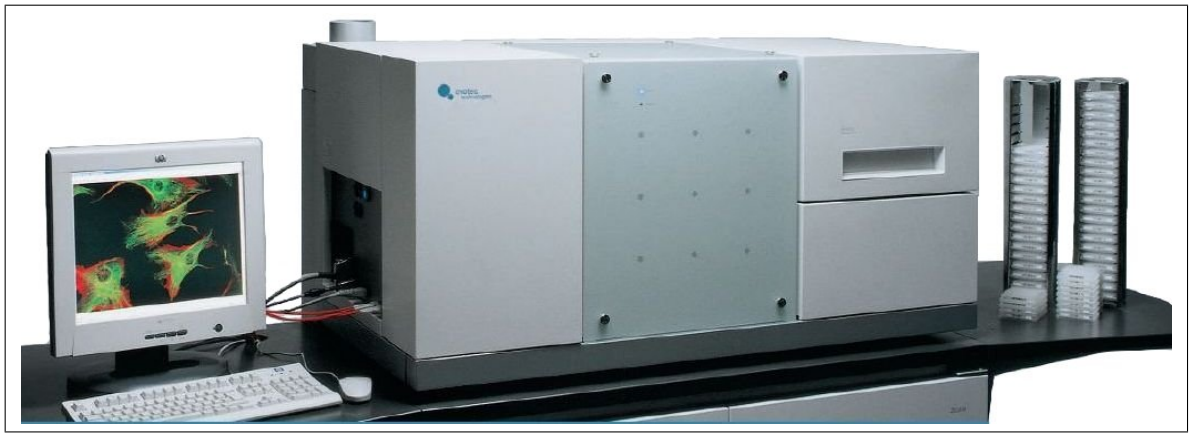


Abbildung 2.7: Konfokaler Mikrotiterplatten Reader *Opera* der Firma Evotec Technologies. Quelle: Evotec Technologies Internetauftritt: www.evotec-technologies.com

Um bei der parallelen Anwendung verschiedener Mikroskopietechniken zur Ermittlung möglichst vieler zellulärer Parameter eine hohe Flexibilität zu erhalten, sind sog. *High-Content-Reader* im Fokus kommerzieller Entwicklungen. Marktübliche Geräte basieren dabei überwiegend auf automatisierten Standardmikroskopen, die um Module zur automatischen Platzierung und zur automatischen Nähr- bzw. Wirkstoffversorgung der Proben erweitert werden. Die folgenden Abschnitte geben einen Überblick über komplexere Screeningsysteme auf Mikroskop-Basis:

Metafer - MetaSystems

Die Firma MetaSystems aus Altlussheim bietet ein automatisches Scan-System *Metafer* mit darauf aufbauender Bildanalysesoftware an. Ein motorisiertes Standardmikroskop kombiniert mit einem Imaging-System[10] bildet hierbei die Basis für die Digitalisierung der zu untersuchenden Proben [132, 134]. Auf dieser Basis aufbauend werden etliche Softwareprodukte zur

[8] Nach der Übernahme durch Becton Dickinson wird der Reader unter dem Namen *BD Pathway* geführt. Weitere Informationen befinden sich auf folgender Webseite: http://www.atto.com/products/pathway/.

[9] GE Healthcare ist ehemals Amersham Biosciences. Weitere Informationen über den Reader befinden sich auf folgender Webseite: http://www5.amershambiosciences.com/.

[10] Das Scanning-System *Metafer* verwendet das aufrechte Forschungsmikroskop *Axioplan2 Imaging Mot* von Carl Zeiss mit einem motorisierten Objekttisch der Firma Märzhäuser.

Auswertung und Analyse für verschiedene biologische und medizinische Anwendungen angeboten. Anwendungsbeispiele finden sich überwiegend auf dem Gebiet der Zytogenetik. Die Steuerung aller motorisierten Komponenten des Mikroskops und des Imaging-Systems erfolgt PC-basiert.

CytoScan - Applied Imaging

Nach ähnlichem Prinzip wie im soeben genannten Beispiel stellt das Scan-System *CytoScan* der Firma Applied Imaging aus San Jose, USA, die Basis zur Digitalisierung zellulärer Proben dar [132]. Mit der jeweils entsprechenden Software kombiniert erstreckt sich die Produktpalette von Komplettsystemen zur automatisierten Detektion und Analyse für eine Reihe von Anwendungen der Zytogenetik, der Pathologie und dem Bereich der Biopharmazeutik. Die Steuerung der Scan-Hardware ist auch hier PC-basiert.

scanˆR - Olympus BioSystems und EMBL Heidelberg

Basierend auf dem in Abschnitt 2.1.1 vorgestellten Imaging-System cellˆR, hat die Entwicklungssparte Olympus BioSystems (Planegg, Deutschland) in Zusammenarbeit mit dem EMBL (**E**uropean **M**olecular **B**iology **L**aboratory) in Heidelberg eine generische Mikroskop-basierte Screening Station entwickelt: *scanˆR*. Hierbei wurden alle beweglichen Funktionen motorisiert (z. B. durch den Einsatz eines motorisierten xy-Tisches[11]) sowie durch optionale Komponenten erweitert (z. B. um den Bestückungsroboter MicroLab SWAP der Firma Hamilton, Reno, USA).

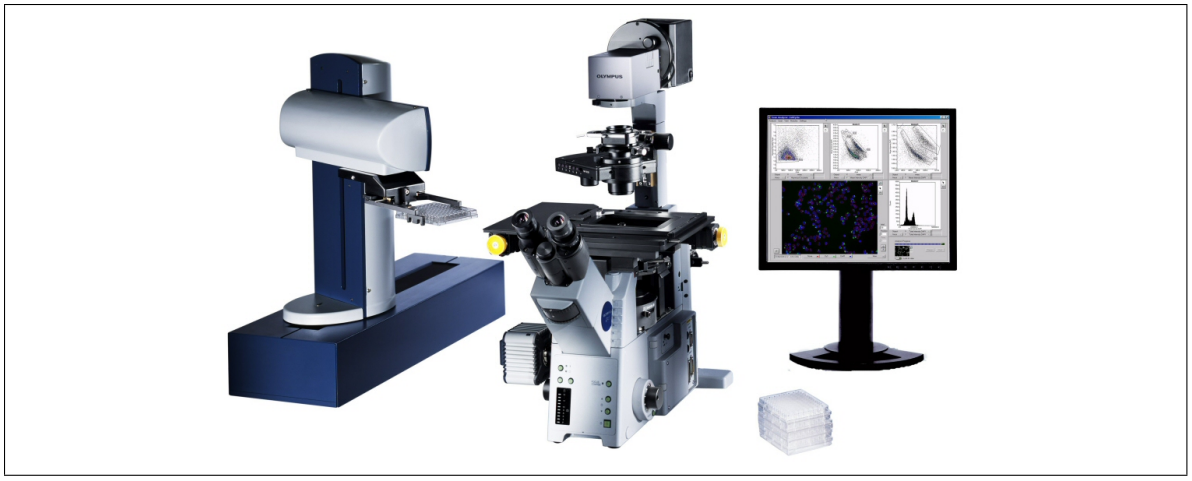


Abbildung 2.8: Screening Station basierend auf einer cellˆR Mikroskopiesystem mit Screening Erweiterung *scanˆR* von Olympus BioSystems. Quelle: Olympus Internetauftritt: www.olympus-europa.com/microscopy

Zur Steuerung der Screening Station wird die im Imaging-System cellˆR verwendete Echtzeiterweiterung von National Instruments in Verbindung mit der dazugehörigen LabVIEW-Software eingesetzt. Die Einführung einer im Rahmen des scanˆR-Systems angekündigten eigenen Echtzeitsteuerung wurde bis heute immer wieder verschoben, da die bestehenden Möglichkeiten den sich ständig verändernden bzw. wachsenden Anforderungen nicht gerecht werden.

[11] Eingesetzt wird (laut Olympus Spezifikationen) der xy-Tisch *SCAN IM IX2* der Firma Märzhäuser, Wetzlar, Deutschland.

Discovery-1 - Molecular Devices

Die bildbasierte **H**igh **C**ontent **S**creening-Plattform *Discovery-1* (s. Abbildung 2.9) der Firma Molecular Devices stellt neben der optischen Plattform iMIC der TILL Photonics (vgl. Abbildung 1.5) den einzigen Ansatz einer Mikroskop-basierten Screening-Station dar, die nicht auf einem Standard-Mikroskop mit herkömmlicher Bauform aufsetzt. Entgegen dem strikt modularen Prinzip des iMIC, über das die parallele Anwendung vieler verschiedener Mikroskopietechniken möglich ist, bietet der Reader Discovery-1 allerdings lediglich die Möglichkeit zur Auswertung von Mikrotiterplatten mittels Weitfeld-Epifluoreszenz-Imaging. Eine optionale Erweiterung ermöglicht zudem Durchlichtmikroskopie. Durch die Integration mit einem Bestückungsroboter, einem Barcode-Lesegerät etc. sowie durch die Anordnung mehrerer Discovery-1 Reader in einem Screening-Aufbau lassen sich vollautomatisierte Systeme aufbauen.

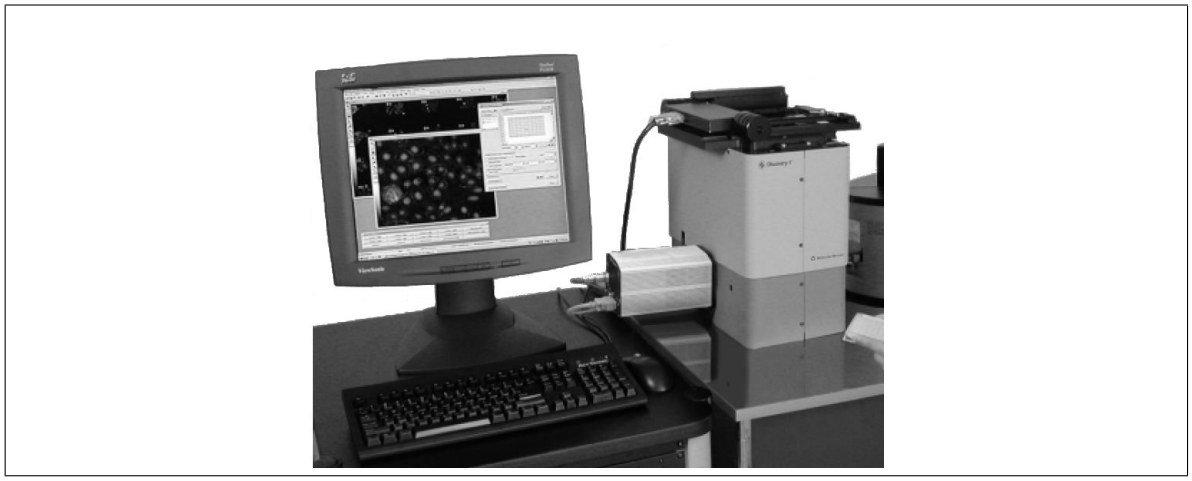


Abbildung 2.9: Bildbasierte **H**igh **C**ontent **S**creening-Plattform *Discovery-1* von Molecular Devices. Quelle: Molecular Devices Internetauftritt: www.moleculardevices.com

2.1.3 Analyse des Stands der Technik

Die Entwicklung kommerziell erhältlicher Mikroskope hat sich bisher überwiegend auf die Optimierung der herkömmlichen Bauweise konzentriert, sowie auf deren Erweiterung durch einzelne Module, die für die Vielzahl neuer Mikroskopietechniken (wie z. B. TIRF, FRET, FLIP, FRAP, KONFOKALES LASER-SCANNING) erforderlich sind. Die grundlegende Form des Mikroskopstativs blieb dabei seit seiner Erfindung im 16. Jahrhundert weitgehend unverändert. Daher sind selbst computergestützte *high-end* Mikroskope namhafter Hersteller a priori für die manuelle Bedienung konzipiert und für den Einsatz innerhalb vollautomatisierter Anwendungen nur mäßig gut geeignet. Der Computer dient vorwiegend der Bildaufnahme mit Hilfe einer Kamera und der Befehlseingabe durch den Nutzer, der in den meisten Fällen beim Experiment gegenwärtig sein muss. Bedingt durch die eingeschränkte Echtzeitfähigkeit (auf dem gewünschten Zeitraster) gängiger Computer-Betriebssystem gibt es immer wieder Verzögerungen bei der Übermittlung von Steuerbefehlen. Um sicherzugehen, dass alle Voraussetzungen für ein gegebenes Experiment erfüllt sind, müssen zeitliche Sicherheitsmargen vorgesehen werden. Diese führen nicht nur dazu, dass Messreihen länger dauern als eigentlich erforderlich, sie führen auch zu einer unnötigen Photonenbelastung der lebenden Untersuchungsobjekte, die in den meisten Fällen - vor allem wenn sie mit Fluoreszenzfarbstoffen markiert wurden - zum Ausbleichen

des Farbstoffs und zu phototoxischen Reaktionen führen. Eine zeitgenaue Synchronisation der verschiedenen Untersuchungsverfahren könnte diese Problematik entschärfen, doch wurde auf diese bei herkömmlichen Mikroskopen bisher zugunsten einer möglichst großen Vielseitigkeit verzichtet.

Getrieben von den immensen Investitionen der pharmazeutischen Industrie [37] hat die Entwicklung von vollautomatischen Analysegeräten für die pharmakologische Wirkstoffsuche in den letzten vier Jahrzehnten erhebliche Fortschritte gemacht [148, 67]. Vor allem die Miniaturisierung des Einzelexperiments und die damit einhergehende Parallelisierung der durchgeführten Versuche haben zu einem Durchsatz von bis zu 100.000 Proben und mehr geführt, die von einem einzelnen Analysegerät pro Tag untersucht werden. Solche im HTS eingesetzten *Reader* basieren dabei überwiegend auf einfachen photometrischen Fluoreszenztests: Liegt die Intensität des von der untersuchten Probe emittierten Fluoreszenzlichts über einem vorher definierten Schwellwert, so ist das Ergebnis positiv, ansonsten negativ. Photometrische Messungen ergeben damit lediglich die Erfassung von Effekten, die über die gesamte Probe integriert sichtbar sind. Einzeleffekte auf inter- bzw. innerzellulärer Ebene können somit i. d. R. nicht detektiert werden. In der unmittelbaren Vergangenheit gab es zwar Ansätze den Blick von Reader-Systemen derart „zu schärfen“, dass auch subzelluläre Domänen erfasst und charakterisiert werden können. Hierdurch wird ein differenzierteres Bild der zellulären Vorgänge erfasst, jedoch beschränken sich diese Systeme auf einfachere mikroskopische Standardverfahren und weisen bei weitem nicht die von Forschungsmikroskopen erreichte Leistungsfähigkeit auf. Durch die stetig zunehmenden Bedeutung von HCS [56, 57, 157, 30, 29, 21, 12] besteht allerdings ein Bedarf an leistungsstarken Analysegeräten zum hoch aufgelösten Screenen zellulärer Proben. Mit der für diese Dissertation zur Verfügung stehenden optischen Plattform gibt es erstmals die Möglichkeit, alle wesentlichen *high-end* Mikroskopieverfahren in ein einziges Gerät zu integrieren, in automatisierbarer Weise zwischen den verschiedenen Modi hin und her zu schalten und damit eine erschöpfende Charakterisierung automatisierungstauglich zu machen.

Ziel der vorliegenden Arbeit war, ein wissenschaftlich fundiertes, skalierbares und adaptierbares Steuerungskonzept für eine modulare Mikroskopieplattform, wie dem iMIC, zu entwickeln und zu realisieren. Entgegen einer industriellen Entwicklung war der wissenschaftliche Anspruch daher nicht nur alle unmittelbaren Anforderungen der derzeitigen Systeme zu erfüllen, sondern auch zu gewährleisten, das neue - heute noch nicht bekannte - Anforderungen die durch das Hinzufügen neuer noch unbekannter Funktionalitäten entstehen, ebenso erfüllt werden können. Gefordert war daher nicht nur die Möglichkeit zur Adaption des Ablaufs während eines Experiments an sich ändernde Bedingungen, sondern auch eine generelle Anpassungsfähigkeit an unterschiedliche experimentelle Fragestellungen und Vorgehensweisen.

2.2 Anwendungsgebiete der Echtzeitumgebung als Plattform für Systemintegration und Automatisierung

Als Grundlage zur Erarbeitung eines Steuerungskonzepts (Kapitel 3) und um die Anforderungen an Hardware (Kapitel 4), Firmware (Kapitel 5) und PC-Software (Kapitel 6) definieren zu können, sind Kenntnisse über die geplanten Anwendungsgebiete der Echtzeit-Plattform erforderlich. Im Folgenden werden deshalb typische Applikationen, Anwendungsbeispiele sowie die dazu notwendigen Geräte bzw. Systeme vorgestellt. Auf die gelisteten Beispielanwendungen

wird nicht ausführlich eingegangen, da Kenntnisse über die Details für das weitere Verständnis dieser Arbeit nicht zwingend erforderlich sind. Interessierte Leser seien allerdings auf die Einträge im Glossar sowie die angegebene Literatur hingewiesen.

2.2.1 Imaging-System - Digitale Bildaufnahme

Unter einem Imaging-System versteht man im Allgemeinen ein System zur Aufnahme, Archivierung, Verarbeitung und Auswertung von digitalen Bildern [68]. Neben dem Einsatz innerhalb der Qualitätsprüfung auf Gebieten wie den Materialwissenschaften und der Halbleiterindustrie werden Imaging-Systeme im Bereich der Life Sciences und in der Pharmaindustrie für die Mikroskop-basierte Zellforschung und bei Screening-Anwendungen eingesetzt.

Ein von der TILL Photonics vertriebenes Fluoreszenz-Imaging-System besteht im Einzelnen aus einem Detektor (üblicherweise einer CCD Kamera, gelegentlich aber auch aus einem oder mehreren PHOTOMULTIPLIERN), einer monochromatischen Beleuchtungsquelle (Polychrome V), einer Kontrolleinheit und der entsprechenden PC-Software (TILLVISION). Um die Einflüsse (Erwärmung, Phototoxizität, etc.) der Beleuchtung auf die untersuchten Proben zu minimieren, werden die Messzeit durch den Detektor und die Beleuchtungszeit optimal aufeinander abgestimmt. Die hierfür notwendige hardwareseitige Synchronisation erfolgte bisher durch eine Kontrolleinheit, die speziell für den Einsatz im TILL Imaging-System konzipiert worden und in Bezug auf den Ausbau des Systems nicht erweiterbar ist. Die im Rahmen dieser Arbeit entstandene Echtzeitplattform (**I**ntegration **C**ontrol **U**nit (ICU)) soll diese Limitation beseitigen und die bisher verwendete Kontrolleinheit ersetzen.

Anwendungsbeispiele für das Imaging-System:

- Messung intrazellulärer Ionenkonzentrationen (Kalzium, Magnesium, Kalium, pH-Wert, etc.) [144, 17, 84, 160] mittels ratiometrischen Farbstoffen wie fura-2, furaptra, BCECF oder Einwellenlängen-Farbstoffen wie calcium green, fluo-3,
- kombiniertes Messen von z. B. Kalzium und dem Grün-Fluoreszierenden-Protein (GFP) [163, 168, 159],
- Mehrfarben-Imaging zur gleichzeitigen Detektion verschiedener fluoreszierender Proteine [64, 75].

2.2.2 Computer-gestützte, automatisierte Mikroskopie

Moderne high-end Lichtmikroskope bieten überwiegend die Möglichkeit der digitalen Bildaufnahme. Hierdurch wird die PC-basierte Dokumentation, Analyse und/oder Archivierung der im Verlauf eines Experiments digitalisierten Bilder ermöglicht. Weiterhin können bei Verwendung von digital steuerbaren Mikroskopkomponenten und/oder Zubehör, wie z. B. motorisiert verfahrbaren XY-Tischen, motorisierten Fokussiereinheiten oder steuerbaren Durchflusseinrichtungen die Experimentabläufe über den Anwender-PC vordefiniert und anschließend automatisch ausgeführt werden. Dies führt zu technisch 100 % reproduzierbaren Experimenten bzw. Assays und ermöglicht so z. B. die Vernetzung von räumlich getrennten Arbeitsgruppen mittels vereinheitlichten, digitalen Experimentbeschreibungen.

Kombiniert mit dem im vorherigen Abschnitt vorgestellten Imaging-System stellt das iMIC ein komplettes, voll-automatisiertes, computergestütztes Mikroskopiesystem dar (Abbildung 2.10).

Der strikt modulare Aufbau sowohl des iMIC als auch des Gesamtsystems ermöglicht einen flexiblen Einsatz verschiedener Mikroskopietechniken und zusätzlich die Integration der einzelnen Module oder des kompletten Systems in größere, bestehende Aufbauten bzw. Produktionsstraßen. Die bereits im Imaging-System eingesetzte ICU soll innerhalb des Mikroskopiesystems zusätzlich die Steuerung der motorisierten Komponenten des iMIC übernehmen und diese mit den Abläufen der digitalen Bildaufnahme synchronisieren.

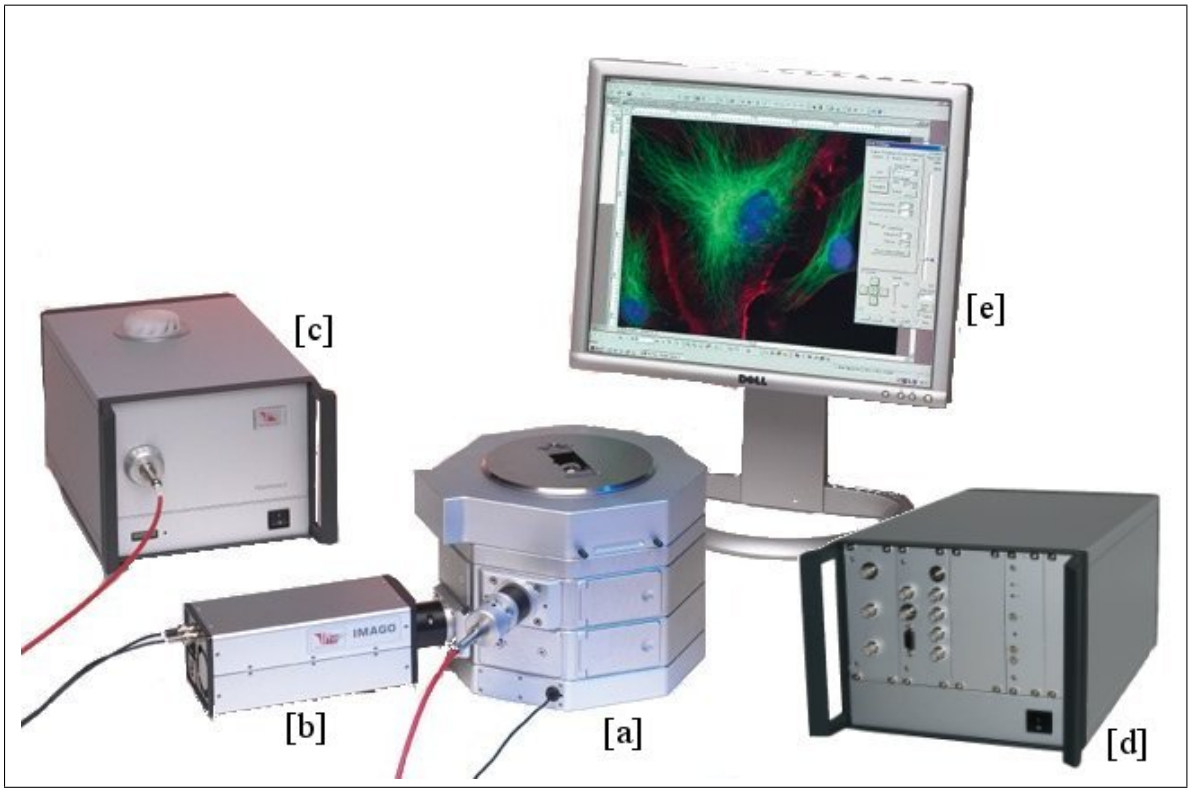


Abbildung 2.10: TILL Photonics PC-gestütztes, automatisiertes Mikroskopiesystem: Mikroskop iMIC [a] mit CCD-Kamera [b], Lichtquelle Polychrome V [c], Kontrolleinheit ICU [d] und PC-Software TILLvisION (Screenshot) [e].

Die modulare Bauart und die kompromisslose Führung der Strahlengänge innerhalb des iMIC erlauben einen direkten Zugriff auf die zentral im Mikroskopkörper gelegene, optische Achse und ermöglichen damit moderne Mikroskopietechniken, wie z. B. :

- **T**OTAL **I**NTERNAL **R**EFLECTION **F**LUORESCENCE (TIRF) [8, 9],
- **F**LUORESCENCE **R**ESONANCE **E**NERGY **T**RANSFER (FRET) [156, 124, 137, 70, 15],
- **F**LUORESCENCE **L**OSS **I**N **P**HOTOBLEACHING (FLIP) [145, 162],
- **F**LUORESCENCE **R**ECOVERY **A**FTER **P**HOTOBLEACHING (FRAP) [181, 105, 22, 126],
- NIPKOW KONFOKALMIKROSKOPIE [113, 42],
- LASER-SCANNING-MIKROSKOPIE [34, 139],
- MULTI-PHOTONEN LASER-SCANNING-MIKROSKOPIE [35, 176, 87, 151].

Die meisten der aufgelisteten Mikroskopietechniken erfordern eine computergestützte Bildaufnahme und/oder Auswertung: Beim Einsatz der FRET-Technik oder bei der Laser-Scanning-Mikroskopie z. B. müssen die aufgenommen Daten rechnerisch zu einem Bild zusammengesetzt werden. Zudem erfordern Analyseroutinen wie die Kinetikanalyse [127, 78, 130] eine

PC-gestützte Verarbeitung der Bilddaten. Auch für eine gute zeitliche Koordination der Systemkomponenten sind digitale Systeme von entscheidender Bedeutung, z. B. bei der bereits genannten Synchronisation zwischen CCD-Auslese und verschiedenen Beleuchtungsmodi.

Beispielanwendungen für die computergestützte, automatisierte Mikroskopie:

- Zeitaufgelöste Beobachtung und quantitative Messung intrazellulärer Signale (z. B. Kalzium) [17],
- Analyse von Zellwachstum, Zellteilung, Apoptose (physiologischer Zelltod) [72],
- Aufnahme von Bildstapeln zur Gewinnung dreidimensionaler Daten [104, 87],
- multispektrale Messungen (in 2 und mehr Dimensionen) zur differentiellen Unterscheidung zellulärer Strukturkomponenten,
- Beobachtungen und Echtzeit-Analyse von Zellen und Geweben, die zur Steigerung des Durchsatzes mit gleichbleibender Geschwindigkeit relativ zum Objektiv bewegt werden.

Von den aufgelisteten Techniken soll lediglich auf die Laser-Scanning-Verfahren und die **T**ime **D**elayed **I**ntegration (TDI)-Verfahren näher eingegangen werden, da ein prinzipielles Verständnis im Weiteren zur Erarbeitung der Konzepte und zur Ableitung der Anforderungen an Hard- und Firmware hilfreich ist. Zudem lassen sich anhand dieser anspruchsvollen Beispiele die Leistungsanforderungen an die ICU am besten definieren.

Time Delayed Integration

TDI stellt eine spezielle Art des Auslesens von CCD-Chips dar[12], die für die Inspektion bewegter Objekte genutzt werden kann [74]. Bei der regulären Bildaufnahme eines feststehenden Objekts wird der komplette CCD-Chip als ein einzelnes komplettes Bild nach einer gewissen Belichtungszeit ausgelesen. Beim TDI-Verfahren wird stattdessen die Geschwindigkeit eines bewegten Objekts mit dem Verschieben der Ladungen auf dem CCD-Chip synchronisiert. Die von einer Zeile zur nächsten verschobenen Ladungen der parallelen CCD-Zellen werden so jeweils aufsummiert. Anstelle des kompletten CCD-Chips wird beim TDI-Verfahren nach jedem Verschieben der Ladungen lediglich die letzte Zeile ausgelesen. Solange die Bewegung des Objekts und die Ladungsverschiebung synchron sind, wird ein Bild über einen kontinuierlichen Datenstrom in Form eines Bildstreifens ausgelesen. Je genauer die Bewegungsgeschwindigkeit und die Ladungsverschiebung auf einander abgestimmt sind, desto schärfer wird der aufgenommene Bildstreifen. Durch die damit ermöglichte verlängerte Belichtungszeit kann bei gleichem Signal/Rauschabstand mit wesentlich geringeren Lichtintensitäten gearbeitet werden als bei der Bildaufnahme mit den üblicherweise eingesetzten Zeilensensoren [117, 100]. Anders als bei der ebenfalls häufig eingesetzten stop-and-go Bildaufnahme ergeben sich durch eine kontinuierliche Bewegung des Objektes zusätzliche Vorteile: Große Objekte (z. B. Gewebeschnitte oder Multititerplatten) können schneller „gescannt“ werden, ohne dass die empfindliche lebende Probe häufig durch ein Bremsen und Beschleunigen „irritiert“ wird. Zudem ergeben sich keine Totzeiten zwischen den einzelnen Bildaufnahmen.

Abbildung 2.11 veranschaulicht das TDI-Prinzip anhand einer Serie von einzelnen, sequenziellen Momentaufnahmen zu den Zeitpunkten T1 bis T9. Jedes Einzelbild besteht aus vier Teilbe-

[12] Nicht alle CCD-Kameras unterstützen den TDI-Modus.

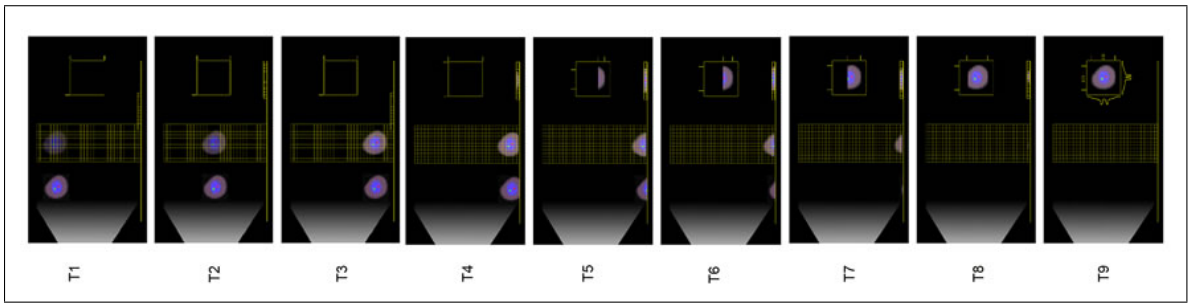


Abbildung 2.11: Graphik zur Veranschaulichung des TDI-Verfahrens; Quelle: [3]

reichen: Die Beleuchtungsquelle, ein von links nach rechts bewegtes Objekt (Zelle), den CCD-Detektor, auf den die Zelle projiziert wird, und den Bilddatenspeicher.

Das TDI-Verfahren wird in folgenden Gebieten angewendet:

- Schnelles Scannen von großen Proben (Gewebe, Multitierplatten),
- Durchfluss-Cytometrie (z. B. amnis - ImageStream).

Konfokale Laser-Scanning-Mikroskopie

Konfokales Laser-Scanning ist ein spezielles Verfahren zur Gewinnung von Schichtaufnahmen und der Erzeugung dreidimensionaler Bilder aus Schichtaufnahmen-Bildstapeln. Das Prinzip des konfokalen Laser-Scannings beruht darauf, dass ein fokussierter Laserstrahl über eine Probe gescannt wird und das emittierte Fluoreszenzlicht durch eine kleine, sog. konfokale Punktblende gefiltert und dadurch von „out-of-focus“ Bildinformation befreit wird. Dies ermöglicht die Aufnahme von Schnittbildern nur aus der jeweiligen Fokusebene.

Durch Ändern der Fokussierung zwischen einzelnen Aufnahmen kann man einen ganzen Bildstapel aufnehmen und erhält so einen 3D-Datensatz. Die Bewegung des Laserstrahls mittels einer zweidimensionalen Scan-Vorrichtung (Scankopf) und die Detektion des Emissionslichts müssen präzise synchronisiert sein, damit eine spätere räumliche Zuordnung der einzelnen Messpunkte für die Rekonstruktion des Bildes möglich ist.

Die für ein Laser-Scanning-Mikroskop benötigten Komponenten werden an einer späteren Stelle detailliert vorgestellt (s. Abschnitt 3.2.4).

2.2.3 Screening-Anwendungen

In Kapitel 1 wurden in den Beschreibungen der einzelnen Phasen einer Medikamentenentwicklung bereits IT und ZB Screening-Verfahren vorgestellt. Die innerhalb der Screens eingesetzten Geräte zum Auslesen und Analysieren von Mikrotiterplatten, die so genannten Reader, sind überwiegend in sich abgeschlossene, vollautomatische Laborgeräte. Mit *einer* Auslesetechnik ausgestattet ist ein Reader für die Durchführung jeweils einer bestimmten Art von Assay vorgesehen. Der Bedarf an effektiven Analysegeräten für den Bereich des ZB-Screening und der zunehmende Trend zu HCS mit bis zu sub-zellulärer und damit mikroskopischer Auflösung wurde bereits beschrieben. Herkömmliche Mikroskope, obwohl teilweise mit motorisierten Komponenten ausgestattet, sind allerdings nur bedingt für automatisierte Anwendungen und die Integration in HCS-Systeme geeignet: Wenngleich die Mikroskopie seit ihrer Erfindung im Jahr 1590 [103, 98] über die Verbesserung der optischen Komponenten und Entwicklung neuer Techniken und Methoden stetig leistungsfähiger geworden ist, blieb der prinzipielle Aufbau

der Mikroskope nahezu unverändert. Durch die traditionelle Bauform der Stative, die unvermeidliche Integration von Okularen und durch üblicherweise vorgefundene Halterungen für die Durchlichtbeleuchtung wird der automatisierte Umgang mit Proben durch Platzierungsroboter oder Pipettiersysteme und damit die Integration eines Mikroskops in ein Screening-Gerät erheblich erschwert.

Wie in Abschnitt 2.2.2 beschrieben, wurde das im Mikroskopiesystem von TILL Photonics eingesetzte iMIC auf die Anforderungen von automatisierten Systemen hin optimiert. Da die Bildaufnahme CCD-basiert ist und die Ausgabe über den PC-Monitor erfolgt bzw. an automatische Auswerteverfahren geleitet wird, konnte auf ein Okular verzichtet werden. Darüber hinaus kann eine Konstruktion zum Positionieren der Beleuchtungsquelle für Durchlicht-Mikroskopie [149] i. d. R. ebenfalls weggelassen werden, da das INVERSE MIKROSKOP iMIC vorrangig für fluoreszenzmikroskopische Anwendungen entwickelt wurde. Dies führt dazu, dass der Bereich oberhalb des Objekttischs komplett unbehindert und damit für manuelles und/oder automatisches Bewegen oder Bearbeiten bzw. Manipulieren der Proben frei zugänglich ist. Damit eignet sich das iMIC als Basis für optische Anwendungen jeder Art, für den Einsatz innerhalb automatisierter Anwendungen und insbesondere für die Integration in vollautomatische Laborgeräte bzw. Reader[13]. Für einen möglichst hohen Probendurchsatz (s. HTS) ist das optimale Zusammenspiel aller Funktionen eines Readers von entscheidender Bedeutung. Daher muss die ICU als die zentrale Steuerungseinheit des Mikroskopiesystems die Möglichkeit bieten, dass die Abläufe der übrigen Reader-Komponenten (Platzierungs- und Beprobungsautomatik) mit denen des Mikroskopiesystems synchronisiert werden können (und vice versa).

Durch die mögliche Anwendung verschiedenster Mikroskopietechniken (s. Abschnitt 2.2.2) und der Möglichkeit ohne Umbau des Systems zwischen den Anwendungen der einzelnen Techniken umzuschalten, bietet ein Mikroskop-basierter Reader folgende Eigenschaften:

- Optische Auflösung bis auf sub-zelluläre Ebene (Die Auflösung wird durch das verwendete Objektiv bestimmt.),
- große örtliche Flexibilität durch einen frei verfahrbaren Objekttisch (Hierdurch besteht keine Limitation bei der Auswahl der verwendeten Mikrotiterplatte[14], da die angefahrenen Messpositionen softwareseitig definiert werden können.),
- flexibler Einsatz von modernen, optischen Screeningmethoden,
- bildbasierte Analyse,
- parallele Anwendung verschiedener Mikroskopie- und Analysetechniken,
- flexibler Einsatz von modernen, optischen Screeningmethoden.

2.2.4 Integrierte Systeme für Langzeitanalyse

Im Zuge der Optimierung von ZB-Assays ist ein Trend zu einer höheren Anzahl der untersuchten Parameter und zur Parallelisierung der Datenaufnahme innerhalb eines durchgeführten Versuchs bzw. Screens zu beobachten (s. Kapitel 1). Zudem nimmt die Bedeutung von Langzeituntersuchungen zur kontinuierlichen Beobachtung des lebenden Zellmaterials innerhalb der

[13] Das iMIC-"Gehäuse" wird derzeit bereits im Konfokal Fluoreszenz Mikrotiterplatten Reader „Opera" der Firma Evotec Technologies, Hamburg, Deutschland, eingesetzt; Das Opera-System wird z. B. im Bereich Ultra-High–Throughput Screening eingesetzt [119].

[14] Durch den Arbeitsabstand des eingesetzten Objektivs ist lediglich die Dicke des Well-Bodens eingeschränkt.

biologischen und medizinischen Grundlagenforschung, der klinischen Diagnostik und Anwendungen des Medikamenten-Screenings zu. Als Basis für solche Untersuchungen sind neben einer minimalinvasiven Parameteraufnahme geeignete Methoden notwendig, durch welche die in-vitro-Umgebung der Proben (Kultur-Schalte oder Wellplatte) möglichst genau die in-vivo Bedingungen simuliert. Technische Systeme für Langzeitmessungen dieser Art integrieren daher neben verschiedenen Komponenten zur Aufnahme von Messdaten auch Baugruppen zur Versorgung der Proben mit Nährlösungen bzw. Wirkstoffen und zur Anpassung und Konstanthaltung der Umgebungsparameter.

Vor diesem Hintergrund wird derzeit in enger Zusammenarbeit des LME mit mehreren industriellen Partnern[15] u. a. an dem modularen Gesamtsystem IMR (s. Abbildung 1.4) gearbeitet, das erstmals die mannigfaltigen Möglichkeiten optischer Analyseverfahren mit denen bioelektrischer Sensortechnologie vereint [95, 180, 54, 53]. Speziell für den Einsatz im Bereich Life Science und hier insbesondere für Langzeitmessungen wurden modulare Erweiterungen wie ein automatisierbares Fluidiksystem zur Versorgung von lebenden Proben mit Nähr- und Wirkstofflösungen und ein Klima-Kontrollsystem entwickelt. Hierdurch wird eine kontinuierliche multi-parametrische Beobachtung lebender Zellen für die detaillierte Analyse intra- und extrazellulärer Vorgänge ermöglicht.

Basierend auf über zehn Jahren Erfahrung auf dem Gebiet bioelektronischer Sensortechnik wurde am LME eine mikroskopierbare Multititerplatte mit integrierten Mikrosensorarray-Messsystemen entwickelt [129, 128]. Konzipiert für Langzeitanalysen an lebenden Zellen integriert die Multititerplatte zusätzlich Mikrofluidikeinheiten für einen geregelten Austausch von Kulturmedien zur Versorgung der Zellen mit Nährstoffen und zum Abtransport von Metaboliten einerseits und zur genau dosierten Zugabe von Wirkstoffen andererseits [95, 96]. Der kontrollierte Medium-Austausch wird mit Hilfe von Ausgleichsbehältern ermöglicht (Drei-Kammer-System, s. Abbildung 2.13(a)), die wiederum von einem Pipettierroboter (s. späteren Abschnitt 3.2.6) versorgt werden. Über mikrostrukturierte Kanäle im Drei-Kammer-System kommt es zu einem reproduzierbaren Mediumaustausch, bei dem Nähr- bzw. Wirkstofflösung durch hydrostatische Druckdifferenzen vom Zulaufbehälter über den Kulturbereich in den Ablaufbehälter fließt (Prinzip der kommunizierenden Röhren). Durch Einführen eines passgenauen Zylinders von oben in den Zelltopf, kann ein definiertes Messvolumen eingestellt werden. Schließlich wird nur über die Einstellung eines ausreichend kleinen Mikro-Reaktionsvolumens im Bereich der Zellkultur die Messung von Stoffwechselraten (z. B. extrazelluläre Ansäuerung, Sauerstoffverbrauch) an kleinen Zell- und Gewebekulturen möglich. Den unteren Abschluss des Zellkulturbereichs bildet ein mikroskopierbarer Glas-Sensorarraychip. Diese Einheiten werden auf dem Format einer 24-Well Platte angeordnet. Leiterbahnen an der Unterseite der Multititerplatte dienen der Signalführung und ermöglichen eine einfache Kontaktierung an deren Längsseite (s. Abbildung 2.13(b)).

[15] Projektpartner:
Steinbeiszentrum Lab-on-Chip-Systeme, Bernried;
TILL Photonics, Gräfelfing;
RAWE Electronic GmbH, Weiler;
PreSens - Precision Sensing GmbH, Regensburg;
MicroCoat Biotechnologie GmbH, Bernried;
Heraeus Holding GmbH, Hanau;
Thermo Electron (Oberschleißheim) GmbH (ehemals H&P Labortechnik AG), Oberschleißheim.

(a) Mikrotiterplatte mit Fluidik (b) Unterseite (c) Sensorchip

Abbildung 2.12: Intelligente Multititerplatte mit integrierten Fluidikeinheiten und Sensorarraychip bestückter, mikroskopierbarer Unterseite [95, 96]; Quelle: LME

Die derzeit verwendeten Sensorchips (s. Abbildung 2.13(c)) beinhalten **I**nter**D**igitated **E**lectrode **S**tructures (IDES) für Impedanzmessungen, die Rückschlüsse auf die Anhaftung und morphologische Veränderungen adhärenter Zellkulturen erlauben [59, 51, 43, 44]. Der strukturelle Aufbau des IDES ist ein Finger-Kondensator mit 50 µm Strukturbreite, auf den ein schwacher Wechselstrom bei einer Frequenz von 10 - 20 kHz aufgeprägt wird [43, 44]. Aufgrund der isolierenden Eigenschaften von Zellmembranen innerhalb dieses Frequenzbandes nimmt die Impedanz mit steigender Zell-ADHÄRENZ zu und ist somit ein Maß für Zellausbreitung, ADHÄSION und Umordnung des Cytoskeletts in Verbindung mit Zell-Zell- und Zell-Matrix-Bindungen [44]. Zusätzlich zu den IDES sind zwei optische Sensoren der Firma PreSens GmbH, Regensburg, für die Messung von Sauerstoff und pH-Wert auf den Sensorchips aufgebracht. Die extrazelluläre Ansäuerung kann hiermit in einem Bereich von pH 5 bis pH 8 bis auf eine Auflösung von pH 0,01 und einer Genauigkeit von pH 0,05 gemessen werden. Mittels der optischen Sauerstoffsensoren kann die Sauerstoffsättigung bzw. der Sauerstoffpartialdruck des gelöst-Sauerstoffs innerhalb des extrazellulären Mediums von 0 % bis 150 % mit einer Auflösung von mindestens 3 % bei einer Genauigkeit von ca. 5 % gemessen werden. Diese beiden optischen Sensoren basieren auf fluoreszierenden Farbstoffen, die auf der Sensoroberfläche befestigt werden und in unmittelbarem Kontakt mit den Zellkulturen und dem Medium stehen. Die von der Sauerstoffkonzentration und dem pH-Wert der Sensorumgebung abhängigen Änderungen der fluoreszierenden Farbstoffe [36, 78, 79, 80] können über optische Verfahren wie z. B. über die Fluoreszenzmikroskopie ausgelesen werden. Über die beschriebenen optischen Sensoren können die Zellatmung, die Ansäuerung des Mediums und damit der Stoffwechsel der Zellkultur überwacht werden. Derzeitige Arbeiten konzentrieren sich auf die Vergrößerung der Well-Dichte (Ziel: 96-Well Platte). Zudem könnten anstelle der optischen Sensoren zur Messung der Sauerstoffkonzentration planare Sauerstoffsensoren [173, 172] und zur pH-Wert-Messung Metalloxidsensoren [47, 182, 170] verwendet werden.

Für die Anwendung optischer Analysemethoden wird das bereits in Abschnitt 2.2.2 beschriebene automatisierte Mikroskopiesystem eingesetzt. Das hohe Maß an Integration des iMIC führt im Vergleich zu herkömmlichen Mikroskopen zu einer erheblichen Reduktion des Volumens und des Gewichts. Durch Montieren des Mikroskop-Bausteins auf einem motorisiert verfahrbaren Sockel ist man somit erstmals in der Lage das komplette Mikroskop unter einem feststehenden Objektivtisch und damit relativ zur Probe zu verfahren. Die Sensorik, die Beprobungsautomatik und die Klimastatisierung ist dadurch komplett vom Mikroskopiesystem entkoppelt.

Der hier vorgestellte IMR ist ein erstes Beispiel für die angestrebte Integration des Mikroskopiesystems und anderer Komponenten in ein übergeordnetes Gesamtsystem. Wie im Abschnitt über Screening-Anwendungen (Abschnitt 2.2.3) bereits beschrieben wurde, soll die ICU als Steuereinheit des Mikroskopiesystems ebenfalls Möglichkeiten zur Synchronisation der übrigen Systemkomponenten bieten.

Beispielanwendungen für integrierte, multi-parametrische Systeme für Langzeitanalyse:

- „Life Cell Oberservation“/„In Vitro Monitoring“,
- Identifikation und Evaluierung von Targets für Screening-Anwendungen,
- ZB-Screening,
- patientenorientierte Diagnostik und Medikamentierung.

2.2.5 Mikroskop-basiertes „Machine-Vision“

Der Begriff „Machine-Vision“ beschreibt im Allgemeinen die Nutzung von Informationen, die - aus der Bildanalyse gewonnen - dem automatischen Eingreifen bzw. Ändern von Ablaufsteuerungen einer Maschine während deren Laufzeit dienen. Dabei kommen Methoden aus dem Gebiet der COMPUTER VISION zur Verarbeitung und Analyse der von bildgebenden Geräten ermittelten Daten zum Einsatz, um Bildinhalte zu extrahieren und/oder zu erkennen. Auf Basis dieser Informationen werden automatisch Entscheidungen generiert, die entweder einzelne Aktionen der „Maschine“ oder angepasste bzw. komplett neue Abläufe zur Folge haben. Derzeitige Anwendungsgebiete von Machine-Vision-Systemen umfassen die Ermittlung von Stückzahlen in industriellen Produktionen, die Qualitätskontrolle, Sicherheitssysteme (z. B. beim Einsatz von Industrierobotern oder fahrerlosen Transportfahrzeugen), die visuelle Lagerverwaltung und Verkehrsleitsysteme mit automatischer Ermittlung der Verkehrsdichte (Anwendungsbeispiele: [158, 121, 81, 152, 1, 153]).

Wie in den vorangegangenen Abschnitten erläutert, soll ein auf der Echtzeitplattform ICU aufbauendes Mikroskopiesystem die Möglichkeit zur vollständig automatisierten Ausführung von vordefinierten Abläufen bieten. Durch unmittelbare Anzeige der Bilddaten bzw. der Analyseergebnisse während der Laufzeit soll ein Benutzer in der Lage sein, in den von ihm vordefinierten Ablauf einzugreifen und Anpassungen vorzunehmen (Interaktivität). Zudem soll über eine PC-basierte Bildanalyse das automatische Anpassen der Abläufe bzw. eine automatische Reaktion des Mikroskopiesystems hervorgerufen werden können (Adaptivität).

Die Funktionalität der ICU muss die gewünschte Interaktivität und Adaptivität des Gesamtsystems ermöglichen. Hierfür müssen einerseits Mechanismen vorhanden sein, die erlauben, die innerhalb eines vordefinierten Ablaufprotokolls verwendeten Parameter während der Laufzeit anzupassen (z. B. die Belichtungs- bzw. Beleuchtungszeit einer Bildaufnahme). Andererseits müssen, abhängig von während der Laufzeit auftretenden Ereignissen, komplette Änderungen des Ablaufs möglich sein.

Beispielanwendungen für ein Mikroskop-basiertes Machine-Vision-System:

- Protokoll-gesteuertes, Mikroskop-basiertes Imaging mit automatischer Anpassung
  - der Belichtungsdauer oder Beleuchtungsintensität bei ausbleichenden Proben,
  - des Fokus beim Mikroskopieren,

    - der Objekttischposition bei sich bewegenden oder wachsenden Objekten/Zellen;
- automatisches Absuchen des Probenbereichs nach
    - bestimmten Merkmalen (z. B. Veränderungen der Probenvitalität innerhalb von Screening-Anwendungen),
    - Zellen (in Verbindung mit einer OPTISCHEN PINZETTE: „Zellsorter“) [6, 5, 142];
- Synchronisation auf Ereignisse in der Zelle, wie z. B. : die Reaktion auf Veränderungen bestimmter Zellparameter.

3 Konzept zur Steuerung integrativer, modularer Systeme

Vor dem Hintergrund des bisher beschriebenen Kontextes dieser Dissertation und der soeben dargestellten Anwendungsgebiete der Echtzeitplattform, wird im Folgendem ein Konzept zur Steuerung von modular aufgebauten, integrativen Systemen vorgestellt. Nach der Zusammenstellung allgemeiner Anforderungen werden konkrete Anforderungen anhand eingehender Vorüberlegungen zu den einzelnen Systemkomponenten erarbeitet, auf deren Basis das Konzept aufbaut.

3.1 Generelle Anforderungen an das Steuerungskonzept

Bereits aus einer groben Betrachtung der in Kapitel 2 vorgestellten Anwendungen und der jeweils verwendeten Systeme bzw. Systemkomponenten können Anforderungen an das Steuerkonzept abgeleitet werden. Im Allgemeinen werden Labor- bzw. Analysegeräte zunächst für den Betrieb als Einzelapparate entwickelt, um in den entsprechenden Marktsegmenten vertrieben zu werden. Je nach Komplexitätsgrad werden die Geräte für diesen Zweck zunehmend mit Standardschnittstellen zwecks Konfiguration oder Bedienung über einen PC ausgestattet. Für zeitkritische Anwendungen werden zusätzliche Schnittstellen integriert, die der Synchronisation mit anderen Baugruppen dienen. Nahezu alle Komponenten der in Kapitel 2 vorgestellten Systeme können hierfür als Beispiel betrachtet werden. Ein Konzept zur Steuerung mehrerer, autonomer, in einem Verbund zusammengefügter Geräte sollte diese aus der Geräteentwicklung stammenden Eigenschaften berücksichtigen, um die Funktionalität der Einzelkomponenten möglichst optimal auszunutzen und im Gesamtkontext sinnvoll zu kombinieren.

In allen vorgestellten Anwendungen werden jeweils diverse Geräte bzw. verschiedene Kombinationen von Geräten in einem Gesamtaufbau betrieben. Die Struktur der Gesamtsysteme ist demnach in allen gezeigten Beispielen strikt modular. Vor allem in Einsatzbereichen der Grundlagenforschung können beliebig viele Szenarien entstehen, bei denen zusätzliche Geräte in bereits bestehende Messaufbauten integriert werden sollen. Ein ganzheitliches Steuerungskonzept muss daher Möglichkeiten bereitstellen, damit ein bestehendes System ohne großen technischen Aufwand um zusätzliche Geräte erweitert werden kann.

Die beschriebenen Anwendungsbeispiele zeichnen sich durch hohe Anforderungen an ein präzises zeitliches Verhalten der Gesamtsysteme aus. Insbesondere Anwendungen auf den Gebieten Screening (Abschnitt 2.2.3) und Mikroskop-basiertem Machine-Vision (Abschnitt 2.2.5) benötigen möglichst optimale Ausnutzung des Funktionsumfangs und ein optimales Zusammenspiel aller Einzelkomponenten. Unnötige Zeitverluste durch unzureichende Synchronisation der Komponenten ist hier in den Regel nicht tolerabel. Zusätzlich zu der Optimierung der zeitlichen Abläufe für Hochdurchsatz-Anwendungen (high-throughput) zeigen die Beispiele der Bildaufnahme mittels TDI-Verfahren (s. Abschitt 2.2.2) und Laser-Scanning-Mikroskopie (s. Abschnitt 2.2.2), dass die Synchronisation von mehreren, an sich autarken Geräten eine grundsätzliche Voraussetzung der Funktionalität einiger Systeme ist. Das Steuerungskonzept muss diese Anforderung und die damit verbundene Echtzeit-Charakteristik der Systeme berücksichtigen und entsprechende Möglichkeiten der Synchronisation aller zeitkritischen Abläufe vorsehen.

Bei der Fülle der potenziell in einem Gesamtaufbau verwendbaren Geräte kann im Zuge der

Konzeption der Systemsynchronisation nicht explizit bestimmt werden, welches Gerät (Master) die Synchronisation der restlichen Komponenten (Slaves) übernimmt. Prinzipiell muss vielmehr davon ausgegangen werden, dass je nach Anwendung die Synchronisation von unterschiedlichen Geräten ausgeht. Auch sind Systeme denkbar, in denen die Rolle des Masters während der Laufzeit in einem Wechselspiel von einem Gerät an ein anderes übergeben wird (Multi-Master-System). Als Beispiele seien hier alle Systeme genannt, in denen einzelne Geräte lediglich über den Anwender-PC betrieben werden können und Hardwaresynchronisation dieser Komponenten daher nicht möglich ist. Für die Zeit der Bedienung eines solchen Gerätes muss der PC die Rolle des Systemmasters übernehmen und anschließend für hardwaresynchronisierte Abläufe wieder abgeben. Ein Steuerungskonzept muss daher die komplexe Thematik eines Multi-Master-Systems berücksichtigen und die denkbaren Kombinationen abdecken.

Nachdem die angestrebten Anwendungsgebiete auf Aspekte der Steuerung hin betrachtet worden sind, können die generellen Anforderungen an das Steuerungskonzept einer modularen und integrativen Plattform für ein Machine-Vision System folgendermaßen zusammengefasst werden:

- Ausnutzung gerätespezifischer Merkmale,
- modularer Aufbau/Möglichkeit der Erweiterung,
- deterministisches Zeitverhalten/Echtzeitfähigkeit,
- Multi-Master-Fähigkeit.

3.2 Vorüberlegungen zu den Systemkomponenten

Im vorangegangenen Abschnitt wurden die Unterschiede der verschiedenen Systemkomponenten und die Bedeutung von deterministischem und zeitoptimalem Verhalten der Gesamtsysteme angesprochen. Im Hinblick auf ein flexibles, ganzheitliches Steuerungskonzept ist demnach das optimale Zusammenspiel der einzelnen Geräte von entscheidender Bedeutung. Daher werden die derzeit denkbaren Systemkomponenten auf ihre Funktionsweise und ihre Rolle innerhalb der Anwendungen im Weiteren genauer vorgestellt und zwecks Ableitung konkreter Anforderungen detaillierter betrachtet.

3.2.1 Mikroskop

Alle in Kapitel 2 vorgestellten Anwendungen verwenden ein Mikroskop als Basisplattform für jegliche optische Funktionalität. Wie bereits beschrieben wurde, wird hier das Mikroskop iMIC der Firma TILL Photonics verwendet, da es auf den Einsatz in einer vollautomatisierten Anwendung hin optimiert ist und durch seine kompakte Form und flexible Funktionalität in alle beschrieben Systeme integrierbar und in Bezug auf die gewünschten Mikroskopietechniken individuell abstimmbar ist. Vor allem im Hinblick auf ein Machine-Vision System, in dem das iMIC das „Auge“ des Systems darstellt, muss gewährleistet sein, dass seine Funktionalität optimal ausgenutzt wird. In der Grundkonfiguration beinhaltet das iMIC jeweils motorisierte Objektivwechsler für bis zu vier Objektive, zwei Filterwechsler mit jeweils bis zu sechs Filtern, einen xy-Tisch und einen zweistufigen z-Trieb für Grob- und Feineinstellung des Fokus. Die motorisierten Filterwechsler werden gleichzeitig zum Auswählen der Ein- und Ausgangsports verwendet, um ohne Umbau des Mikroskops unterschiedliche Beleuchtungsquellen und/oder Detektionsgeräte benutzen zu können.

Die Motorisierung aller Komponenten erfolgt über Schrittmotoren, deren Steuerungen in den Mikroskopkörper integriert sind. Die bereits erwähnte Feineinstellung des Fokus wird zusätzlich über einen PIEZO-AKTOR realisiert, der mit seinem kapazitativen Sensor eine Positionierung mit einer Genauigkeit von 50 nm erlaubt.

3.2.2 Detektionsgeräte

Wie bereits angesprochen, können mehrere, unterschiedliche Detektionsgeräte an einem Mikroskop betrieben werden. Je nach Anwendung besteht so die Möglichkeit z. B. unterschiedliche Digital-Kameras, leistungsstarke CCD-Kameras oder Photomultiplier einzusetzen. Durch die Verwendung einer standardisierten, mechanischen Schnittstelle für optische Geräte (C-Mount) kann die Auswahl der Kamera nahezu beliebig und herstellerübergreifend erfolgen[1]. Die Schnittstellen zum Betreiben der Geräte sind überwiegend einheitlich gestaltet:

Alle neueren Digital Kameras und insbesondere CCD-Kameras unterstützen Standard Schnittstellen (z. B. Camera Link, **I**nstitute of **E**lectrical and **E**lectronics **E**ngineers (IEEE) 1394 bzw. FireWire, USB 2.0 oder (Gigabit-) Ethernet) für die Konfiguration und das Auslesen der Bilddaten. Zur Synchronisation mit anderen Geräten bieten nahezu alle Kameras Trigger- oder bidirektionale Synchronisierungsschnittstellen an.

Photomultiplier liefern i. d. R. als permanentes Ausgangssignal einen Strom, der proportional zur Intensität des detektierten Lichtes ist. Dieser muss in eine Spannung umgewandelt und dann mit entsprechenden analog-digital Wandlern ausgelesen werden. Hierdurch bleibt die Synchronisation mit anderen Geräten bei dieser Art der Detektion dem Wandelprozess überlassen. Einige Photomultiplier-Messmodule bieten über analoge Steuerleitungen bzw. Standard PC-Schnittstellen zusätzliche Konfigurationsmöglichkeiten, z. B. kann an einigen Geräten so die Beschleunigungsspannung und damit die Verstärkung der Photodetektion konfiguriert werden.

3.2.3 Beleuchtungsquellen

Analog zur Möglichkeit mehrere Detektionsgeräte am iMIC zu betreiben, können ebenfalls verschiedene Lichtquellen eingekoppelt werden, um diese im Betrieb des Mikroskops wahlweise zur Beleuchtung zu verwenden. Je nach Mikroskopieart und Anwendung besteht somit die Möglichkeit Auflichtquellen, Lichtquellen für Fluoreszenzmikroskopie, Blitzgeräte oder unterschiedliche Arten von Lasern in das System zu integrieren. Vor allem die Beleuchtungssysteme für die Fluoreszenz- und Laser-Scanning-Mikroskopie sind überwiegend konfigurierbar, so dass sich z. B. bei Beleuchtungsquellen für die Fluoreszenzmikroskopie verschiedene Wellenlängen und die Beleuchtungsdauer einstellen lassen. Ersteres erfolgt entweder traditionell über eine analoge Steuerspannung zwischen $\pm 10\,\mathrm{V}$ oder über eine Standard PC-Schnittstelle. Der Beginn bzw. die Dauer der Beleuchtung wird i. d. R. über einen digitalen Auslösepuls (Trigger) gesteuert.

Innerhalb der in Kapitel 2 vorgestellten Anwendungen wird die Lichtquelle Polychrome V für fluoreszenzmikroskopische Anwendungen verwendet. Der Polychrome V ist durch seine Eigenschaften (vgl. Abschnitt 1.2) und Synchronisationsmöglichkeiten verglichen mit Konkurrenzprodukten besonders für den Einsatz in zeitkritischen und vollautomatisierten Anwendungen geeignet.

[1] Der Aspekt der Einbindung auf Softwareebene ist hierbei allerdings zu berücksichtigen.

Einfache Laser und Laser älterer Bauart (Ionenlaser, Helium-Neon-Laser, DPSSL, etc. [88]) sind im Allgemeinen nicht konfigurierbar. Bei neueren Diodenlaser ist oftmals die Ausgangsleistung (bei einigen auch die Wellenlänge) über eine analoge Spannung oder eine Standard PC-Schnittstelle einstellbar. Über rein PC-basierte Steuerung besteht bei den sehr aufwändigen und im Vergleich teuren[2], gepulsten Femtosekundenlasern je nach Ausführung die Möglichkeit, die Ausgangsleistung, Wellenlänge und zusätzliche Parameter zu konfigurieren [123, 76]. Um einen Laser, der mit einem Punktstrahl prinzipiell zunächst null-dimensional ist, sinnvoll einsetzen zu können, wird der Laserstrahl z. B. mittels eines Scankopfs in die erste und zweite Dimension abgelenkt. Scanköpfe, Shutter zum Ein- und Ausblenden des Laserstrahls und weiteres Laser-Scanning-Zubehör werden in einem späteren Abschnitt noch detaillierter vorgestellt (s. Abschnitte 3.2.4 und 3.2.6.

Blitzlichtquellen wie z. B. UV-Blitzlichtquellen kommen u. a. in Anwendungen der Photostimulation von lichtempfindlichen Zellen und Organismen oder der chemischen Stimulation durch PHOTOSTIMULATION [82] entsprechender Verbindungen zum Einsatz. Blitzlichtquellen werden überwiegend durch digitale Triggersignale aktiviert.

3.2.4 Erweiterungen für konfokale Mikroskopie

Bei der normalen Lichtmikroskopie mit Weitfeldbeleuchtung besteht ein Bild aus der Überlagerung einer scharfen Abbildung der Punkte in der Fokalebene und einer unscharfen Abbildung der Punkte außerhalb derselben. Zur Verbesserung der Auflösung wurden spezielle Techniken mit teilweiser Beleuchtung entwickelt [34, 23, 123]. So wird in einem Konfokalmikroskop das Anregungslicht in die Probe hineinfokussiert. Licht aus diesem Fokus wird nun i. d. R. durch das gleiche Objektiv auf eine Lochblende abgebildet und gelangt von dort auf einen Detektor (z. B. einen CCD-Chip oder Photomultiplier). Anregungsfokus und Lochblende liegen konfokal, d.h werden aufeinander abgebildet, so dass Licht, das nicht aus der Fokalebene kommt, zweifach unterdrückt wird: Einerseits wird es nicht stark angeregt und andererseits wird Licht von außerhalb der Fokalebene nicht auf die Lochblende fokussiert und damit durch diese vor dem Auftreffen auf den Detektor abgeblockt. Durch ausschließliches Detektieren des Lichtes, das aus einer sehr dünnen Schicht der Probe stammt, ermöglicht die Konfokalmikroskopie somit präzise, optische Schnitte einer Probe.

Streifen-Scan-Verfahren

Beim Streifen-Scan-Verfahren wird auf der Objektebene anstelle der gesamten Ebene lediglich ein Streifen fokussiert belichtet. Nach dem konfokalen Prinzip wird für das Erstellen von Bildern eine Steifenmaske in die Strahlengänge des Anregungs- und des Fluoreszenzlichts eingebracht und entsprechend verschoben [101, 123]. In einem aktuell am BIZ nach einem neuen Ansatz entwickelten streifenkonfokalen Fluoreszenzmikroskop wurde die gewöhnlich mechanische Streifenmaske im Detektionsstrahlengang durch eine virtuelle ersetzt. Die Streifenmaske wird hierbei über ein spezielles Ausleseverfahren von der CCD-Kamera „virtuell“ erzeugt, in dem die jeweils auszublendenden Streifen bei der rechnerischen Rekonstruktion des Bildes nicht berücksichtigt werden. Die Anregung geschieht weiterhin über eine mechanische Streifenmaske. Die Linearbewegung des von der Maske erzeugten Streifenmusters über die Probe

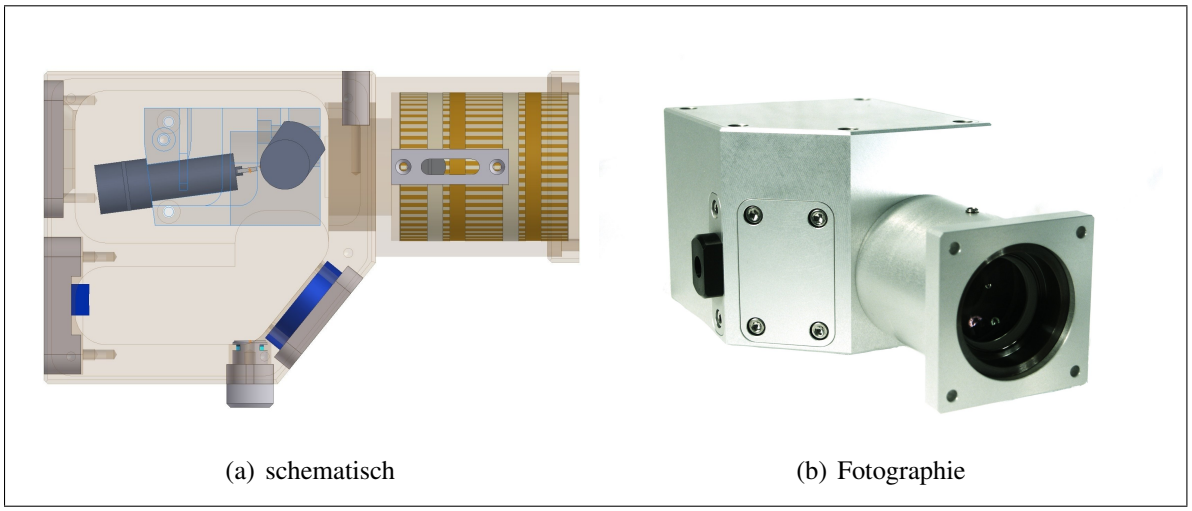
[2] Kosten für Femtosekundenlaser liegen in der Größenordnung >100.000 EUR.

erfolgt mittels eines auf einem galvanometrischen Motor befestigten, beweglichen Spiegels. Streifenkonfokale Fluoreszenzmikroskopie ist durch Synchronisation der Scan-Bewegung des Streifenmusters und der Auslesetechnik der CCD-Kamera realisiert [116].

Durch Wechseln bzw. Weglassen der Streifenmaske im Anregungslicht und durch Anwenden anderer Auslesetechniken sind mit diesem Aufbau zusätzliche Verfahren wie die der strukturierte Beleuchtung oder der Weitfeldmikroskopie möglich [115, 61, 66, 62].

Punkt-Scan-Verfahren

Entgegen dem soeben beschriebenen Verfahren zur teilweisen Beleuchtung verwenden Punkt-Laser-Scan-Techniken keine Punkt- bzw. Streifenmasken zur Strukturierung des zur Beleuchtung verwendeten Lichtes, sondern basieren auf lasergestützter Abrasterung der zu untersuchenden Proben (vgl. Abschnitt 2.2.2). Für die Ablenkung eines Laserstrahls zum Abrastern wird ein so genannter Scankopf benötigt. Unter einem Scankopf versteht man eine optische Apparatur, durch die ein Laserstrahl über eine adäquate, rechtwinklige Anordnung von zwei beweglichen Spiegeln in unterschiedliche Richtungen abgelenkt wird. Durch die Ablenkung kann der punktförmige Laserstrahl beliebig über eine Fläche bewegt werden. In dem als Gemeinschaftsprojekt vom BIZ und der TILL Photonics entwickelten Scankopf „Yanus III“ (s. Abbildung 3.1) sind die Spiegel jeweils auf der Achse von galvanometrischen Motoren befestigt, die in unseren Anwendungen von digitalen Reglern[3] betrieben werden. Im Vergleich zu herkömmlichen analogen Reglern nutzen digitale Regler die Motorleistung bis an die physikalischen Grenzen aus und sind damit bei mindestens gleicher Genauigkeit bis zu vier Mal schneller. Die Ansteuerung der Regler kann entweder traditionell über analoge Spannungen im Bereich $\pm 10\,\mathrm{V}$ oder digital mit jeweils einer Auflösung von 16Bit bei 100 kHz Taktung erfolgen. Die digitalen Regler operieren prinzipiell als autarke Geräte und werden im Verbund eines Scankopfs synchron betrieben.



(a) schematisch (b) Fotographie

Abbildung 3.1: Scankopf Yanus III für zwei-Achsen Ablenkung eines Laserstrahls bei stationärem Austrittspunkt

Um einen Scankopf von außen als eine Einheit betrachten zu können, wurde im BioImaging Zentrum eine Digitale Scanner Kontrollsteuerung (DSC) entwickelt, die entweder über eine

[3] Verwendet wird das Model DC900 der Firma SmartMove GmbH, Martinsried, Deutschland.

Standard PC-Schnittstelle (derzeit EIA-232, zukünftig wahlweise auch USB) oder über ein **S**erial **P**eripheral **I**nterface (SPI) [118] konfiguriert und mittels Triggerleitungen synchronisiert werden kann. Über die Konfigurationsmöglichkeit der DSC kann ein Laserstrahl so z. B. - über ein Synchronisationssignal gestartet - mit vorbestimmter Geschwindigkeit eine spezifizierte geometrische oder arbiträre Fläche beschreiben bzw. "abfahren". Diese Technik wird bei der Laser-Scanning-Mikroskopie zum Abrastern der Proben und damit zum Erstellen hochauflösender, kontrastreicher, zwei und dreidimensionaler Bilder verwendet [104]. Wird der Laserstrahl nicht auf eine Fläche, sondern auf einer Kurve bewegt, können bei Verwendung eines entsprechenden Lasers, Schnitte in die untersuchte Probe eingebracht werden (Mikrodissektion [73, 45, 141, 85]). Mittels entsprechend konfigurierbaren Lasern und der Scan Kontrollsteuerung können neben dem Abrastern von Proben [34, 139] und Mikrodissektionen auch andere Laser-basierte Anwendungen aufgesetzt werden, wie das Bleichen von Farbstoffen, dem so genannten „Photobleaching" [22] oder die Anwendung einer optischen Pinzette [138, 16, 142, 58].

Weiteres Zubehör zur Laser-Scanning-Mikroskopie sind Laser Abschwächer, mit denen die Intensität von Laserstrahlen verringert werden kann, und so genannte „Laser-Combiner" (**A**cousto-**O**ptical **T**unable **F**ilter (AOTF)), mittels derer verschiedene/verschiedenfarbige Laser in einen Strahlengang (z. B. in eine Glasfaserleitung) eingekoppelt werden können.

Nipkow-Konfokal-Mikroskopie

Durch Rotation einer so genannten Nipkow-Scheibe, die in den Detektionsstrahlengang eines Mikroskops mit Weitfeldbeleuchtung eingebracht wird, können ebenfalls konfokale Effekte erzeugt werden [42, 113]. Abbildung 3.3(a) zeigt den schematischen Aufbau eines iMIC-basierten Nipkow Konfokalmikroskops. Über Spiegel im Innern des iMIC wird der Strahlengang nach außen umgelenkt, so dass die Nipkow-Scheibe in diesen eingebracht werden kann (Abbildung 3.3(b)).

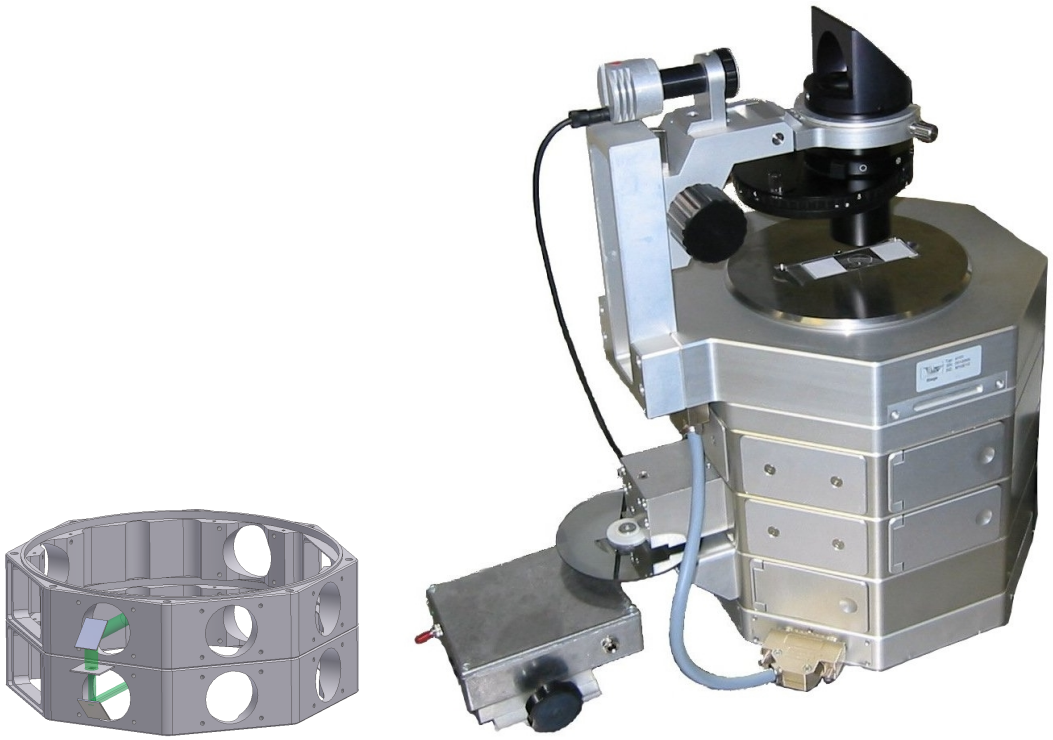
3.2.5 Sensorik

Die im Zuge der Beschreibung des IMR für Langzeituntersuchungen von lebendem Probenmaterial (s. Abschnitt 2.2.4) vorgestellten Sensoren werden über eine Sensorelektronik betrieben. Diese Elektronikeinheit liefert mit einer EIA-232-Schnittstelle eine Standard PC-Schnittstelle, über welche die Möglichkeit besteht verschiedene Messparameter einzustellen, Messungen zu initiieren bzw. die ermittelten Messdaten auszulesen.

3.2.6 Probenmanipulation

Unter der Rubrik der Probenmanipulation seien hier alle Geräte zusammengefasst, die direkt oder indirekt manipulierenden Einfluss auf die zu untersuchenden Proben haben. Exemplarisch werden im Folgenden einige „Manipulatoren" vorgestellt:

- *Bewegung von Proben:* Roboter, die z. B. in einem High Throughput System die automatische Entnahme der Proben aus einem INKUBATOR, Aufsetzen auf die Analyseplattform und Entsorgen der Proben nach abgeschlossener Analyse übernehmen [114].
- *„Liquid Handling":* Stationäre oder bewegliche, strömungstechnische Geräte (Roboter), die ein automatisches, kontrolliertes Hinzugeben von Nähr- oder Wirksubstanzen zu den



(a) schematischer Auszug inkl. Strahlengang

(b) Photographie

Abbildung 3.2: iMIC-basiertes Nipkow Konfokalmikroskop: In den aus dem Mikroskop geführten Strahlengang wird eine rotierende Nipkow-Scheibe eingeführt.

untersuchten Proben (Zellkulturen) ermöglichen. Als Beispiel sei hier der in Zusammenarbeit des LME der TUM und der Firma H&P Labortechnik, entwickelte Pipettierroboter genannt (s. Abbildung 3.3), der innerhalb des in Abschnitt 2.2.4 vorgestellten **I**ntelligent **M**icroplate **R**eader eingesetzt wird. Der Pipettierroboter versorgt die Proben nicht nur mit Nährmedien, sondern kann nach Wunsch auch zu testende Wirksubstanzen beifügen. Über eine EIA-232-Schnittstelle können alle Funktionen des Roboters (u. a. Anfahren aller Positionen, Ein- und Auspipettieren bestimmter Volumina, Aufnahme und Abwurf von Pipettenspitzen) ausgeführt werden.

- *Klimastatisierung:* Geräte zur Regelung der klimatischen Bedingungen im Umfeld der zu untersuchenden Proben. Hierzu gehört u. a. die Regelung von Temperatur, pH-Wert und O_2-Konzentration in der unmittelbaren Umgebung der Proben. Während stabile Basisparameter für einige Zellen bzw. Zellkulturen überlebenswichtig sind, stellen sie generell die Grundlage für reproduzierbare in-vitro Experimente dar.
- *Laserbasierte Manipulation:* Der Vollständigkeit halber sei erwähnt, dass die in Abschnitt 3.2.4 beschriebenen Verfahren der Mikrodissektion, des Photobleaching und der optischen Pipette selbstverständlich auch in den Bereich der Probenmanipulation gehören. Die dazu notwendigen Geräte, deren Schnittstellen und groben Funktionsweise wurden bereits in Abschnitt 3.2.4 vorgestellt.

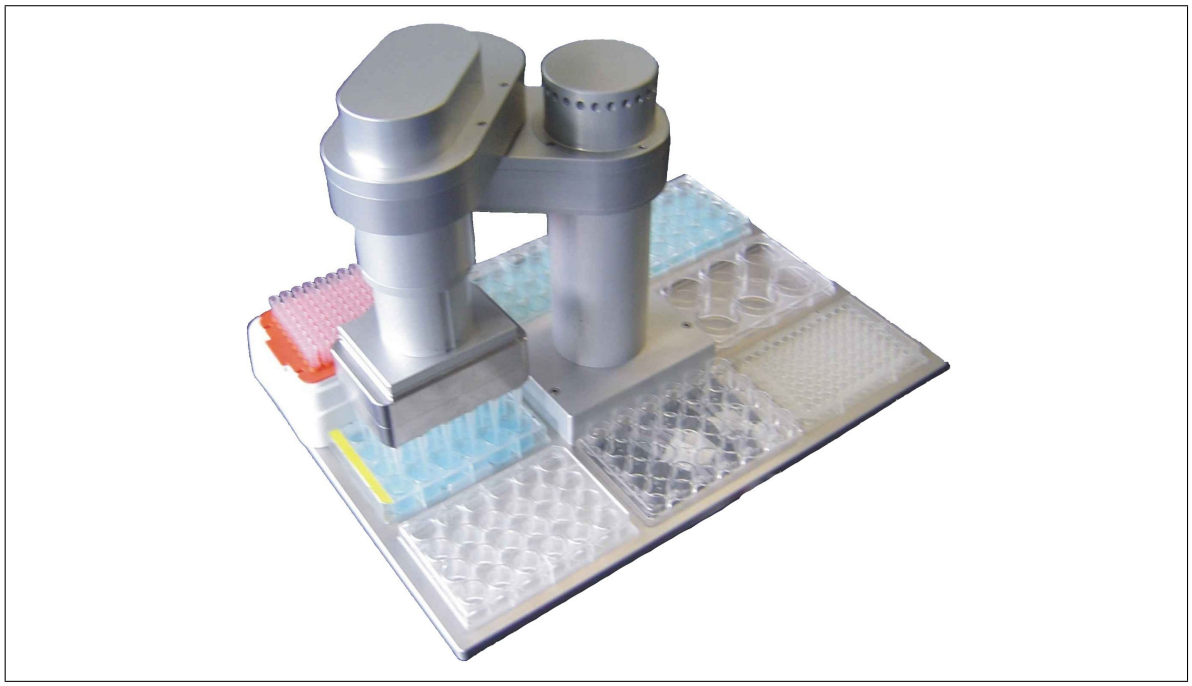


Abbildung 3.3: Pipettierroboter; Gemeinschaftsprojekt des Steinbeiszentrum Lab-on-Chip-Systeme, des LME und Thermo Electron (Oberschleissheim) GmbH (ehemals H&P Labortechnik AG)

3.2.7 Peripheriegeräte

Analysesysteme für die Life Sciences beinhalten neben Komponenten zum Messen und Beobachten i. d. R. eine Vielzahl von mehr oder weniger aufwändigen Peripheriegeräten. In Aufbauten mit geringem Durchsatz werden für die Versorgung von Proben mit Nährlösungen und Wirkstoffen z. B. einfache Zufluss- und Absaugvorrichtungen mittels Quetschventilen[4], Motorventile[5], Dosierpumpen und/oder Schlauchquetschpumpen[6] verwendet. Innerhalb von Hochdurchsatzsystemen kommen bei der Verwendung von Mikrotiterplatten zudem Geräte zur Beschleunigung und Automatisierung der Prozesse zum Einsatz. Solche Komponenten sind z. B. „Sealer Workstations“, BarCode Reader oder Beschriftungsautomaten für automatische Zuordnung der Messdaten zu den entsprechenden Versuchsreihen. Der überwiegende Teil dieser Peripheriegeräte kann, über eine EIA-232-Schnittstelle mit einem PC verbunden, softwareseitig betrieben und/oder für automatischen Betrieb programmiert werden .

3.2.8 Anwender-PC

In allen vorgestellten Anwendungsbeispielen wird ein PC als Hauptschnittstelle zwischen dem Benutzer und den Mess- bzw. Analysesystemen verwendet. Einerseits verfügen viele der Systemkomponenten anstelle von direkten Bedienelementen (z. B. Druckschalter und LCD-Anzeige) über eine Standardschnittstelle zu einem PC, um über diesen konfiguriert und/oder betrieben zu werden. Andererseits wird in allen Systemen, unabhängig von der Anzahl und Beschaffenheit der einzelnen Komponenten, generell *eine* standardisierte Schnittstelle zum Be-

[4]Quetschventile sind z. B. von der Firma Neptune Research Inc, West Caldwell, NJ, USA, erhältlich.

[5]Motorventile sind z. B. von der Firma Besta, Heidelberg, Deutschland erhältlich.

[6]Dosierpumpen und/oder Schlauchquetschpumpen sind z. B. von der Firma IsmaTec Laboratoriumstechnik GmbH, Wertheim-Mondfeld, Deutschland, erhältlich.

nutzer benötigt. Der PC stellt mit den bekannten Eingabe- und Ausgabegeräten (Tastatur, Maus, Monitor, etc.) eine relativ preiswerte und in Forschung und Industrie allgemein anerkannte Benutzerschnittstelle dar. Er nimmt als Ein- und Ausgabemedium und der Möglichkeit zur Verarbeitung, Analyse und Archivierung von Messdaten daher in einem Steuerungskonzept eine Sonderrolle ein. In einem ersten Ansatz würde der PC daher als zentraler Knotenpunkt alle Systemkomponenten sternförmig miteinander verbinden, wie es in Abbildung 3.4 exemplarisch dargestellt ist.

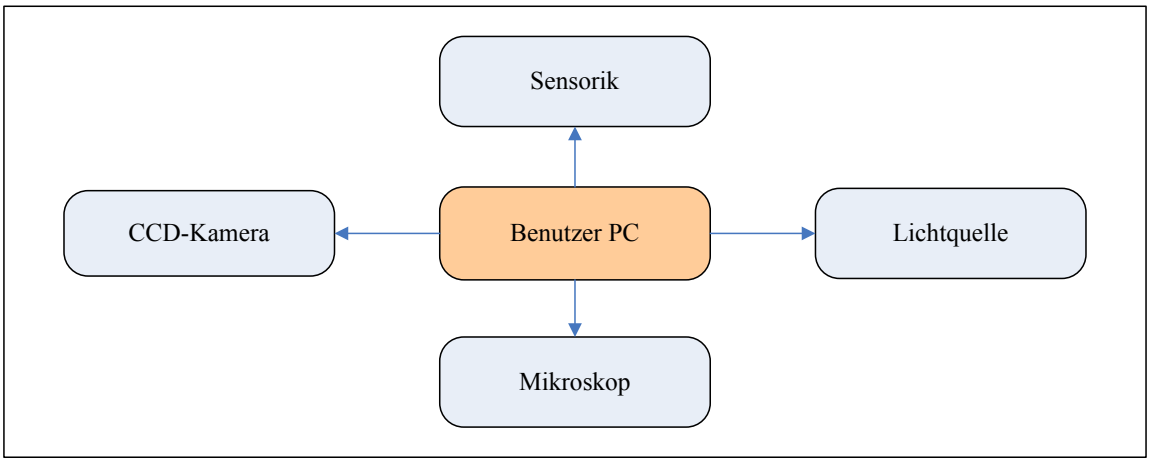


Abbildung 3.4: Exemplarische Darstellung eines ersten Ansatzes für ein Steuerungskonzept: Der Anwender-PC verbindet alle Systemkomponenten sternförmig miteinander und übernimmt die Ablaufkontrolle.

Für die Umsetzung dieses Ansatzes müsste ein Standard-PC i. d. R. allerdings um Schnittstellenkarten erweitert werden, welche die Verbindung aller Komponenten mit dem PC ermöglichen. Benötigte Schnittstellen sind u. a.: zusätzliche serielle Anschlüsse, da moderne PCs i. d. R. maximal über eine (bzw. Laptops oft gar keine) serielle Schnittstelle verfügen, analoge Ausgänge zum Steuern von u. a. Piezo-Motoren oder Laser-Combinern, analoge Eingänge u. a. zum Auslesen von Sensoren, etc.

Bei der Steuerung der Systemkomponenten über den PC ist allerdings zu bedenken, dass moderne Betriebssysteme kein deterministisches Zeitverhalten haben und damit keine Echtzeitkontrolle der angeschlossenen Geräte garantieren können. Optimale Ausnutzung von leistungsstarken Systemkomponenten, wie es in einer high-end Anwendung wie z. B. einem Machine-Vision System notwendig ist, kann daher mit einem rein PC-basiertem Steuerungskonzept nicht realisiert werden. Neben Betriebssystemen wie Windows XP® existieren auch so genannte Echtzeitbetriebssysteme, die auf Standard PCs betrieben werden können . Allerdings bieten diese Betriebssysteme maximale Zeitauflösungen im Bereich von Millisekunden, während einige Komponenten der erwähnten Analysesysteme bereits jetzt auf einem Zeitraster von 10 µs arbeiten. Für das angestrebte Machine-Vision System kommt eine Steuerung der Systemkomponenten über einen PC aus Gründen des Zeitverhaltens daher nicht in Frage.

3.3 Aus den Vorüberlegungen abgeleitete Anforderungen

Bei der näheren Betrachtung der möglichen Systemkomponenten, den PC mit seiner Sonderrolle ausgenommen, fallen wiederkehrende Aufbau- und Funktionsmuster auf. Die zu integrie-

renden Geräte können daher anhand ihrer Betriebsart generell in zwei Kategorien unterteilt werden:

- *„Intelligente Geräte“*: Hierbei handelt es sich um Geräte, die neben einer oder mehrerer konfigurierbarer Betriebsarten die Möglichkeit zur Synchronisation auf Hardware-Ebene, z. B. über eine Trigger-Leitung, zur Verfügung stellen. Beispiele hierfür sind u. a. CCD-Kameras oder die Lichtquelle Polychrome V.
- *„Nicht intelligente Geräte“*: Geräte, die keine eigenen, zusammenhängenden Funktionen selbständig ausführen können und/oder keine Möglichkeit der Synchronisation auf Hardwareebene bieten, werden im Folgenden als „nicht intelligente Geräte“ bezeichnet. Diese Geräte zeichnen sich dadurch aus, dass sinnvolle bzw. zusammenhängende Operationen lediglich über komplexe und damit zeitaufwändige Verfahren bzw. Kommunikation mit dem übergeordneten Steuerungsgerät ausgeführt werden können. Beispiele hierfür sind u. a. Schrittmotoren, deren Steuerungen über eine serielle Verbindung betrieben werden, die aber außer einer zunächst bedeutungslosen Motorbewegung keine in sich abgeschlossene Gerätefunktion ausführen können.

Die Einordnung von zu betreibenden Geräten in eine der vorgestellten Kategorien ist je nach Funktionalität nicht immer eindeutig. So kann ein Gerät (z. B. ein Roboter), das über eine serielle Schnittstelle bedienbar ist, durchaus als intelligentes Gerät bezeichnet werden, da man z. B. mit einem übertragenen Befehl mit definierter Länge und damit definierter Übertragungsdauer eine Kette von vordefinierten, komplexen Bewegungen anstoßen kann. In folgender Liste werden die vorgestellten Systemkomponenten den verschiedenen Kategorien zugewiesen:

- „Intelligente Geräte“
  - CCD-Kamera: Konfiguration von Belichtungsdauer, verschiedener Aktivierungs- und Auslesemodi[7], Übertragen der Bilddaten zum Anwender-PC; mögliche Schnittstellen zum PC: USB, IEEE 1394 bzw. FireWire, Camera Link, u. a.; mögliche Synchronisationsschnittstellen: digitale I/O, bidirektionale Synchronisationsleitung;
  - Polychrome V: Konfiguration von Beleuchtungsdauer, verschiedener Aktivierungsmodi: einfach-Trigger mit definierter Beleuchtungsdauer, zweifach-Trigger zum Bestimmen von Start und Ende einer Beleuchtung und Gate-Modus (Start der Beleuchtung bei steigender Flanke, Ende der Beleuchtung bei fallender Flanke); mögliche Schnittstellen zum PC: EIA-232 oder USB; mögliche Synchronisationsschnittstellen: digitaler Ein- oder Ausgang;
  - Scan Kontrollsteuerung: Steuerung von bis zu vier digitalen Reglern für galvanometrische Motoren. Programmierung von Form und Größe der Flächen bzw. Kurven, und der Geschwindigkeit, in der z. B. ein Laserstrahl von einem Scankopf auf der Probe „verfahren“ werden soll; Schnittstelle zum PC: EIA-232 bzw. SPI;
  - Sensorkontrolleinheit: Betreiben und Auslesen der Sensoren; Schnittstelle zum PC: EIA-232 (mittelfristig auch USB);

[7] Auslesemodi wie z. B. Belichtung mit anschließendem Auslesen, dass so genannte überlappende Auslesen, oder TDI.

    - Pipettierroboter: Anfahren von vorher definierten Positionen (z. B. Pipetten-Reservoir, Nährlösungs-, Wirkstoff- und Abfallbehälter), Ein- und Auspipettieren von beliebigen Volumina (bis auf wenige µl; Schnittstelle zum PC: EIA-232
    - Platzierungsroboter: z. B. Caliper Twister oder Qiagen Biorobot 3000, 8000, Twister I und Twister II; Schnittstelle zum PC: überwiegend EIA-232;
- „nicht intelligente Geräte“
    - iMIC: Steuerung für Schrittmotoren der x-y-z-Achsen, Filterwechsler und Objektivwechsler über EIA-232-Schnittstellen; Steuerung des Piezo-Aktors für die Feineinstellung des Fokus über eine analoge Steuerspannung;
    - Scankopf: Digitale Regler für galvanometrische Motoren - serielle Schnittstelle über die alle 10 µs ein Positionswert (16 Bit) an den Regler übertragen wird.
    - Sensoren: Ausgangssignale in Form von z. B. analogen Spannungen, Strömen, pulsweitenmodulierten Signalen, frequenzmodulierten Signalen, etc.
- Ausnahmen:
    - Sonderrolle der Benutzerschnittstelle: PC übernimmt zusätzlich die Ablaufvorbereitung, Datenverarbeitung und Archivierung
    - Klimakontrolleinheit: Ist funktionsbedingt nicht echtzeitfähig, da z. B. Temperaturänderungen verhältnismäßig langsam ablaufen.

Die aus den Vorüberlegungen zu den Systemkomponenten zusammengetragenen Anforderungen lassen sich folgendermaßen zusammenfassen:

- Einteilung der potenziellen Systemkomponenten in drei Kategorien:
  „intelligente Geräte“
  „nicht intelligente Geräte“
  Ausnahmen
- Ausnutzung von geräteeigenen Funktionen und Möglichkeiten der Vorabkonfiguration;
- Ausnutzung von hardwareseitigen Synchronisationsmöglichkeiten
- Unterstützung von rein softwarekontrollierten Geräten

3.4 Konzept der verteilten Intelligenzen

Ausgehend von den generellen und den aus den Vorüberlegungen zu den Systemkomponenten abgeleiteten Anforderung wurde ein Steuerungskonzept erarbeitet, das die Funktionalität aller beteiligten Systemkomponenten bestmöglich im Kontext der Gesamtsysteme einbringt. Der Funktionsumfang und die Leistungsfähigkeit der „intelligenten Geräte“ sollen dabei über Hardwaresynchronisation optimal ausgenutzt werden. Gleichzeitig wird die Einbindung von „nicht intelligenten Geräten“ über PC-basierte Steuerung ermöglicht.

Abbildung 3.5 illustriert in einem Flussdiagramm die groben Abläufe des Betriebs eines modularen Mess- bzw. Analysesystems nach dem erarbeiteten Steuerungskonzept: Der Benutzer beschreibt die Durchführung einer Messung bzw. Analyse durch Eingabe eines Ablaufprotokolls. Daraufhin werden die „intelligenten“ Systemkomponenten über den PC vorkonfiguriert und damit auf die individuelle Rolle innerhalb des Gesamtsystems vorbereitet. Nach Starten

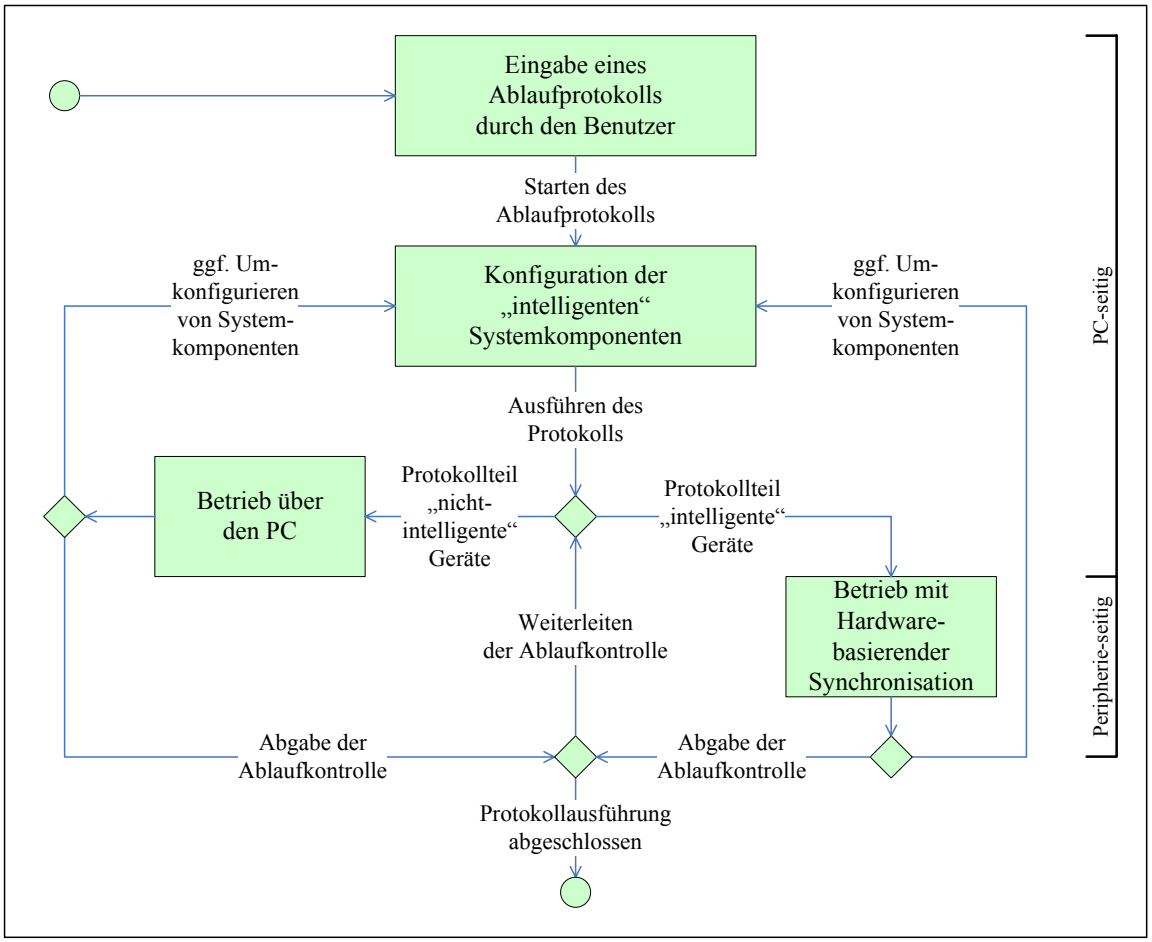


Abbildung 3.5: Grober Ablauf des Betriebs eines automatisierten Mess- bzw. Analysesystems nach dem erarbeiteten Konzept der „verteilten Intelligenzen“

der Protokollausführung wechselt die Ablaufkontrolle entsprechend den im Protokoll verwendeten Geräten zwischen hardwaresynchronisiertem und PC-basiertem Betrieb. Ein Eingreifen in die bzw. Adaptieren der Ablaufkontrolle ist durch Umkonfigurieren der Systemkomponenten möglich.

Die Hardwaresteuerung eines Mess- bzw. Analysesystems soll allerdings nicht die bereits existierende Funktionalität der Systemkomponenten in einer gigantischen Steuereinheit kopieren. Nach einer etwaigen Vorkonfiguration durch den PC sollen vielmehr die vorhandenen, auf die einzelnen Baugruppen „verteilten Intelligenzen“ über Triggersignale synchronisiert werden. Für die hardwarebasierte Ablaufkontrolle sollen die „intelligenten“ Geräte daher über ein zentrales „Steuerungs-Rückgrat“ verbunden werden, welches die durch die vorkonfigurierten Systemkomponenten ausgeführten individuellen Aktionen (Teile des Ablaufprotokolls) zeitgenau koordiniert.

Durch Übertragung der groben Abläufe des Betriebs auf die Hardwareebene eines Gesamtsystems, ergeben sich die erforderlichen, logischen Verbindungen zwischen den Systemkomponenten. Abbildung 3.6 stellt ein fiktives Gesamtsystem dar, in dem Geräte aller Kategorien, mit den denkbaren Schnittstellen und mit verschiedensten Anforderungen an die Zeitgenauigkeit eingebracht wurden. Anhand dieses Aufbaus sollen nun im Weiteren die möglichen Verbindungsszenarien erarbeitet werden.

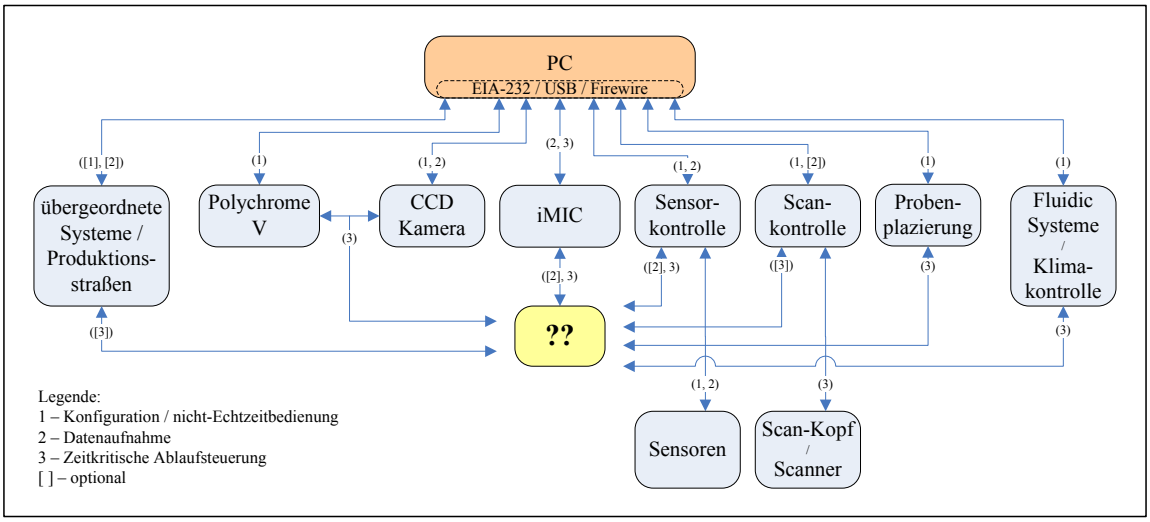


Abbildung 3.6: Logische Verbindungen zwischen den einzelnen Systemkomponenten

Geräte, die als Konfigurations- und Datenschnittstelle USB, IEEE 1394 bzw. Firewire, Camera Link oder ähnliche nicht echtzeitfähige Schnittstellen benutzen, werden generell mit dem Anwender-PC verbunden, da dieser i. d. R. bereits über die entsprechenden Schnittstellen verfügt bzw. wie im Falle von Camera Link für den Anschluss über kommerziell erhältliche PCI-Einsteckkarten aufgerüstet werden kann. Zudem existiert für die entsprechenden Geräte i. d. R. PC-Software (komplette Anwendersoftware oder ein **S**oftware **D**evelopment **K**it (SDK) mit low-level Software-Schnittstelle, **G**raphical **U**ser **I**nterface (GUI)-Elementen, etc.), mittels derer die Geräte entweder komfortabel konfiguriert, betrieben oder in eine Gesamtsoftware eingebunden werden können. Die von einem Detektionsgerät anfallenden Datenmengen[8] werden außerdem über diese Schnittstellen an den PC für die Anzeige, Weiterverarbeitung oder Archivierung übertragen.

Verfügen Komponenten über keine Standard PC-Schnittstelle, müssen sie nach dem erarbeiteten Konzept über eine Kontrolleinheit betrieben werden. Diese muss einerseits die Intelligenz besitzen, um die entsprechenden Geräte zu bedienen (z. B. Messelektronik zum Betreiben und Auslesen von Sensoren) und andererseits Schnittstellen liefern, welche die Verbindung zum PC zwecks Konfiguration und/oder Datenaustausch und die Verbindung zum „Kontroll-Rückgrat“ zur Hardwaresynchronisation ermöglicht. In Abbildung 3.6 ist diese Geräteklasse sowohl durch die Kombination von Sensoren und Sensorkontrollsteuerung als auch durch Scankopf und Scan Kontrollsteuerung vertreten.

Das bereits erwähnte „Steuerungsrückgrat“ soll für optimales Zeitverhalten der Gesamtsysteme die Hardware-Synchronisation durch Anstoßen der vorkonfigurierten Abläufe zu im Protokoll definierten Zeiten gewährleisten. Dem in Abbildung 3.6 dargestellten Knotenpunkt (gelber Block mit Beschriftung: „??“) kommt daher die zentrale Rolle der Hardware-basierenden Synchronisation zu. Aufgrund der unterschiedlichen elektrischen Eigenschaften (digital/analog; Eingang/Ausgang, etc.) der im Knotenpunkt zusammenlaufenden Steuerungssignale ist ein schlichtes physikalisches Verbinden nicht möglich. Zudem kann nicht davon ausgegangen werden, dass jeweils alle Geräte gleichzeitig Aktionen ausführen, sondern dass vielmehr indivi-

[8] Anfallende Datenmengen z. B. einer CCD-Kamera liegen derzeit in der Größenordnung bis über 80 MB/**S**ekunde (s). Typische Datenmenge in den TILL-Systemen sind derzeit 10-20 MB/ s.

duelles oder durch unterschiedliche Vorlaufzeiten bedingtes zeitlich versetztes Initiieren von Abläufen erforderlich ist. Daher muss die Kontrolle jeder einzelnen Steuerungsleitung individuell anpassbar und zeitlich flexibel konfigurierbar sein. Um dies zu realisieren, sieht das Steuerungs-Konzept eine extra Hardware vor, die **I**ntegration **C**ontrol **U**nit (ICU). Auf Abbildung 3.6 basierend zeigt Abbildung 3.7 das um die ICU erweiterte Blockschaltbild eines fiktiven Gesamtsystems mit den logischen Verbindungen aller Systemkomponenten.

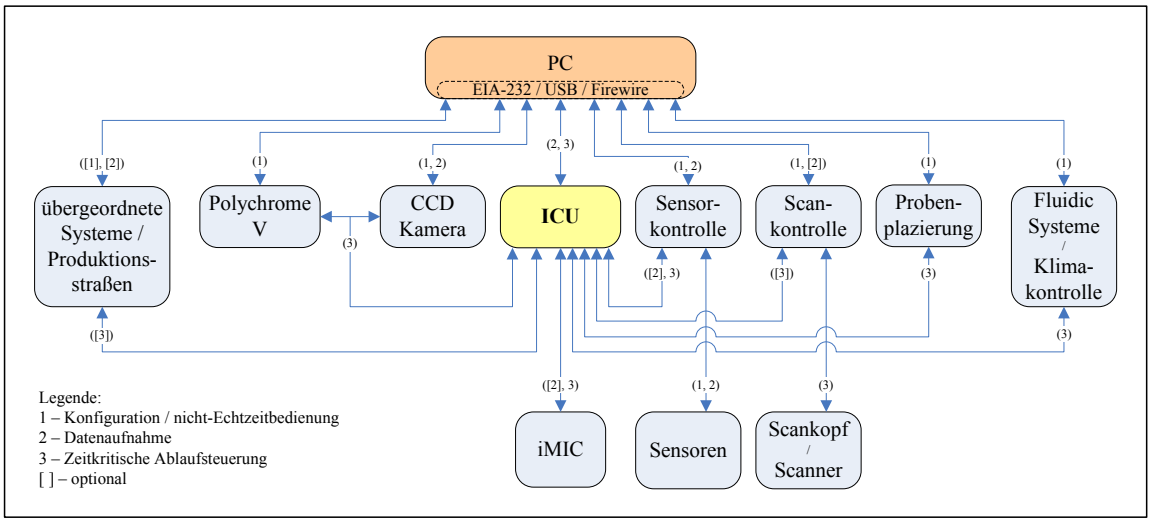


Abbildung 3.7: Blockschaltbild eines fiktiven Gesamtsystems nach dem erarbeiteten Steuerungskonzept mit den logischen Verknüpfungen aller Systemkomponenten

Die ICU reiht sich nach Abbildung 3.7 innerhalb des Steuerungskonzepts in die Klasse der intelligenten Geräte ein - mit einer Standardschnittstelle zum Anwender-PC einerseits und mit diversen Schnittstellen zur Hardware-Synchronisation andererseits. Die Funktionalität („Intelligenz“) der ICU muss nach den erörterten Anforderungen die individuelle Kontrolle jeder einzelnen Synchronisationsschnittstelle zu im Protokoll definierten Zeitpunkten ermöglichen. Dabei ist die Kontrolle einer bestimmten Schnittstelle prinzipiell unabhängig von der Kontrolle der anderern Schnittstellen und Systemkomponenten, es sei denn eine Abhängigkeit ist durch das Protokoll erwünscht (z. B. Ausgabe eines Triggers zum Starten der Beleuchtung, nach dem Erfassen eines „Kamera-Bereit“ Signals). Mittels der Protokolle soll somit ermöglicht werden, dass einzelne Aktionen oder ganze Ablaufsequenzen entweder zu vorgegebenen Zeiten oder abhängig von externen Ereignissen (z. B. Trigger-Eingang) angestoßen werden.

Nach dem Steuerungskonzept wird die hardwarebasierte Synchronisation über einfache elektrische Schnittstellen (z. B. Trigger- oder Analogsignale) oder zeitlich deterministische Datenschnittstellen realisiert. Wie in Abschnitt 3.3 angedeutet wurde, können serielle Punkt-zu-Punkt Verbindungen wie z. B. SPI oder EIA-232-Schnittstellen diese Anforderung durch definierte Größe des ausgetauschten Datenvolumens und damit definierter Übertragungsdauer erfüllen. Voraussetzung ist allerdings, dass keine softwareseitigen Latenzen der Kommunikation auftreten, wie es z. B. bei einem PC mit Windowsbetriebssystem der Fall ist. Innerhalb der beschriebenen Anwendungsbeispiele ist es wünschenswert, dass Systemkomponenten, die über eine EIA-232-Schnittstelle betrieben werden müssen (z. B. die im iMIC verwendeten Schrittmotorsteuerungen), mit in die Hardware-synchronisierten Abläufe einbezogen werden können. Daher ist vorgesehen, dass die ICU zusätzlich zu den „einfachen“ analogen und digitalen Steuerleitungen EIA-232-Schnittstellen zur Verfügung stellt, die in Echtzeit betrieben werden können.

Die in den Anforderungen verlangte Modularität, Erweiterbarkeit und Möglichkeit zur Einbindung in übergeordnete Systeme ist beim Erarbeiten des Steuerungskonzepts prinzipiell berücksichtigt worden: Die ICU als Knotenpunkt und verantwortlich für hardwarebasierte Synchronisation soll den modularen Charakter der Gesamtsysteme sowohl in Bezug auf die ihre Hardware als auch auf ihre Firmware übernehmen. Die benötigten Schnittstellen (Hardware und Firmware) sollen daher modular gestaltet, sowie austausch- und erweiterbar sein. Eine Einbindung in übergeordnete Systeme, wie z. B. eine Produktionsstraße oder ein Screening Gerät mit automatischem Probe-Handling, ist durch die vorgesehene Multi-Master-Fähigkeit und die Möglichkeit zur Synchronisation auf externe Steuerungssignale ebenfalls möglich. Die Umsetzung des Konzepts auf Hardware- und Firmware-Ebene wird in den folgenden beiden Kapiteln detailliert beschrieben.

4 Hardware-Plattform für Systemintegration und Echtzeitsteuerung

In Kapitel 3 wurde herausgearbeitet, dass für den Echtzeitbetrieb eines modularen Systems, in dem mehrere an sich autarke Geräte miteinander komplexe und zeitkritische Aufgaben bewältigen, eine spezielle Hardware nötig ist. Für optimale Ausnutzung der Eigenschaften und Funktionalitäten aller Systemkomponenten sollen diese über eine Kontrollhardware physikalisch miteinander verbunden werden. Die Synchronisation soll aber dennoch individuell und flexibel konfigurierbar erfolgen. Im Hinblick auf die in Kapitel 2 vorgestellten Anwendungen wurde daher eine Kontroll-Hardware konzipiert, welche die Anforderungen der Systemintegration und der flexiblen Echtzeitsteuerung erfüllt: **I**ntegration **C**ontrol **U**nit (ICU). Innerhalb der hier vorgestellten Dissertation stellt die Konzeption und die Implementierung dieser Hardware einen großen Teil der Arbeiten dar und wird daher in den folgenden Abschnitten eingehend behandelt.

4.1 Anforderungen an die Hardware

Generelle Anforderungen ergeben sich automatisch aus der Betrachtung der möglichen Anwendungsgebiete und dem daraus abgeleiteten Steuerungskonzept: Die ICU muss in der Lage sein, alle Systemkomponenten mit hardwarebasierenden Synchronisationsmöglichkeiten physikalisch zu verbinden und dabei die in den Abschnitten 3.2 und 3.3 vorgestellten Merkmale der unterschiedlichen Gerätekategorien berücksichtigen. Weiterhin muss die Hardware der ICU den modularen Charakter der Systeme übernehmen und entsprechend der unterschiedlichen Gerätekonstellationen an deren Synchronisationsmöglichkeiten anpassbar sein. Vor allem für den Einsatz innerhalb der Grundlagenforschung ist eine flexible Erweiterungsmöglichkeit von großer Bedeutung, da sich die Anforderungsprofile von Mess- und Analysesystemen ständig verändern bzw. erweitern (man spricht hier vom sog. Bastler-Anwender). Um ein deterministisches Zeitverhalten der Gesamtsysteme zu ermöglichen, muss die Hardware die Echtzeitfähigkeit aller Schnittstellen zu den Systemkomponenten gewährleisten. Das anvisierte Zeitraster sollte 10 µs betragen. Dies geschah in Anlehnung an die derzeit schnellste Systemkomponente, dem digitalen Scan-Regler. Eine Synchronisationsschnittstelle zu einer der Systemkomponenten muss demnach mit einer maximalen Verzögerung von 10 µs bedient werden. Die Verbindung zum Anwender-PC sollte über eine standardmäßig vorhandene Schnittstelle wie eine EIA-232-Schnittstelle oder eine USB-Schnittstelle erfolgen.

Zusätzliche, generelle Anforderungen zielen auf ein kompaktes und funktionelles Design im Stil von Laborgeräten, das eine sinnvolle Einbindung der Kontrollhardware in bestehende Mess- und Analyseaufbauten ermöglichen (z. B. bezüglich Erreichbarkeit von Kontaktsteckern, Schaltern, usw.). Weiterhin sollen die Herstellungskosten der ICU verglichen mit den Gesamtkosten der verschiedenen Systeme in einem adäquaten Rahmen bleiben.

Obwohl die ICU durch ihre Modularität und Flexibilität generell vielseitig in Anwendungen einsetzbar sein soll, die hardwarebasierte Synchronisation in Echtzeit benötigen, ist sie primär für den Einsatz in den in Kapitel 2 vorgestellten Anwendungen bzw. dem angestrebten Mikroskop-basierten Machine-Vision System konzipiert. Einige Geräte zählen daher zur Grundausstattung und können als Komponenten aller Systeme angesehen werden: Neben der ICU wird die

Mikroskop-Plattform iMIC, eine Beleuchtungsquelle (standardmäßig Polychrome V für Anwendungen der Fluoreszenzmikroskopie) und ein Detektionsgerät (i. d. R. CCD-Kamera) als minimale Ausstattung anzunehmen sein. Eine Vorgabe war daher, die benötigten Schnittstellen für diese Geräte in Kontaktsteckern zusammengefasst mit in die ICU zu integrieren, so dass möglichst wenige Verbindungskabel zu den Geräten notwendig sind. Zudem sollten die zur Versorgung des Systems notwendigen Netzteile weitestgehend in die ICU integriert werden. Da der Polychrome V bereits über ein integriertes Netzteil verfügt und die CCD Kameras unterschiedliche Netzteile mit stellenweise integrierter Kameralüftung benötigen, wurde die Vorgabe auf die Netzteile für das iMIC und die ICU selbst beschränkt. Spezielle Anforderungen an geringe Leistungsaufnahme der ICU werden nicht gestellt, da sich keine der in Frage kommenden Anwendungen für den mobilen Einsatz eignet und daher grundsätzlich von Netzbetrieb ausgegangen werden kann. Die folgende Liste fasst die Vorgaben und die Anforderungen an die Hardware der ICU zusammen:

- Standardisierte Schnittstelle zum Anwender-PC (EIA-232- oder USB-Schnittstelle),
- physikalische Verbindung aller Systemkomponenten zur Hardwaresynchronisation,
- modularer Aufbau der Synchronisationsschnittstellen zwecks:
  - flexibler Erweiterbarkeit
  - Anpassung an unterschiedliche Gerätekonstellationen,
- Echtzeitfähigkeit aller Schnittstellen zu den Systemkomponenten (wünschenswertes Zeitraster 10 µs),
- Integration spezieller Schnittstellen für das iMIC und den Polychrome V,
- Integration der Netzteile für die Versorgung der ICU und des iMIC,
- geringe Herstellungskosten,
- kompaktes und funktionelles Design im Stil von Laborgeräten.

4.2 Prinzipieller Aufbau der ICU

Nach den Vorgaben und den Anforderungen an die Hardware wurde beim prinzipiellen Aufbau vorrangig auf die flexible Erweiterbarkeit und Anpassbarkeit bei strikter Einhaltung der Modularität der ICU geachtet. Das daraus resultierende grobe Hardwarekonzept wird durch Abbildung 4.1 veranschaulicht. Die anschließende Beschreibung der Hardwarekomponenten erfolgt zunächst auf einer generellen Ebene. Eine detaillierte Beschreibung der einzelnen Platinen und deren Funktion sowie Beispiele aus dem derzeitigen Design erfolgt in den anschließenden Abschnitten.

Das Kernstück der ICU ist die Hauptplatine, auf der sich u. a. die zentrale Recheneinheit befindet über welche die protokollbasierte Ablaufsteuerung erfolgt und die eine Schnittstelle zum Anwender-PC bietet. Über einen 96-poligen Stecker ist die Hauptplatine mit einer Rückwand-Platine, der so genannten Backplane, verbunden. Zusätzlich zur Verteilung der Spannungsversorgung an alle angeschlossenen Platinen, werden vom Mainboard über die Backplane zu den Peripherie-Platinen sowohl interne Kontrollsignale als auch Signale von Schnittstellen zu den Systemkomponenten geleitet. Über die Backplane können so in der derzeitigen Version der ICU zwei Nebenplatinen (Slave I und Slave II) mit der Hauptplatine verbunden werden. Da die Belegung der Backplanestecker zu den Nebenplatinen identisch ist, kann über eine schlichte Ver-

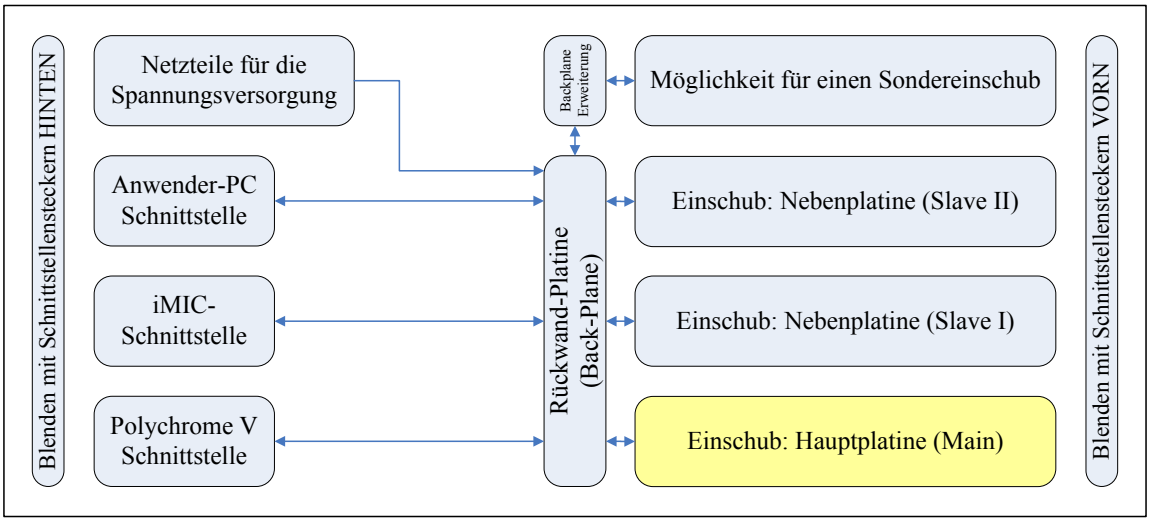


Abbildung 4.1: Prinzipieller Aufbau der ICU: Eine Backplane verbindet alle ICU-Einschübe. Der Abschluss der Einschübe zu Gehäusevorderseite erfolgt über Blenden, in denen z. B. Synchronisationsschnittstellen über Kontaktstecker zur Verfügung gestellt werden. Netzteile und Zugriff auf die PC-Schnittstelle sowie Sonderschnittstellen für z. B. das iMIC und den Polychrome V befinden sich in der Rückseite des Gehäuses.

vielfachung der Slave-Steckplätze problemlos eine Verbreiterung der Backplane erreicht werden, so dass zusätzliche Nebenplatinen angeschlossen werden könnten. Um bereits existierende Platinen, deren Steckerbelegung nicht konform mit der einer Nebenplatine ist, als Einschub in die ICU zu integrieren, kann die Backplane um eine Backplane Erweiterung (s. Blockdiagramm in Abbildung 4.1) ergänzt werden. Über diese Erweiterung können Anpassungen in der Signalführung und der Steckerbelegung, bzw. der Wahl des Steckers gemacht werden.

Alle Platinen, die von der Vorderseite der ICU in die Backplane eingesteckt werden, müssen auf einer Basisplatine im Europakartenformat[1] aufbauen. Der Abschluss dieser Basisplatine zur Gehäusevorderseite wird jeweils durch eine Frontplatte (10 Teileinheiten (TE) mal 3 Höheneineheiten (HE); entspr. 50,8 mm x 128,5 mm) gebildet. Auf der Platine aufgebrachte Bauteile oder Aufsteckplatinen dürfen nicht über die Projektion der Frontplatte hinausragen. Diese Art der Platinen wird im Folgenden „ICU-Einschub“ genannt.

In der rückseitigen Hälfte der ICU werden die Netzteile für die Spannungsversorgung aller ICU-Einschübe und der Versorgung des iMIC untergebracht. Zusätzlich werden die Schnittstellen zum Anwender-PC, zum iMIC (inklusive Versorgungsspannung) und zum Polychrome V über Kontaktstecker in den Blenden der Rückwand nach Außen geführt.

4.3 Logische Verknüpfung der Platinen

Bei der Konzeption der logischen Zusammenhänge zwischen den Platinen und hierbei letztendlich zwischen den eingesetzten elektronischen Bauteilen, stehen vor allem die Aspekte des deterministischen Verhaltens und der Echtzeit im Vordergrund. Der auf der zentralen Recheneinheit[2] der ICU verwendete **D**igitale **S**ignal **P**rozessor DSP bietet neben einer parallelen Schnitt-

[1] Das Europakartenformat ist in DIN 41494, Blatt 2 (1972) beschrieben.

[2] Die zentrale Recheneinheit wird von der Firma SmartMove bezogen. Einzelheiten zu den Bauteilen dürfen aus firmenpolitischen Gründen nicht genannt werden.

stelle diverse serielle Schnittstellen. Zwei identische serielle Schnittstellen können jeweils in einem Modus konfiguriert werden, der eine zeitlich gemultiplexte, synchrone, serielle Schnittstelle (**T**ime **M**ultiplexed **S**erial **I**nterface (TMSI)) darstellt. Dieser Modus ist ursprünglich für das Vernetzten von mehreren DSPs der verwendeten Chip-Familie vorgesehen. Von allen zur Verfügung stehenden Schnittstellen eignet sich das TMSI aufgrund ihres generellen Aufbaus am besten für die zu bewältigenden Aufgaben der ICU: Innerhalb eines konstanten Zeitraums (Frame) werden alle verbundenen Bauteile angesprochen, womit garantiert ist, dass eine Kommunikation zwischen dem Master (DSP) und jedem einzelnen Bauteil (Slaves) innerhalb eines Frames stattfindet. Das Flussdiagramm in Abbildung 4.2 veranschaulicht die Abläufe des TMSIs: Nach entsprechender Konfigurierung werden in n Time-Slots, die immer in gleicher Reihenfolge angeordnet sind, Daten gesendet und empfangen. Ordnet man den Time-Slots unterschiedliche Bauteile zu, werden im einfachsten Fall n Bauteile über das TMSI mit dem DSP verbunden.

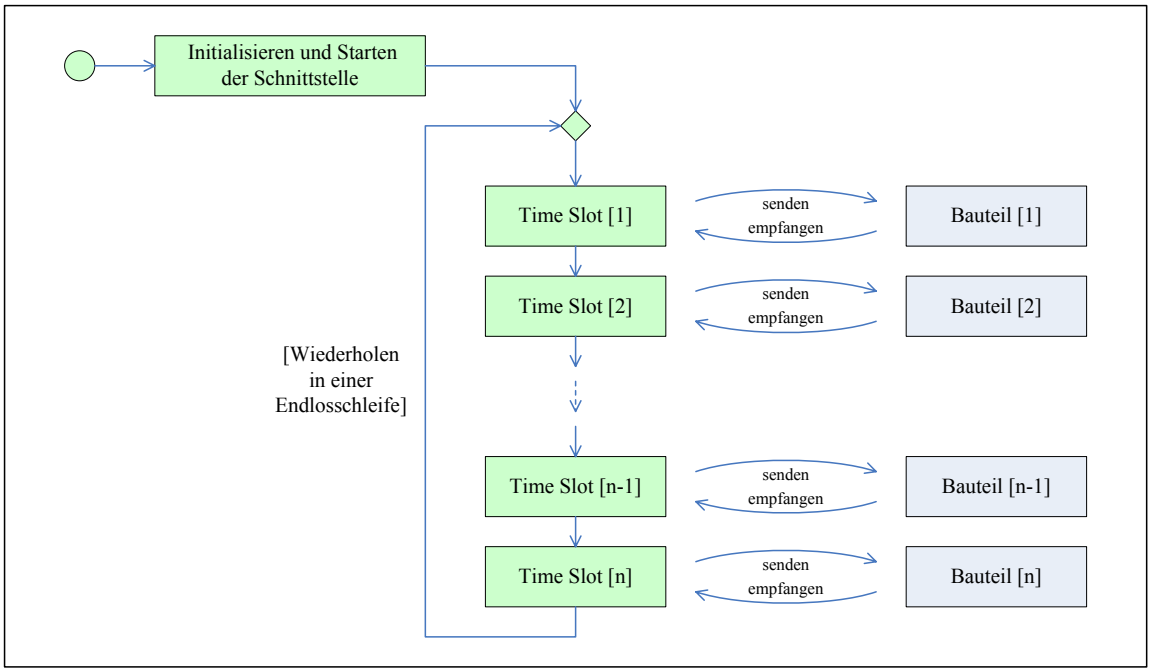


Abbildung 4.2: Flussdiagramm der zeitlich gemultiplexten, seriellen Schnittstelle, welche die logische Verknüpfung aller elektronischen Bausteine mit dem DSP darstellt. Die Kommunikation zwischen dem Master (DSP) und den einzelnen Slaves (Bauteile 1 bis n) erfolgt jeweils innerhalb eines Time-Slots. Alle Time-Slots (1 bis n) bilden zusammen einen Frame.

Die Struktur des TMSI auf elektrischer bzw. Protokollebene orientiert sich an dem sehr lockeren Standard des SPI [118], auch bekannt als Microwire. Dieser serielle Datenbus besteht aus drei Leitungen, an denen alle Teilnehmer (1 Master und n Slaves) parallel angeschlossen sind: **M**aster **O**ut; **S**lave **I**n (MOSI)-1, **M**aster **I**n; **S**lave **O**ut (MISO) und CLK (Serial Clock). Zusätzlich ist zum Ansteuern der Slaves jeweils eine eigene Leitung zwischen Master und Slave erforderlich (Slave Select). Soll ein Datentransfer zwischen Master und einem der Slaves erfolgen, muss dieser über die Slave Select Leitung für die Kommunikation aktiviert werden. Gezielte und effektive Kommunikation mit einzelnen bzw. ausgewählten Gruppen von Slaves ist somit möglich. Allerdings ist bei wechselndem Kommunikationsbedarf zwischen dem Master und mehreren Slaves nicht determiniert, wann die komplette Übertragung abgeschlossen ist.

Da der Master beim TMSI innerhalb eines Frames garantiert mit jedem Slave kommuniziert, ist hier die Übertragungszeit von Daten bzw. Kommandos entsprechend ihrer Länge (Anzahl der Bytes) genau definiert. Durch den Aufbau der Schnittstelle ist es wie z. B. beim Verbinden mehrerer DSPs möglich, dass das TMSI auf die Slave-Select Leitungen verzichtet und damit die Anzahl der benötigten Leitungen konstant und unabhängig von der Anzahl der angeschlossenen Bauteile ist. Allerdings wird hierbei vorausgesetzt, dass die angeschlossenen Bauteile (Slaves) entsprechend konfigurierbar sind und eigenständig einen vorbestimmten Time-Slot innerhalb eines Frames für die Kommunikation benutzten. Pro Time-Slot werden in der derzeitigen Konfiguration der TMSI jeweils acht Bit (entspr. einem Byte) übertragen. Wie die Kommunikation mit Bauteilen funktioniert, die einerseits mit dem üblichen SPI ausgestattet sind und daher eine Slave-Select Leitung benötigen und andererseits für eine komplette Übertragung von einem Daten- bzw. Kommandowort mehr als ein Byte benötigen, wird in einem späterem Abschnitt (s. Abschnitt 4.4) detailliert beschrieben.

Neben den der SPI-Konvention entnommenen Leitungen bietet das TMSI zusätzlich zwei weitere MOSI (MOSI-2, MOSI-3) und eine weitere Leitung zur generellen Synchronisation der seriellen Kommunikation (**F**rame **S**ync - FS). Die Leitungen MOSI-2 und MOSI-3 können optional für die Erweiterung der Sendebandbreite verwendet werden, während die Frame Sync Leitung für dauerhaft synchronen Betrieb der Schnittstelle unabdingbar ist. Insgesamt besteht das TMSI somit aus sechs Leitungen (CLK, FS, MOSI1-3 und MISO) von denen die zweite und dritte MOSI-Leitung optional verwendbar sind. Abbildung 4.3 veranschaulicht den logischen Aufbau eines Frames auf elektrischer bzw. Protokoll-Ebene in Form eines Zeitverlaufsdiagramms.

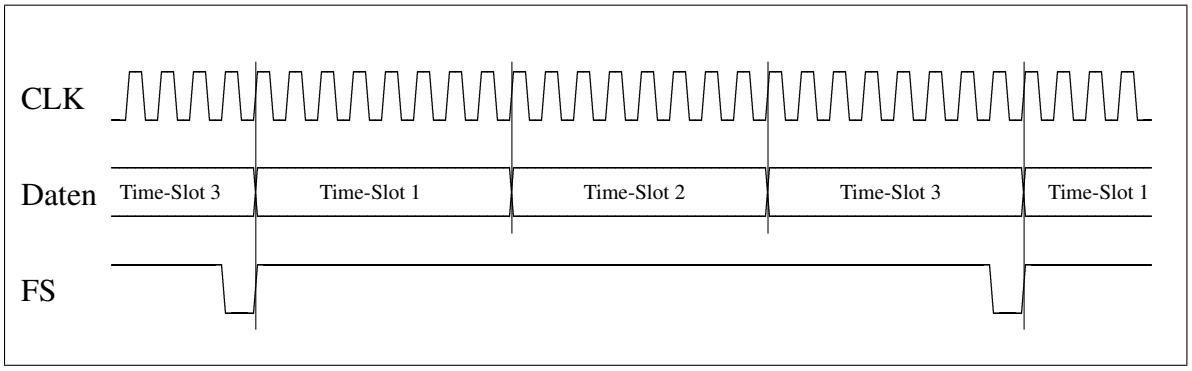


Abbildung 4.3: Vereinfachtes Zeitverlaufsdiagramm des TMSI bestehend aus Clock (CLK), Frame Sync (FS) und Daten (MOSI[1-3], MISO).

Nach der Initialisierung und dem Starten der Schnittstelle gibt der Master kontinuierlich einen Takt auf der entsprechenden Leitung aus. Das letzte Bit eines Frames wird auf der Frame-Synch Leitung markiert, so dass die Möglichkeit zur Synchronisation auf den Anfang eines jeden Frames gegeben ist. Auf den Datenleitungen (MOSI[1-3] und MISO) werden ab Beginn eines Frames Datenpakete in einzelnen Time-Slots in aufsteigender Reihenfolge übertragen.

Wie bereits erwähnt bietet der verwendete DSP *zwei* der beschriebenen TMSIs. Obwohl identisch aufgebaut, sind sie individuell konfigurierbar, so dass zwei verschiedene „Kommunikationspfade“ erzeugt werden können: Über die Einstellung des seriellen Takts und der Anzahl der Time-Slots pro Frame können so z. B. zwei Gerätegruppen mit jeweils unterschiedlichen Anforderungen an das Zeitverhalten separat angesprochen werden. Beide TMSIs sind von der

Hauptplatine ausgehend über die Backplane auf allen Nebenplatinen verfügbar. Der Ausstattung der ICU und der Bestückung der einzelnen Platinen entsprechend, können somit über eine adäquate Konfigurierung der TMSIs alle Schnittstellenbausteine abhängig von ihrer Funktion und ihren Anforderungen in das Gesamtsystem eingebunden werden.

4.4 Aufbau der Hauptplatine

Die Hauptplatine besteht aus einer Basisplatine im Standard Euroformat, auf die über Steckerleisten sowohl die Platine der zentralen Recheneinheit als auch optionale Platinen mit Analogtechnik bzw. Digitaltechnik zur Einbindung von entsprechenden Schnittstellen aufgesteckt werden können. Damit lassen sich, wie in Abbildung 4.4 gezeigt, die jeweiligen Schnittstellen einbinden. Auf der Basisplatine befinden sich ein 96-poliger Stecker als Verbindung zur Backplane, verschiedene Leitungstreiber, sowie ein **C**omplex **P**rogrammable **L**ogic **D**evice (CPLD), dessen zentrale Funktion im Weiteren noch detailliert vorgestellt wird.

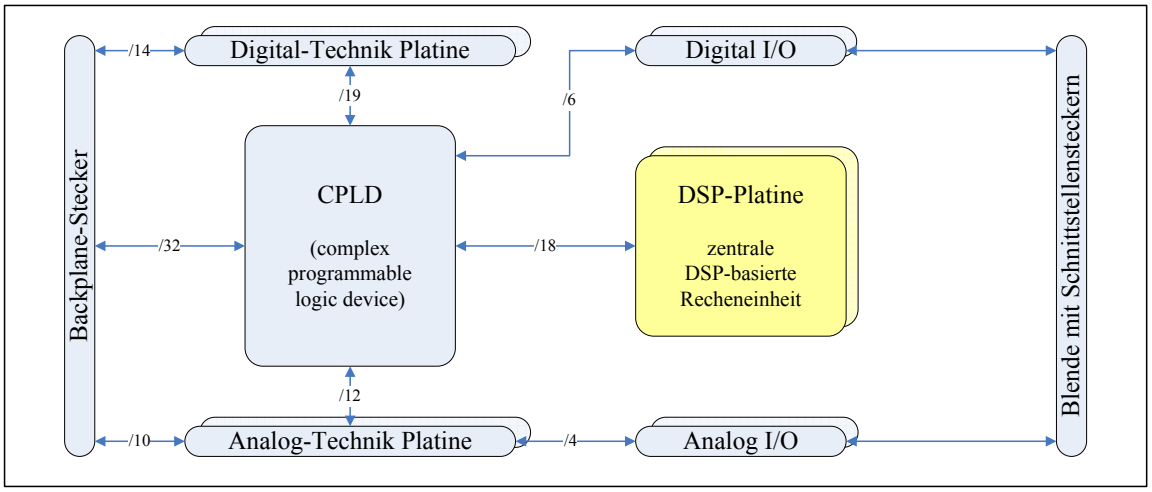


Abbildung 4.4: Schematischer Aufbau der ICU-Hauptplatine bestehend aus der Basis-Platine inklusive CPLD, der DSP-Platine als zentraler Recheneinheit, einer optionalen Analogtechnik-Platine und einer ebenfalls optionalen Digitaltechnik-Platine. Schattierte Blöcke kennzeichnen Aufsteck-Platinen; Zahlenwerte beschreiben die Anzahl der Leitungen (z. B. : „ /18 “ ≙ 18 Leitungen zwischen der DSP-Platine und dem CPLD).

Die DSP-basierte, zentrale Recheneinheit (im Folgenden: DSP-Platine) ist eine essentielle Komponente der ICU und als Aufsteckmodul ein fester Bestandteil der Hauptplatine. Der auf der Hauptplatine basierende ICU-Einschub wird daher auch als DSP-Einschub bezeichnet. Die DSP-Platine wird von der Firma Smart Move GmbH zugekauft[3]. Auf ihr befindet sich neben Bauteilen zur Anpassung der Spannungsversorgung, Flash-Speichereinheiten, einer EIA-232-Schnittstelle und einem weiteren CPLD, das Herzstück des Systems, der DSP. Das CPLD der DSP-Platine wird für den Bootvorgang des Prozessors, den Zugriff auf die Flash-Speichereinheiten und der EIA-232-Schnittstelle sowie zur flexiblen Führung von Signalleitungen auf die Steckerleisten benutzt. Über letztere ist die DSP-Platine mit der Basisplatine des DSP-Einschubs verbunden. Mit der EIA-232-Schnittstelle verfügt die DSP-Platine und somit die ICU über eine Standard PC-Schnittstelle, die allerdings mittelfristig durch eine USB-Schnittstelle ersetzt werden soll.

[3] Aktuell wird das Model: AR9632_5 verwendet.

Wie in Abbildung 4.4, erkennbar sind alle Leitungen die von der DSP-Platine über Steckerleisten auf die Basisplatine geführt werden mit dem CPLD der Basis-Platine verbunden. Vom CPLD ausgehend sind weitere Leitungen, sowohl mit dem Backplanestecker als auch mit den Steckerleisten zu den Platinen der Analog- und Digitaltechnik verbunden. Damit agiert das CPLD als zentraler, programmierbarer Knotenpunkt aller Signalleitungen und bietet die Möglichkeit sowohl Daten- als auch Kontrollsignale flexibel zu verteilen. Die CPLD interne Logik ermöglicht zudem die Erstellung von Schieberegistern, Zustandsmaschinen (state machines) und kleineren Datenspeichern zur Datenpufferung. Über logische Verknüpfungen können ferner zusätzlich benötigte Kontrollsignale generiert werden. Diese werden z. B. genutzt, um aus den in Abschnitt 4.3 beschriebenen TMSI-Signalen Slave-Select Kontrollsignale für Bausteine mit Standard SPI zu generieren. Da sich über das CPLD beliebige logische Verknüpfungen erzeugen lassen, können *einem* Slave auch *mehrere* TMSI Time-Slots zugeordnet werden, so dass Bausteine unterstützt werden können, deren SPI-Schnittstelle eine Wortbreite von ganzzahligen Vielfachen von acht Bit verwenden. Die auf diesem Prinzip aufbauende Basiskonfiguration der TMSI wird in Abschnitt 4.6 zusammengefasst.

Eine der primären Funktionen der ICU ist die physikalische Verbindung aller Systemkomponenten innerhalb eines Gesamtsystems. Wie aus dem Steuerungskonzept abgeleitet, ist es hierbei notwendig, dass die ICU in Art und Anzahl unterschiedliche Schnittstellen für die hardwarebasierte Synchronisation zur Verfügung stellt. Um die Auswahl der Schnittstellen möglichst modular, an verschiedene Systemkonstellationen anpassbar und flexibel erweiterbar zu gestalten, wurden daher keine spezifischen Schnittstellen direkt auf die Basis-Platine integriert. Vielmehr wurden Steckerleisten vorgesehen, über die Platinen mit den entsprechenden Schnittstellenbausteinen in die ICU eingebracht werden können. Die Steckerleisten setzen sich aus zwei Typen von Kontaktpins zusammen:

- *Bausteinkontrollpins:* Diese Pins werden zum CPLD der Basis-Platine geführt, wodurch sie wie beschrieben flexibel konfiguriert und damit z. B. als CLK, MOSI, MISO oder Slave Select Leitung verwendet werden können.
- *Schnittstellenspezifische Pins:* Da je nach Auswahl der Schnittstellen die entsprechenden Signale unterschiedliche, elektrische Eigenschaften haben können, werden sie von den schnittstellenspezifischen Pins nicht zum CPLD, sondern unmittelbar an den Backplane-Stecker geführt. Von hier werden sie über die Backplane an entsprechende Nebenplatinen geleitet oder dem Benutzer über Kontaktstecker in den Blenden der Rück- bzw. Vorderseite der ICU direkt zur Verfügung gestellt.

Die Basis-Platine wurde mit separaten Steckerleisten für Analog- und Digitaltechnik ausgestattet, um die unterschiedlichen Anforderungen an die Signalführung und Abschirmung der Leitungen berücksichtigen zu können. Außer der Unterscheidung zwischen Analog- und Digitaltechnik ist die Verwendung der Schnittstellen spezifischen Leitungen in Bezug auf ihre elektrischen Eigenschaften in keiner Weise eingeschränkt. Dies ermöglicht, dass die Platinen zum Aufstecken auf die Steckerleisten gemäß den Anforderungen der Anwendungen flexibel gestaltet werden können und lediglich durch die Anzahl der zur Verfügung stehenden Bausteinkontrollpins und der schnittstellenspezifischen Pins limitiert sind. Wie aus Abbildung 4.4 ersichtlich, ist die Platine für Analogtechnik mit 12 Bausteinkontroll- und 10 schnittstellenspezifischen Leitungen ausgestattet. Die Digitaltechnik-Platine hingegen ist mit 19 Bausteinkontroll-

und 14 schnittstellenspezifischen Leitungen ausgestattet. Beispiele für beide Platinenarten werden in Abschnitt 4.6 vorgestellt.

Um einen Teil der Schnittstellen, welche die Hauptplatine selbst bzw. durch die darauf aufgesteckten Platinen bereitstellt, dem Benutzer unmittelbar durch Kontaktstecker in der Blende an der Vorderseite der ICU zur Verfügung stellen zu können, wurden zwei weitere Steckerleisten für analoge und digitale Signale auf der Basis-Platine vorgesehen. Über die Steckerleisten können diese Signale entweder unmittelbar an einen Kontaktstecker geführt werden, oder es besteht die Möglichkeit über entsprechende, zusätzliche Elektronik etwaige Pegelanpassungen oder Ähnliches vorzunehmen.

4.5 Möglichkeiten zur Erweiterung der ICU-Hardware

Im vorangegangenen Abschnitt wurden die Möglichkeiten zur flexiblen Gestaltung der Platinen für Analog- und Digitaltechnik beschrieben. Sie lassen sich über Steckerleisten auf die Hauptplatine aufstecken. Einzige Einschränkung bei der Wahl und der Anzahl der Schnittstellen sind hier die zur Verfügung stehenden Bausteinkontrollpins und schnittstellenspezifischen Pins. In Abschnitt 4.2 wurde beschrieben, dass neben dem DSP-Einschub (Hauptplatine) zusätzliche (optionale) Slave-Einschübe (Nebenplatinen) hinzugefügt werden können (vgl. Abbildung 4.1). Bei den ICU Slave-Einschüben kann es sich prinzipiell um zwei Typen handeln:

- *Spezial-angefertigte-Einschübe:* Hierunter fallen alle Einschübe, deren Platinen an die jeweilige Anwendung speziell angepasst worden sind, sich dabei aber an der Pinbelegung des Backplanesteckers für Nebenplatinen orientieren. Die Positionierung der Einschübe ist daher innerhalb der verfügbaren Steckplätze für Nebenplatinen frei wählbar (vgl. Nebenplatinen (Slave I/II) in Abbildung 4.1).
- *Hauptplatinen-Basierte-Einschübe:* Bei der Konzeption der Basis-Platine des im vorangegangen Abschnitt beschriebenen DSP-Einschubs (Hauptplatine) wurde darauf geachtet, dass die Platine neben ihrer primären Funktion als Träger der zentralen Recheneinheit zusätzlich als Nebenplatine eingesetzt werden kann. Hierdurch können die beschriebenen Leistungsmerkmale der Hauptplatine durch schlichtes Hinzufügen der gleichen Hardware verdoppelt bzw. vervielfacht werden. Die Erweiterung der ICU geschieht auf diese Weise ohne Neuentwicklung von Hardware, mit den damit verbundenen Einsparungen an Entwicklungszeit und Entwicklungskosten. Neben der Erweiterung der Rechenleistung durch weitere DSPs können zusätzlich sowohl die Anzahl als auch die Vielfalt der Schnittstellen (Analog- und/oder Digitaltechnik) erhöht werden.

Beim Erweitern durch Nebenplatinen besteht zudem die Möglichkeit die zusätzlich geschaffenen Schnittstellen über Kontaktstecker in den Blenden der Einschübe dem Benutzer an der Vorderseite der ICU zur Verfügung zu stellen.

Obwohl sich die ICU-Einschübe prinzipiell an den Konventionen (u. a. Pinbelegung des Backplanesteckers) für Nebenplatinen orientieren sollten, besteht die Möglichkeit Sondereinschübe mit arbiträren Anschlüssen (Steckertyp und/oder Pinbelegung) in die ICU zu integrieren. Dies setzt allerdings voraus, dass diese Einschübe auf einer Grundplatine im Europakartenformat basieren und eine maximale Dicke von 10 TE (10 TE = 50,8 mm) nicht überschreiten. Zum

Anschluss der Sondereinschübe an die reguläre Backplane und damit an den Rest der ICU-Hardware muss allerdings, wie in Abbildung 4.1 veranschaulicht, eine Backplane-Erweiterung konstruiert werden. Über diese Erweiterung müssen von der Backplane ausgehend, die vom Sondereinschub benötigten Signale auf dessen Anschlussstecker und/oder Steckerbelegung angepasst werden.

Beispiele für speziell angefertigte Einschübe, Hauptplatinen-basierte-Einschübe und Sondereinschübe, werden im folgenden Abschnitt (Abschnitt 4.6) präsentiert.

4.6 Derzeitige Hardwarekonfiguration der ICU

Die ICU wird bereits seit Mitte des Jahres 2005 als fester Bestandteil des TILL Photonics „Imaging-Systems“ eingesetzt und hat damit die alte Kontroll-Einheit ersetzt. Zudem wird die ICU ebenfalls seit 2005 im BIZ innerhalb von Systemen zur Anwendung moderner Mikroskopietechniken (FRET, TIRF, FRAP, Nipkow, Laser-Scanning und TDI vgl. Abschnitt 2.2.2) basierend auf der Mikroskopplattform iMIC verwendet. Überdies wird die ICU als Teil des Mikroskopiesystems innerhalb des in Abschnitt 2.2.4 beschriebenen Systems zur Langzeitzellanalyse eingesetzt und könnte hier zusätzliche Komponenten, wie den Pipettierroboter, in zeitlich vorbestimmten Abläufen steuern.

Im Hinblick auf den Einsatz der ICU innerhalb dieser Anwendungen wurden sowohl Analog- und Digitaltechnik-Platinen zum Aufstecken auf die Hauptplatine des DSP-Einschubs als auch ICU-Einschübe auf Basis von Nebenplatinen entworfen. Die hiermit geschaffenen Schnittstellen bieten bereits alle erforderlichen Mittel für hardwarebasierte Synchronisation der verwendeten Systemkomponenten. Die aktuelle Hardwarekonfiguration der Hardware wird in den folgenden Abschnitten vorgestellt.

4.6.1 DSP-Einschub: Hauptplatine

Wie in Abschnitt 4.4 beschrieben wurde, bietet das CPLD vielseitige Möglichkeiten zur logischen Verarbeitung von digitalen Signalen. U. a. wird dies genutzt, um mittels der seriellen Schnittstelle TMSI über Schieberegister digitale I/Os zu erzeugen. Diese digitalen Schnittstellen werden vom CPLD an den Backplanestecker und an die bereits erwähnte Steckerleiste für digitale I/Os geführt (vgl. Abbildung 4.4). Ohne zusätzliche analoge oder digitale Aufsteckplatinen bietet der DSP-Einschub somit bereits digitale Schnittstellen, die in der derzeitigen Ausführung der ICU für die Synchronisation von CCD-Kameras und der Beleuchtungsquelle Polychrome V verwendet werden. Für die Einbindung verschiedener Kameras über Trigger- bzw. Sync-Leitungen wurde eine Schaltung zur Pegelanpassung und zur Verknüpfung eines digitalen Eingangs und eines Ausgangs zu einer bidirektionalen Synchronisationsleitung vorgesehen. Momentan hat der Benutzer somit über BNC - Steckverbinder in der Blende der Hauptplatine Zugriff auf einen digitalen Eingang, einen digitalen Ausgang und eine bidirektionale Synchronisationsleitung (s. Abbildung 4.6).

Aufsteckplatine: Analogtechnik

Alle Anwendungen, in denen die ICU bereits eingesetzt wird (vgl. Abschnitt 4.6), verwenden die Mikroskopplattform iMIC. Die Fokussierung innerhalb des iMIC wird durch einen zweistufigen z-Trieb für Grob- und Feinfokussierung realisiert, mittels dem das benutzte Objektiv be-

wegt wird. Kombiniert mit einer schrittmotorbasierenden Grobpositionierung erfolgt die Feinpositionierung des Objektivs über einen Piezo-Aktor. Das Reglermodul für den Piezo-Aktor wird über ein analoges Signal zwischen 0 und 10 V angesteuert. Weitere Analoge Signale werden bei Anwendungen wie der Laser-Scanning-Mikroskopie oder der Durchlichtbeleuchtung für die Ansteuerung von Combinern und Abschwächern benötigt. Hierdurch wird eine Selektion bzw. Einstellung der Intensität der einzelnen Laser oder LEDs gewährleistet.

Um nicht nur der jetzigen Minimalanforderung gerecht zu werden, sondern mittelfristig auch steigenden Ansprüchen der Systeme zu genügen, wurde eine Aufsteckplatine mit Analogtechnik entworfen, die der ICU in Summe 8 analoge Ausgänge und zwei analoge Eingänge bereitstellt. Die Spezifikation der einzelnen Schnittstellen ergibt sich wie folgt:

- *DAC I:* Vierkanal-Digital-Analog-Wandler mit einer Auslösung von 16 Bit
- *DAC II:* Vierkanal-Digital-Analog-Wandler mit einer Auslösung von 12 Bit
- *ADC:* Zweikanal-Analog-Digital-Wandler mit einer Auslösung von 12 Bit

Die Ausgangs- bzw. Eingangsspannungen der Analogkanäle können innerhalb des Bereichs von maximal ± 10 V für jeden Kanal individuell durch entsprechende Bestückung der Platine konfiguriert werden.

Aufsteckplatine: Digitaltechnik

Wie in Abschnitt 3.4 beschrieben wurde, können EIA-232-Schnittstellen unter Berücksichtigung der Übertragungszeiten prinzipiell in Echtzeit betrieben werden. Hierfür dürfen allerdings keine softwareseitigen Latenzen auftreten, wie es z. B. in einem Windows-basierten System üblich ist. Da mit der ICU eine Echtzeitplattform geschaffen wurde, in der alle Hardwareschnittstellen innerhalb eines festen Zeitrahmens angesprochen werden, können ebenfalls Bausteine für EIA-232-Schnittstellen (UARTs) integriert werden.

In der Mikroskopieplattform integriert, befinden sich die Steuerungen für die Schrittmotoren der motorisierten Komponenten des iMIC (Objektivwechsler, Filter-Wechsler, Grobfokussierung des Z-Triebs, XY-Tisch). Verwendet werden Schrittmotorsteuerungen der Firma Dr. Kassen GmbH[4], die im Falle des iMIC über EIA-232-Schnittstellen betrieben werden. Obwohl prinzipiell als Punkt-zu-Punkt-Verbindung konzipiert, können mittels eines speziellen Verfahrens mehrere Steuerungen über eine EIA-232-Schnittstelle angesprochen werden. Durch sinnvolles Gruppieren werden so in zwei Gruppen jeweils drei Steuerungen über eine EIA-232-Schnittstelle mit der ICU verbunden. Hierbei wurde darauf geachtet, dass Achsen, die i. d. R. gleichzeitig verfahren werden (z. B. x- und y-Achse eines motorisiert verfahrbaren Tischs), nicht an der gleichen EIA-232 Schnittstelle angeschlossen werden. Um in einer Anwendung die Schrittmotorsteuerungen für alle sechs Achsen des iMIC betreiben zu können, bedarf es zweier EIA-232-Schnittstellen.

Durch die Möglichkeit zur flexiblen Gestaltung der Digitaltechnik-Platine können überdies weitere Kommunikationsschnittstellen wie z. B. CAN [92], TTCAN [183], Bluetooth [109], etc. integriert werden. Zum jetzigen Zeitpunkt würden diese Schnittstellen allerdings noch keine

[4]Derzeit wird das Model Pollux - OEM Type-2 verwendet.

Anwendung innerhalb der vorgestellten Systeme finden. Überdies muss abhängig von der jeweiligen Anwendung geprüft werden, ob die zeitlichen Anforderungen des Datentransfers erfüllt werden.

Das aktuelle Design der Digitaltechnik-Platine bietet bei voller Bestückung sechs EIA-232-Schnittstellen. Die Signalleitungen aller Schnittstellen werden zum Backplanestecker geführt, über den sie entweder an eine Nebenplatine innerhalb der ICU weitergeleitet, oder über Kontaktstecker dem Benutzer zur Verfügung gestellt werden. Zwei der Schnittstellen sind für den Anschluss des iMIC vorbehalten, alle weiteren Schnittstellen können individuell und nach den Ansprüchen des jeweiligen Systems eingesetzt werden.

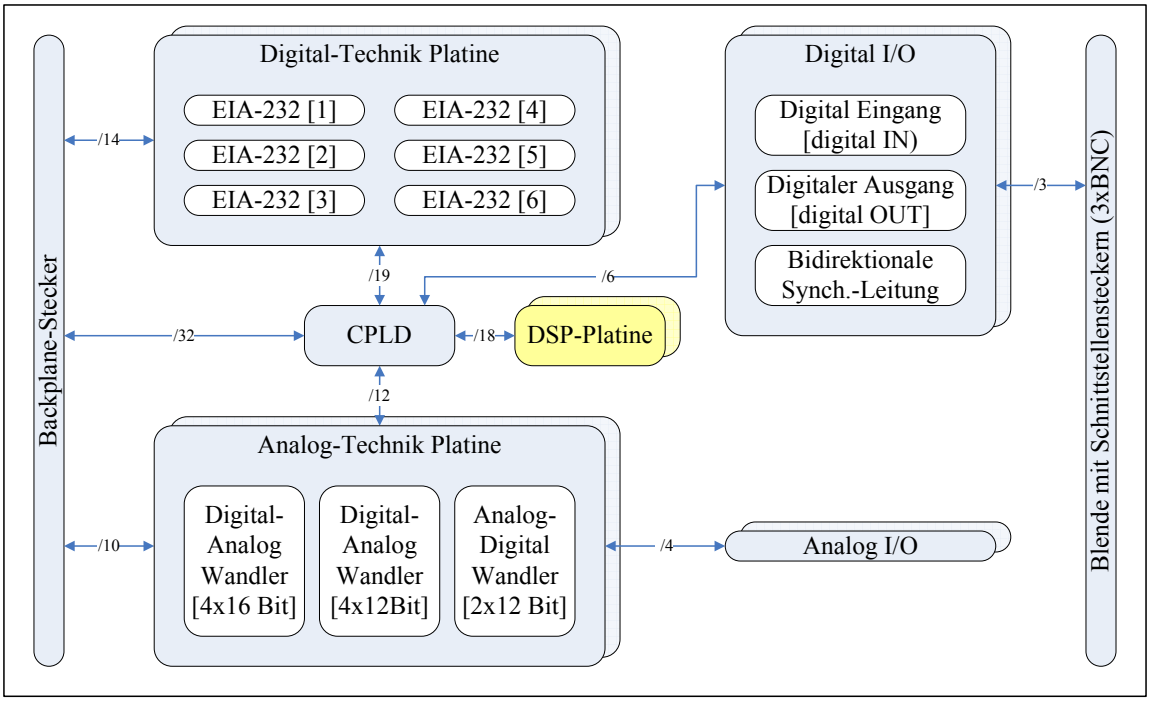


Abbildung 4.5: Schematische Darstellung der aktuellen Hardwarekonfiguration der ICU-Hauptplatine. Schattierte Blöcke kennzeichnen Aufsteck-Platinen. Die bereitgestellten Schnittstellen decken den kurz- und mittelfristigen Bedarf der Systeme, in denen die ICU eingesetzt wird.

Das Block-Diagramm in Abbildung 4.5 fasst die derzeitige Hardwarekonfiguration der ICU-Hauptplatine und deren Aufsteck-Platinen für Analog- und Digitaltechnik sowie die Anschlussmöglichkeiten von digitalen I/O über BNC-Steckverbinder in der Blende an der ICU-Vorderseite zusammen. Es sei darauf hingewiesen, dass dem Benutzer durch die ICU-Hauptplatine derzeit nur ein Teil der digitalen Schnittstellen (digital I/O) über BNC-Steckverbinder direkt zur Verfügung stehen und dass die Möglichkeit zur Bereitstellung von Analogsignalen über die Steckerleiste „Analog I/O“ derzeit nicht in Anspruch genommen wird. Alle beschriebenen Schnittstellen können allerdings über die Backplane von Nebenplatinen verwendet werden bzw. über Kontaktstecker in deren Blenden an der ICU-Vorderseite dem Benutzer zur Verfügung gestellt werden. Die Anschlussmöglichkeiten der ICU-Hauptplatine sind somit derzeit noch nicht vollends ausgeschöpft. Abbildung 4.6 zeigt eine Fotographie der Hauptplatine in der derzeitigen Version (Version 1.1).

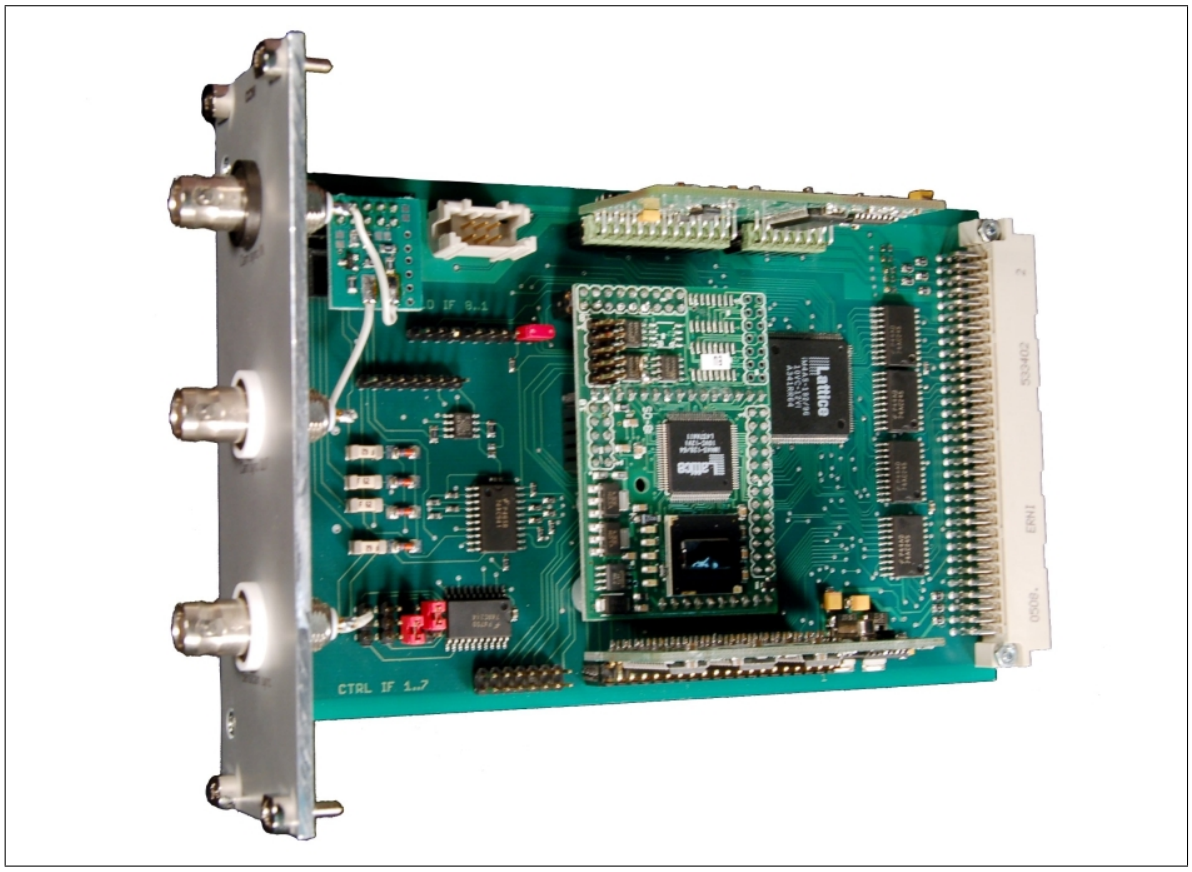


Abbildung 4.6: ICU-Hauptplatine bestehend aus der Basis-Platine (x1) mit CPLD (x2), der DSP-Platine (x3), einer optionalen Analogtechnik-Platine (x4), einer ebenfalls optionalen Digitaltechnik-Platine (x5), der Digital I/O-Platine (x6), dem Backplanestecker (x7) und der vorderen Aluminiumblende mit Schnittstellen-Steckern (BNC).

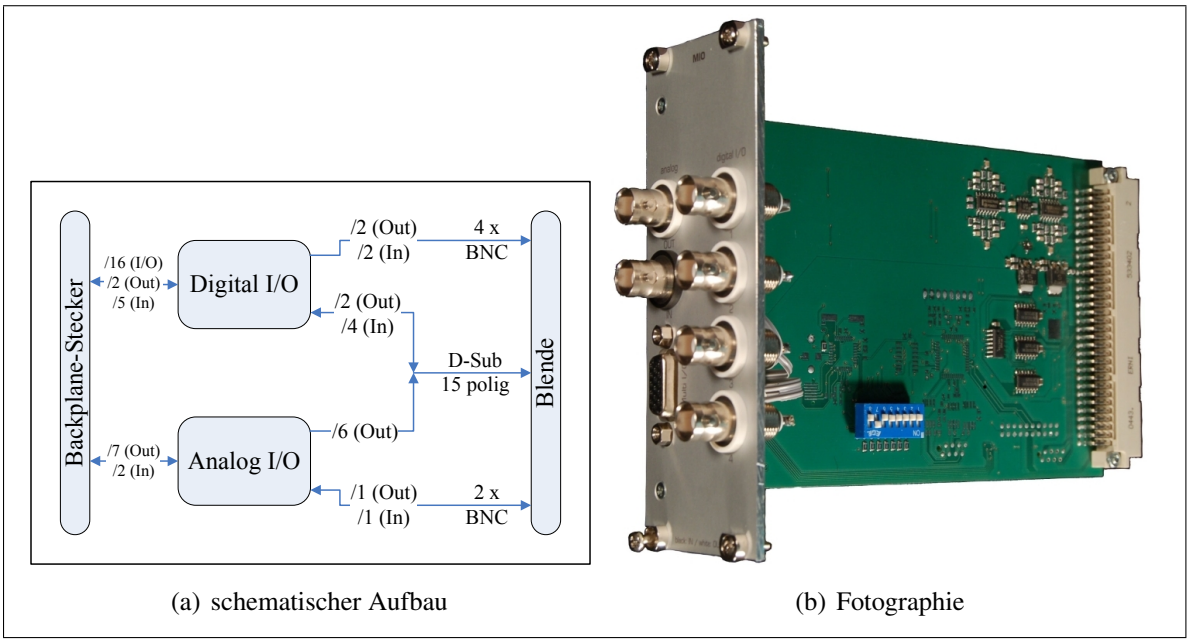
4.6.2 Einschub: Nebenplatinen

In Abschnitt 4.5 wurden bereits die zwei unterschiedlichen Typen von ICU-Einschüben für die Steckplätze der Nebenplatinen beschrieben. Im Folgenden werden die bereits existierenden Einschübe in der aktuellen Hardwarekonfiguration vorgestellt:

Einschub: Multi I/O

Um dem Benutzer die bereits auf der Hauptplatine etablierten, bisher aber noch nirgendwo verwendeten, analogen und digitalen I/Os zur Verfügung zu stellen, wurde ein ICU-Einschub nach den Konventionen für Nebenplatinen entworfen, der die entsprechenden Signale von der Backplane auf Steckverbinder in der Blende der ICU-Vorderseite führt (s. Abbildung 4.7).

Wie bereits beschrieben, ist eines der ersten Anwendungsgebiete der ICU die Realisierung von modernen Mikroskopietechniken mit Hilfe des iMIC. Werden in Systemen der Laser-Scanning-Mikroskopie für die Erstellung mehrfarbiger Fluoreszenzbilder unterschiedliche Laser eingesetzt, kommt als Laser-Combiner ein so genannter AOTF zum Einsatz. Die Ansteuerung dieses Laser-Combiners erfolgt über je ein digitales, sowie ein analoges Signal pro verwendetem Laser. Über das analoge Signal wird die Intensität der einzukoppelnden Laserstrahlen gesteuert. Mittels digitaler Signale können die einzelnen zur Beleuchtung benötigten Laser ausgewählt werden (Shutterfunktion). Ähnliches gilt für die Verwendung einer LED für Durchlichtbeleuch-



(a) schematischer Aufbau (b) Fotographie

Abbildung 4.7: ICU-Einschub „Multi I/O“: Noch nicht anderweitig verwendete Schnittstellen werden dem Benutzer über Kontaktstecker in der Blende der ICU-Vorderseite zur Verfügung gestellt.

tung bzw. für die Verwendung von LED-Combinern beim Einsatz verschiedenfarbiger LEDs. Um *eine* Verbindung zwischen der ICU und diesen Geräten zu ermöglichen, wurde ein 15-poliger D-Sub Steckverbinder vorgesehen, der alle benötigten Signale vereint, um über einen Combiner bis zu sechs Laser oder LEDs zu betreiben. Über einen sechsfach DIP-Switch kann die Belegung des D-Sub Steckverbinders entsprechend Tabelle 4.1 für weitere Anwendungen konfiguriert werden:

Bisher nicht verwendete Signale, wie z. B. die beiden analogen Eingänge, werden über den Multi I/O Einschub an BNC-Steckverbinder in der Blende der ICU-Vorderseite geführt und damit ebenfalls dem Benutzer zur Verfügung gestellt.

Mittelfristig ist geplant einen Teil der noch frei verfügbaren analogen I/Os für die Realisation eines so genannten Fotometrie-Systems zu verwenden. Die Fotometrie erlaubt sowohl qualitative und quantitative Nachweise als auch die Verfolgung der Dynamik chemischer Prozesse von strahlungsabsorbierenden chemischen Verbindungen [14]. Nach der Detektion sind zwei fotometrische Signale auf einer Analogleitung überlagert. Von einem analogen Eingang aufgenommen, könnten diese Signale mittels digitaler Signalverarbeitung auf dem DSP der Hauptplatine von einander getrennt und auf zwei getrennten analogen Ausgängen wieder ausgegeben werden.

Einschub: Digitale Scanner Kontrollsteuerung (DSC)

In Abschnitt 3.2.4 wurde im Rahmen der Beschreibung von Punkt-Scan Techniken die am BIZ entwickele Digitale Scanner Kontrollsteuerung (DSC) vorgestellt, die bis zu vier digitale Regler für galvanometrische Motoren ansteuern kann. Über diesen Controller und die daran angeschlossenen digitalen Regler können so für Anwendungen der Laser-Scanning-Mikroskopie z. B. beide Achsen eines Scankopfs so angesprochen werden, dass ein Laserstrahl auf im Vorfeld spezifizierte geometrische oder arbiträre Kurven bzw. Flächen abgelenkt werden kann. Als zen-

Pin Nr.	Pin Beschreibung	
	DIP-Switch Position 1	DIP-Switch Position 2
1	A-Out	
2	A-Out	
3	DGND	
4	D-Out	D-In
5	D-Out	D-In
6	A-In	A-Out
7	A-Out	
8	AGND	
9	D-In	D-Out
10	D-In	D-Out
11	A-Out	
12	A-Out	
13	AGND	
14	D-In	D-Out
15	D-In	D-Out

Tabelle 4.1: MIO-Einschub: Konfigurationsmöglichkeiten der Belegung des D-Sub Steckverbinders

trale Recheneinheit verwendet die DSC die gleiche DSP-Platine, die auch im DSP-Einschub zum Einsatz kommt. Daher wurde nach einem autarken Aufbau der Steuerung für Testzwecke ein *Hauptplatinen-basierter ICU-Einschub* (vgl. Abschnitt 4.5) für die Integration in die ICU konzipiert. Dabei hat sich allerdings herausgestellt, dass mit der Basis-Platine des Haupt-Einschubs nicht alle gewünschten Funktionen der DSC realisierbar sind. Als Konsequenz wurde ein *speziell angefertigter Einschub* entworfen, dessen Funktionsblöcke in Abbildung 4.8 schematisch dargestellt sind.

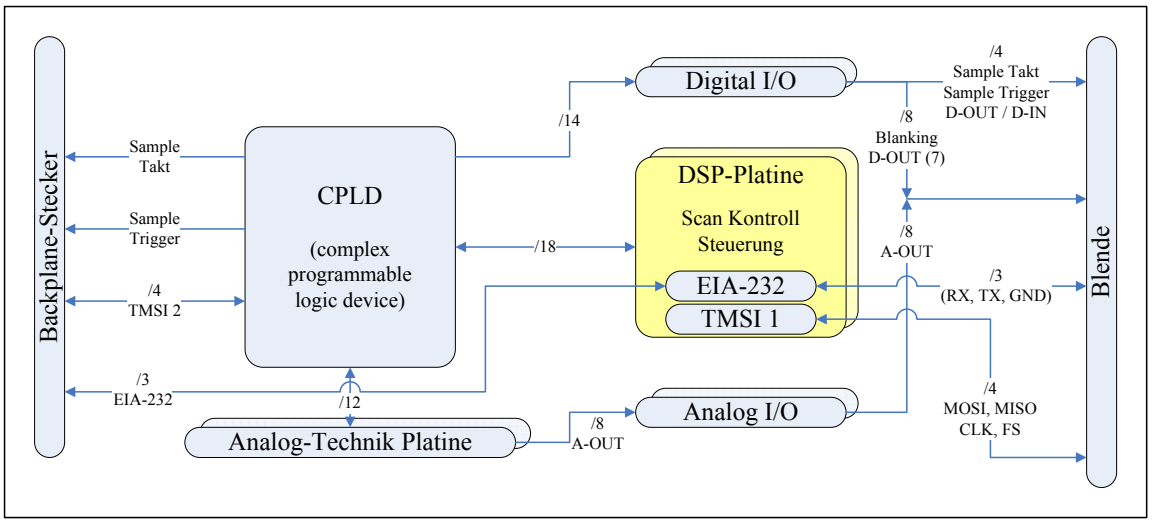


Abbildung 4.8: Schematische Darstellung des ICU-Einschubs „DSC“ (Digital Scan Control): Eine DSP-Basierte, digitale Ansteuerung von digitalen Reglern für galvanometrische Motoren, wie sie z. B. zum Betreiben von Scanköpfen eingesetzt werden (bis zu vier Achsen).

Die zum Betreiben eines Scankopfs verwendeten Regler werden digital mit einer Auflösung von 16Bit bei 100 kHz Taktung angesteuert. Zwei im Verbund synchron betriebene Achsen eines Scankopfs werden über je einen Regler bedient. Die DSC verwendet die bereits in Abschnitt 4.3 beschriebene serielle Schnittstelle TMSI, um gleichzeitig bis zu vier Achsen anzusteuern. Zusätzlich zur Steuerung der Regler generiert die DSC digitale Signale, die von etwaigen Detektionsgeräten (CCD-Kameras oder entsprechende Geräte zum Auslesen von Photomultiplier) als „Sample Takt“ und/oder „Sample Trigger“ verwendet werden können. Diese Signale werden für den ICU-internen Gebrauch an den Backplanestecker und für die Synchronisation von externen Detektionsgeräten an SMB-Steckverbinder in der Blende an der Platinenvorderseite geführt. Da in Anwendungen der Laser-Scanning-Mikroskopie sehr häufig zusätzlich Laser Abschwächer oder Laser-Combiner eingesetzt werden, wurde ebenfalls die Möglichkeit zu deren Steuerung berücksichtigt: Die DSC bietet hierfür bis zu jeweils acht analoge und digitale Ausgänge. Alle beschriebenen Schnittstellen sowie die mittelfristig vorgesehene Anbindung des DSC-Einschubs an die Hauptplatine werden über das TMSI realisiert.

(a) Fotographie des Einschubs (Prototyp) (b) Fotographie (autonom mit Netzteilen)

Abbildung 4.9: ICU-Einschub „DSC“: DSP-Basierte, digitale Ansteuerung von digitalen Reglern für galvanometrische Motoren. Der Einschub „DSC“ kann entweder als ICU-Einschub (a) oder autonom mit entsprechenden Netzteilen verwendet werden (b).

Vorrangig soll die Digitale Scanner Kontrollsteuerung als ICU-Einschub innerhalb von Gesamtsystemen verwendet werden, welche die Mikroskopieplattform iMIC in Laser-Scanning-Anwendungen integriert. Allerdings ist es aus betriebswirtschaftlichen Gründen wünschenswert, dass die DSC ebenfalls autonom als eigenständiges Gerät betrieben werden kann. Für diesen Fall wird die EIA-232-Schnittstelle des DSPs an den Backplanestecker und von dort aus zu einem Steckverbinder an der Gehäuserückseite geführt. Eine Konfigurierung bzw. das Betreiben der DSC kann somit von einem PC aus über diese Standardschnittstelle erfolgen.

Einschub: Analog-Signal-Akquisitions-Module (ASAM)

Photomultiplier wurden bereits mehrfach als Detektionsgeräte innerhalb von Anwendungen der **L**aser-**S**canning **M**icroscopy angesprochen und ihre Funktionsweise in Abschnitt 3.2.2 erläutert. Der zur detektierten Lichtintensität proportionale Strom eines Photomultipliers wird in ein analoges Ausgangssignal gewandelt und muss zur weiteren Verarbeitung digitalisiert und zum Anwender-PC übertragen werden. Innerhalb von Testaufbauten wurde am BIZ hierfür ein PC mit einer entsprechend ausgestatteten PCI-Karte von National Instruments[5] bestückt. Die Steuerung des gesamten Systems, sowie das Auslesen und Verarbeiten der (Bild-)Daten geschieht daher über die dazugehörige LabVIEW-Software[6]. Während die mit der Verwendung der National Instruments Hard- und Software verbundenen, hohen Anschaffungskosten im Rahmen von Testaufbauten vertretbar sind, ist von betriebswirtschaftlicher Seite für größere Stückzahlen von verkauften Systemen eine integrierte und damit kostengünstigere Lösung wünschenswert.

Im Rahmen einer Diplomarbeit ist ein weiterer ICU-Einschub entstanden, der in die Kategorie *speziell angefertigter Einschub* fällt (vgl. Abschnitt 4.5): **A**nalog-**S**ignal **A**cquisition **M**odule (ASAM). Obwohl die Verwendung des ASAMs primär in Zusammenhang mit der Benutzung von Photomultipliern steht, wurde ein generischer Entwurf angestrebt, der es generell erlaubt analoge Signale schnell und präzise zu digitalisieren, über entsprechende Algorithmen zu mitteln und an einen PC zu übertragen. Die Anforderungen und nähere Spezifikation der Leistungsmerkmale des ASAMs wurden durch die Anwendungen der **L**aser-**S**canning **M**icroscopy folgendermaßen vorgegeben:

- Vier analoge, differentielle Kanäle,
- Eingangsspannungsbereich pro Kanal von jeweils $\pm 12\,\mathrm{V}$,
- individuelle Konfiguration eines analogen Offsets pro Kanal von $\pm 12\,\mathrm{V}$,
- individuelle Konfiguration einer analogen Verstärkung pro Kanal von 1x bis 24x,
- Digitalisierung der analogen Signale mit einer Genauigkeit von 12 Bit bei einer maximalen Digitalisierungsrate von 5 MHz pro Kanal,
- konfigurierbares Oversampling (Überabtastung) mit mindestens Faktor fünf,
- konfigurierbare Wahl der Mittelungsverfahren (arithmetischer, geometrischer, harmonischer, logarithmischer oder laufender Mittelwert),
- Vorgabe der Zeitpunkte der Digitalisierung von außen über ein digitales Signal („Sample Takt“),
- konfigurierbare Anzahl von Messwerten, die innerhalb eines Zyklus digitalisiert, gemittelt und zum PC übertragen werden,
- starten eines solchen Zyklus über ein weiteres, externes, digitales Signal („Sample Trigger“).

Über die Kombination des geforderten Offset und der Verstärkung ist es möglich, aus dem Eingansspannungsbereich von $\pm 12\,\mathrm{V}$ einen beliebigen Bereich mit einer Differenz von 1 V als neuen Eingangsspannungsbereich zu definieren. Dabei wird der gewünschte Eingangsspannungsbereich zunächst über den Offset symmetrisch um 0 V verschoben und anschließend mittels

[5] Eingesetzt wird eine National Instruments PCI-Karte (Model NI-PCI6111). Die Anschaffungskosten liegen bei ~2.300 EUR.

[6] Verwendet wird die LabVIEW-Software in Version 8 „base“. Die Anschaffungskosten liegen bei ~1.400 EUR.

der Verstärkerschaltung auf den vom Wandler wahrgenommenen Spannungsbereich von ±12 V verstärkt. Beispiele, welche diese Möglichkeiten der Konfiguration veranschaulichen, sind in Abbildung 4.10 dargestellt.

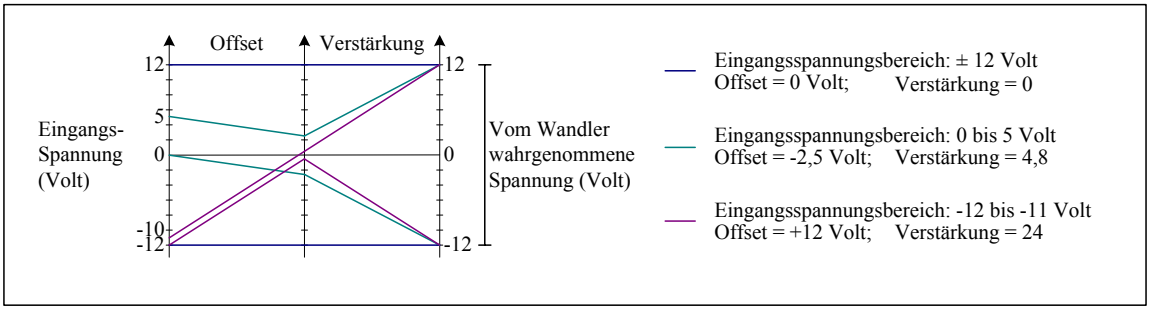


Abbildung 4.10: Konfiguration des Eingangsspannungsbereichs eines ASAM-Kanals anhand von Beispielen

Das nach den Vorgaben entworfene Design realisiert den Offset und die Verstärkung über eine analoge Schaltung mit Hilfe von Operationsverstärkern. Durch analoge Verarbeitung der Eingangssignale vor der Digitalisierung werden die Empfindlichkeitsbereiche der Wandler optimal ausgenutzt und damit bestmögliche Präzision der gewandelten Signale über alle möglichen Eingangsspannungsbereiche erlangt. Die Offset-Spannung kann vom DSP aus mittels eines **D**igital to **A**nalog **C**onverter (DAC)s mit einer Genauigkeit von 12 Bit im Bereich ±12 V eingestellt werden. Die Konfiguration der Verstärkung erfolgt ebenfalls vom DSP aus über ein digitales Potentiometer, das 256 Verstärkungsfaktoren innerhalb des spezifizierten Bereichs von 1x bis 24x gestattet.

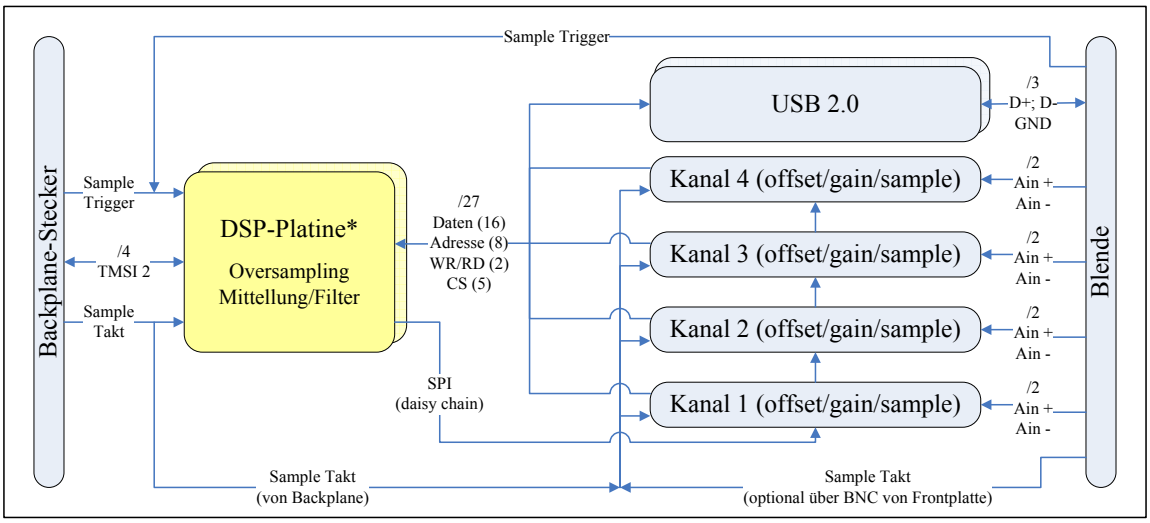


Abbildung 4.11: Schematische Darstellung des ICU-Einschubs „ASAM“: Über vier konfigurierbare Kanäle, können analoge Signal im Bereich von ±12 V mit bis zu 5 MHz digitalisiert werden. Für jeden Kanal kann über Softwarekontrolle individuell ein Offset (±12 V) und eine Verstärkung (1x bis 24x) eingestellt werden. Nach konfigurierbarer Mittelung (minimales Oversampling von fünf) werden die Daten über USB 2.0 zum Anwender PC übertragen.

Abbildung 4.11 veranschaulicht den schematischen Aufbau der ASAM-Platine in einem Blockdiagramm. Angemerkt sei, dass zur Kontrolle der Platine und zur Signalverarbeitung nicht die

gleiche DSP-Plattform wie beim DSC-Einschub oder der ICU-Hauptplatine verwendet wird, sondern ein Nachfolgemodell mit bis zu 3 MB Arbeitsspeicher (SRAM) und aktiviertem parallel Bus (16 Bit-Datenbus; 8 Bit-Adressbus; maximaler Takt: 75 MHz). Die zum Einsatz kommenden Prozessoren stammen allerdings aus der gleichen Bauteilfamilie, so dass die bereits an früherer Stelle beschriebenen Merkmale gleichermaßen zutreffen. Der neu zur Verfügung stehende, parallele Bus wird zum Auslesen der digitalisierten Daten von den DACs und zur Kommunikation mit der USB 2.0 Schnittstelle verwendet. Für den kombinierten Einsatz des ASAMs mit dem DSC-Einschub innerhalb der ICU ist vorgesehen, dass die digitalen Signale des „Sample-Takts“ und des „Sample-Triggers“ über die Backplane geführt werden. Für den autonomen Betrieb des Einschubs besteht optional die Möglichkeit, diese Signale, von Drittgeräten stammend, über BNC-Steckverbinder in der Einschubblende anzuschließen. Alternativ ist die eigenständige Generierung der Signale durch den DSP denkbar. Die Konfiguration der Offsets, der Verstärkungen der einzelnen Analogkanäle, des Oversampling-Verfahrens, sowie der Anzahl von Messwerten pro getriggerten Digitalisierungszyklus erfolgen in einer ersten Ausbaustufe über die platineneigene USB-Schnittstelle vom PC aus. Die vom DSP koordinierten Prozesse der Wandlung, Mittelung und das Übertragen der Messdaten an den PC sind im Flussdiagramm in Abbildung 4.12 dargestellt. Darüber hinaus zeigt Abbildung 4.13 eine Fotographie des ASAM Einschubs (Prototyp).

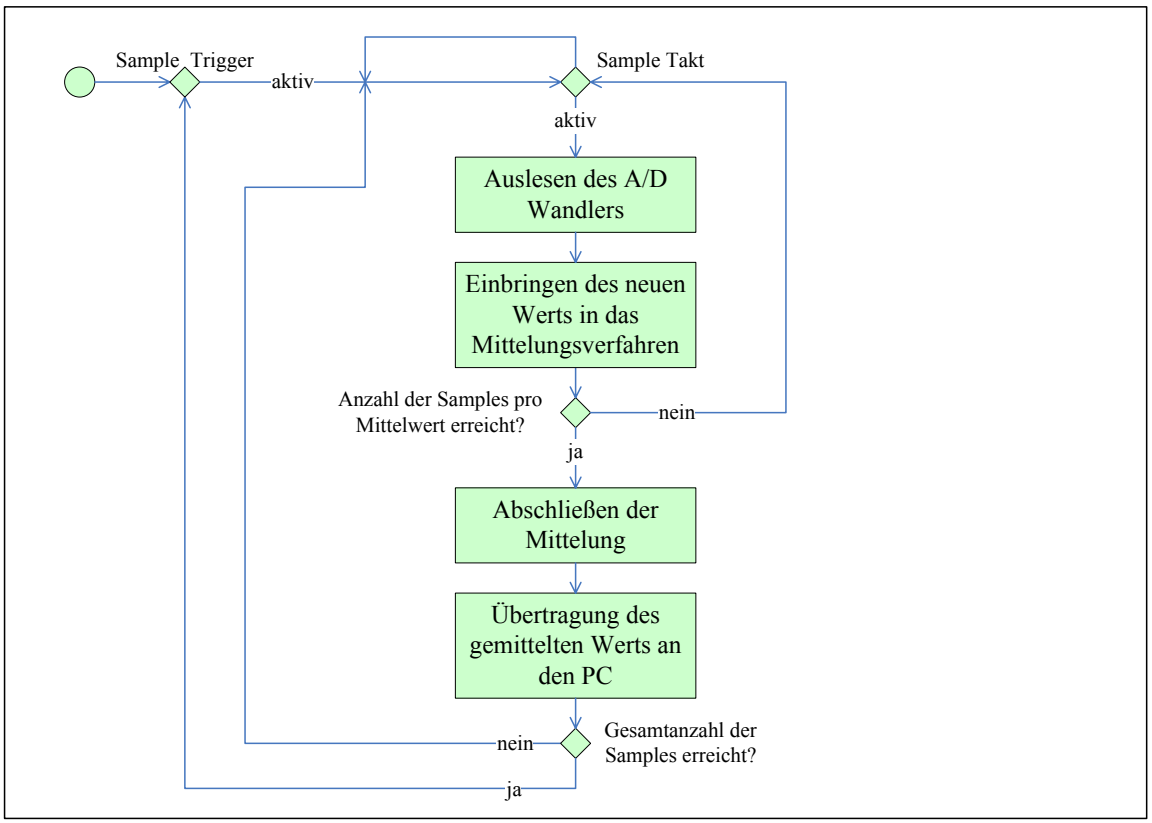


Abbildung 4.12: Flussdiagramm der Prozesse des ASAM DSPs: Datenaufnahme, Verarbeitung und Übertragung an den PC in Abhängigkeit der Signale „Sample Takt“ und „Sample Trigger“

Die Konfiguration der Parameter, welche die Funktionsweise des ASAM-Einschubs beeinflussen, soll mittelfristig ebenfalls durch das, über die Backplane geführte, TMSI erfolgen. Beim Einsatz innerhalb der ICU kann der ASAM-Einschub so über diese interne Schnittstelle

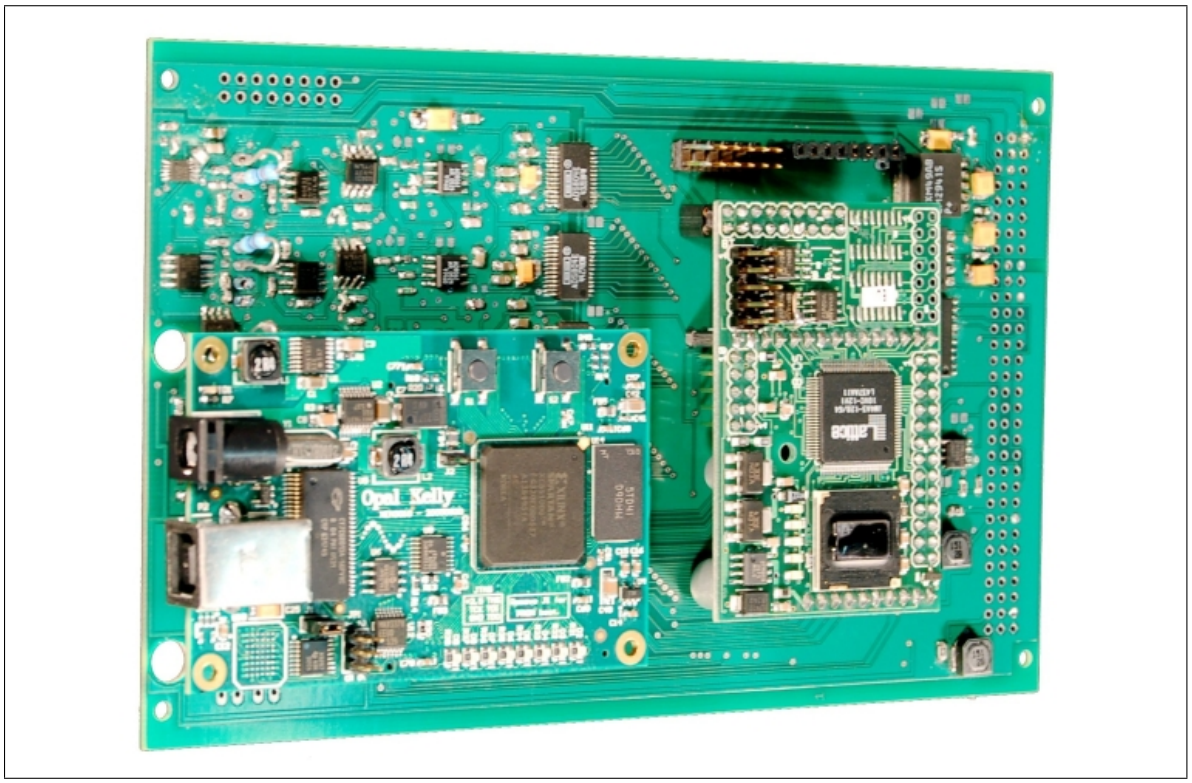


Abbildung 4.13: ICU-Einschub „ASAM“: Fotographie des Prototypen mit aufgesteckter DSP-Platine und aufgestecktem Modul mit Zwischenspeicher und USB 2.0 Schnittstelle.

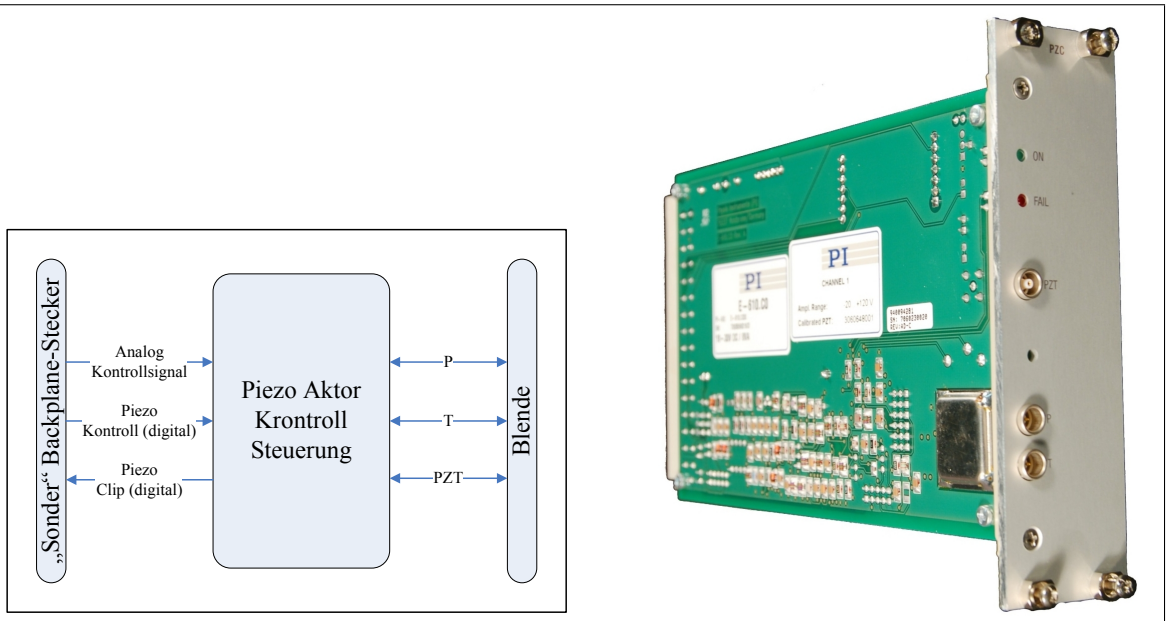
an die Hauptplatine angebunden werden. Wird der ASAM-Einschub in Verbindung mit dem DSC-Einschub (s. Abschnitt 4.6.2) als eigenständiges Gerät eingesetzt, besteht ebenfalls die Möglichkeit der Anbindung der beiden Platinen über das TMSI und damit der Konfiguration durch die Digitale Scanner Kontrollsteuerung.

Diese verschiedenen, vorgestellten Konstellationen der bisher beschriebenen Einschübe verdeutlichen den strikt modularen Charakter der entstandenen Hardware. Durch Ausnutzen des TMSIs als gemeinsame Schnittstelle und durch die Berücksichtigung der Multi-Master-Fähigkeit der einzelnen Einschübe ist es gelungen, eine nahezu beliebige Kombination der Einschübe miteinander zu ermöglichen.

4.6.3 Sondereinschub: Piezo-Control

Wie in Abschnitt 4.5 beschrieben, besteht die Möglichkeit über eine Backplane-Erweiterung Sonder-Einschübe mit arbiträren Pinbelegungen bzw. nicht zum ICU-Backplane-Stecker kompatiblen Steckertypen in die ICU zu integrieren. Dies wird im Falle des von **P**hysik **I**nstrumente bezogenen Piezo-Treibers[7] genutzt. Der Piezo-Treibers eignet sich aufgrund seiner Bauform auf Basis einer Euro-Platine und seines vorderen Abschluss durch eine Aluminiumblende mit integrieren Steckverbindern und Status-LEDs prinzipiell als ICU-Einschub. Durch den Typen des Backplane-Steckers und dessen Pinbelegung entspricht der Piezo-Treiber allerdings nicht den Konventionen für Nebenplatinen und fällt daher unter die Rubrik *Sondereinschub* (vgl. Abschnitt 4.5).

[7] Das von Physik Instrumente GmbH & Co. KG, Karlsruhe, Deutschland bezogene Modell des Verstärker-/Servocontrollermodul hat die Bezeichnung E-610.C0/E-802.55.



(a) schematischer Aufbau (b) Fotographie

Abbildung 4.14: ICU-Einschub „Piezo-Control“: für geregelte Steuerung eines Piezo-Aktors. Im Fall des Mikroskops iMIC wird die Feineinstellung der Fokusposition über einen Piezo-Aktor erlangt.

Die Ansteuerung des Piezo-Kontroll Einschubs erfolgt über ein analoges Steuer-Signal von 0 bis 10 V, was einem maximalen Hub des Piezo Aktors[8] von 250 µm entspricht. Die Genauigkeit der Positionierung liegt bei der Verwendung eines 16 Bit Analog-Signals theoretisch bei 3,8 nm. Durch die anspruchsvolle, mechanische Kombination eines Schrittmotors und des Piezo-Aktors zu einem zweistufigen z-Trieb für Grob- und Feineinstellung des Fokus, ist eine Positionierung des Gesamtsystems allerdings lediglich bis auf eine Genauigkeit von 50 nm spezifiziert. Über eine digitale Steuerleitung „Piezo-CTRL“ kann zwischen dem geregelten und dem ungeregelten Betrieb des Verstärker-/Servocontrollermoduls umgeschaltet werden. Da die Regelung des Piezos für schnellstmögliche Positionierung sehr aggressiv eingestellt ist, können Veränderungen der Belastung wie z. B. Gewichtsveränderungen bei einem Objektivwechsel zu Eigenschwingungen des Piezos führen. Um dies zu verhindern, muss z. B. vor dem Anfahren der Null-Position des z-Triebs und dem damit verbundenen Absetzen des Objektivs, auf ungeregelten Betrieb umgeschaltet werden. Eine zusätzliche digitale Statusleitung „Piezo-Clip“ zeigt an, ob die angelegte Steuerspannung im gültigen Bereich liegt.

4.6.4 Schnittstellen-Steckverbinder für das Basis-System (ICU, iMIC, Polychrome V)

In den vorangegangenen Abschnitten wurden die ICU-Einschübe und die dem Benutzer über Steckverbinder in den Blenden der ICU-Vorderseite zugänglichen Schnittstellen vorgestellt. Die Vorderansicht der ICU bei voller Bestückung ist in Abbildung 4.15 dargestellt. Wie in Abschnitt 4.2 angesprochen und in den Vorgaben und den Anforderungen an die Hardware gefordert, sollten für die Standardausstattung bestehend aus, ICU, iMIC, Polychrome V und CCD-Kamera die für die einzelnen Geräte notwendigen Leitungen in Steckverbindern zusam-

[8] Das von Physik Instrumente GmbH & Co. KG, Karlsruhe, Deutschland bezogene Modell des Piezo-Aktors hat die Bezeichnung PIHera P-622.1CL.

mengefasst werden. Daher wurden neben einem Geräteeinbaustecker[9] mit Netzfilterung für die Spannungsversorgung auch Steckverbinder für Schnittstellen zum Anwender-PC (derzeit 9 poliger D-Sub Steckverbinder für EIA-232), zum iMIC und zur Lichtquelle Polychrome V in die Rückseitigen Blenden integriert.

Abbildung 4.15: Ansicht der ICU-Vorderseite mit Hauptplatinen Einschub, Nebenplatinen Einschub „Multi-I/O“, Nebenplatinen Einschub „DSC“ und Sondereinschub „Piezo Control“

Der Steckverbinder zum Anschluss des iMIC (15-poliger D-Sub), bindet sowohl Spannungsversorgung (+24 V, +5 V, GND) als auch Steuersignale in *einen* Stecker ein. Hierdurch kann die im Sockel des iMIC integrierte Elektronik inklusive der Schrittmotorsteuerungen über *eine* Verbindung zwischen der ICU und dem iMIC betrieben werden.

Ein RJ-45-Steckverbinder dient dem Anschluss eines Polychrome V an die ICU und bietet neben einer analogen Steuerleitung und einer digitalen Triggerleitung auch optionalen Leitungen für eine EIA-232-Schnittstelle. Das Einstellen der Wellenlänge des zum Anregen von Fluoreszenz gewünschten Lichtes, kann somit auf drei verschiedene Weisen erfolgen:

- *Analog:* Einstellung erfolgt über die analoge Steuerleitung. Die Triggerleitung ist in diesem Fall mit der Funktion eines Verschlusses (Shutter) verknüpft.
- *Digital mit Trigger:* Die Konfiguration der Wellenlänge oder ganzer Beleuchtungsprotokolle erfolgt über die EIA-232-Schnittstelle. Mittels der Triggerleitung wird der Wechsel der Einzel-Wellenlängen oder der Ablauf von Belichtungsprotokollen gesteuert.

[9] Der Geräteeinbaustecker ist für den Anschluss von Kaltgerätesteckern nach DIN VDE 0625 Teil 1 Normblatt C14 vorgesehen.

- *Nur Trigger:* Die Konfiguration erfolgt wie im vorherigen Modus mittels der EIA-232-Schnittstelle, allerdings nicht über die ICU sondern z. B. direkt über den Anwender-PC. Der Wechsel der Einzel-Wellenlängen oder der Ablauf von Beleuchtungsprotokollen wird allerdings weiterhin durch die ICU über die Triggerleitung bestimmt.

Die Rückansicht der ICU ist in Abbildung 4.16 dargestellt.

Abbildung 4.16: Rückansicht der ICU mit Lüftungsgitter und Steckverbindern zum Anschluss der Versorgungsspannung, der Verbindung zum Benutzer-PC, des Polychrome V und des iMIC.

5 Flexible Echtzeit-Prozesssteuerung mittels modularer Firmware

Gemäß dem in Kapitel 3 vorgestellten Konzept zur Steuerung von integrativen und modularen Systemen soll die Echtzeitkoordination von zeitkritischen Abläufen hardwareseitig über eine Kontrollplattform geschehen. Die daraufhin entworfene Hardware der **I**ntegration **C**ontrol **U**nit (ICU), ihre Komponenten und deren Verknüpfung sowie die zentrale DSP-basierte Recheneinheit wurden im vorangegangen Kapitel eingehend beschrieben. Obwohl die Verteilung der Daten- und Steuerungssignale vom DSP zu den einzelnen Schnittstellenbausteinen zu großen Teilen auf die programmierbare Logik des CPLDs ausgelagert wurde (vgl. Abschnitte 4.3 und 4.4), bietet erst eine entsprechend programmierte DSP-Firmware die geforderte flexible Funktionalität der ICU. Daher bezieht sich der überwiegende Teil der hier vorgestellten Arbeit auf die Ausarbeitung von Firmwarekonzepten und deren Umsetzung nach den vorgegebenen Anforderungen. Bei der Umsetzung der Firmwarekonzepte wurde stets auf Codeeffizienz im Hinblick auf Speicherbedarf und Verarbeitungsgeschwindigkeit optimiert. Für ein möglichst hohes Maß an Portabilität wurde bei der Implementierung zudem auf die Trennung zwischen hardwarespezifischem und hardwareunspezifischem Programmcode geachtet. Die Programmierung der Firmware erfolgte überwiegend in der Programmiersprache C [83] (im Folgenden *C*) und wurde mittels eines optimierenden Compilers in die Maschinensprache des verwendeten DSPs übersetzt. In Ausnahmen wurde unmittelbar ASSEMBLERCODE erstellt, um elementare und zeitkritische Funktionen zu optimieren.

Die erarbeiteten Konzepte und die daraus entstandenen Firmwaremodule und deren Verknüpfung werden im Folgenden in diesem Kapitel vorgestellt.

5.1 Ausgangspunkt und Anforderungen

Als Basishardware wird die bereits vorgestellte DSP-Platine der Firma SmartMove verwendet. Ursprünglich als Rechenmodul für digitale Regler entwickelt, wird die DSP-Platine mittlerweile auch im Polychrome V, der Digitalen Scan Kontrollsteuerung DSC und in der ICU eingesetzt. Weitere Projekte, welche ebenfalls die DSP-Platine verwenden werden, sind derzeit in Bearbeitung (Regler für Linearmotoren, ASAM) bzw. in Planung. In enger Zusammenarbeit zwischen SmartMove, BIZ und TILL Photonics wurde bzw. wird im Rahmen zweier durch das Bundesministerium für Bildung und Forschung (BMBF) und dem Freistaat Bayern geförderter Projekte[1] u. a. an einem rudimentären Betriebssystem gearbeitet, das nun als Ausgangspunkt für alle Firmwareprojekte dient: **S**mall **E**mbedded **R**eal-Time **O**perating **S**ystem (SEROS). Auf diesem Betriebssystem aufsetzend besteht die Firmware der verschiedenen Projekte aus einem gemeinsamen Teil SEROS und aus einer projektspezifischen Anwendung, welche die gewünschte Funktionalität liefert.

5.1.1 SEROS - Small Embedded Real-Time Operating System

Der Funktionsumfang von SEROS (s. Abbildung 5.1) beinhaltet in erster Linie eine rudimentäre DRIVER-Struktur und die darauf aufbauenden Driver für die einzelnen Hardwareschnittstellen des DSPs. Die Driver liefern einen zeitoptimierten C-seitigen Zugang (ein so genann-

[1] Details zu den geförderten Projekten befinden sich im Abschnitt: Danksagungen auf Seite iii.

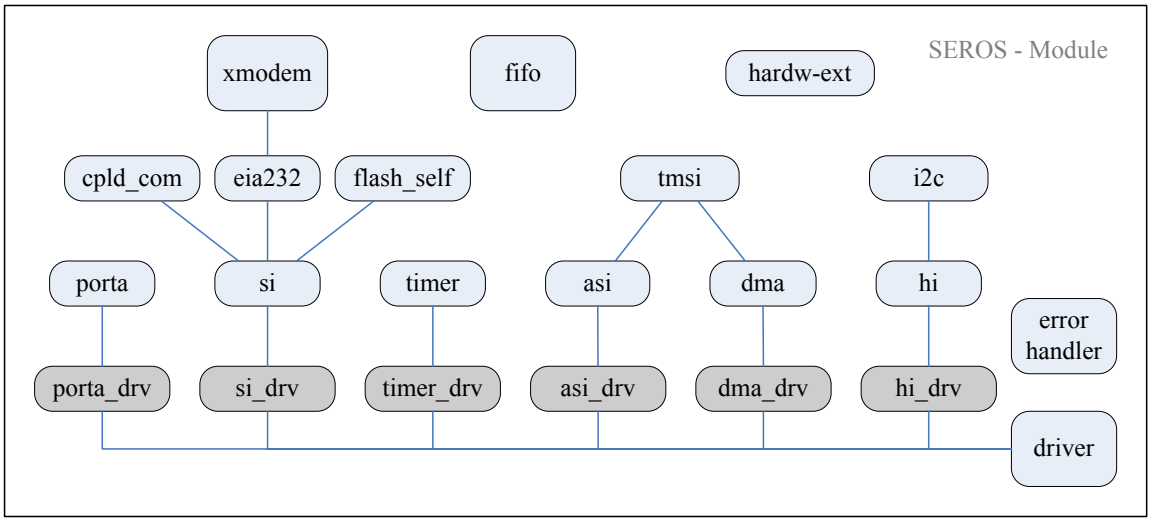


Abbildung 5.1: SEROS Firmwaremodule (vereinfachte Darstellung): rudimentäre Driver für DSP-eigene Hardwareschnittstellen, darauf aufbauende Standardschnittstellen (EIA-232, XModem, I^2C) und DSP-Platinen spezifische Schnittstellen (TMSI, CPLD-Kommandoschnittstelle, Flash-File System), Error-Handler, FIFO und verschiedene hardwarespezifische Systemfunktionen. Blau hinterlegte Modulblöcke können aus Anwendungen heraus benutzt werden. Grau hinterlegte Blöcke kennzeichnen SEROS interne Module.

tes **A**pplication **P**rogram **I**nterface (API)) zur Hardware und koordinieren die Verwendung der Schnittstellen. Innerhalb einer Firmwareanwendung kann eine einzelne Schnittstelle somit mehrfach verwendet werden, wobei durch den Driver ein fehler- bzw. überschneidungsfreies Zugreifen von verschiedenen Teilen der Anwendung garantiert wird. Bereits innerhalb von SEROS wird dies z. B. bei der Mehrfachverwendung der seriellen Schnittstelle (Modul *si*) für die Kommunikation zum CPLD der DSP-Platine, für deren Konfigurierung (Modul *cpld_com*), für die DSP-eigene EIA-232-Schnittstelle mit ggf. aufgesetztem XMODEM Protokoll (Module *eia232* und *xmodem*) und für den Zugriff auf den Flash-Speicher (Modul *flash_self*) eingesetzt (vgl. Abbildung 5.1).

Neben der Driver-Struktur und den Schnittstellen-Drivern bietet SEROS weitere Module mit allgemein nützlichem Funktionsumfang, wie z. B. das Modul *fifo* für einfaches Implementieren von Software-FIFOs, einer Sammlung nützlicher, hardwareabhängiger Funktionen (Modul *hardw_ext*) und eines Error Handlers (Modul *error handler*). Letzterer wird bereits in den SEROS Modulen verwendet, um z. B. Fehler aufgrund von unerlaubten Schnittstellenzugriffen zu dokumentieren. Die Fehlerbehandlung und die Art der Ausgabe von auftretenden Fehlern wird innerhalb der jeweiligen Anwendung definiert. Im Fall der ICU wird ein Fehlverhalten zum einen durch Status LEDs der DSP-Platine angezeigt und zusätzlich über eine entsprechende Fehlermeldung an den Anwender-PC weitergeleitet. Eine solche Fehlermeldung beinhaltet neben einer Aussage über die Art des Fehlers auch einen Verweis auf die entsprechende Stelle innerhalb des Firmware QUELLCODES. So kann leicht identifiziert werden, ob es sich um Fehler in der Bedienung oder um Programmierfehler handelt. Im letzteren Fall kann ein Fehler durch die Angabe der Quellcodestelle komfortabel lokalisiert und daraufhin behoben werden.

Von allen SEROS Modulen soll lediglich auf das Modul *tmsi* näher eingegangen werden, da es die Schnittstelle bildet, die den DSP mit allen innerhalb der ICU eingesetzten Schnittstellen-

bausteinen verbindet. Die hardwareseitige Funktionsweise des TMSI wurde in den Abschnitten 4.3 und 4.4 bereits ausführlich vorgestellt. Da dieses Modul einen entscheidenden Einfluss auf die Leistungsfähigkeit der gesamten ICU Firmware hat, wurde es speziell hierfür entsprechend ausgebaut und optimiert. Das TMSI basiert auf der DSP-eigenen Schnittstelle **A**dvanced **S**erial **I**nterface (ASI), die in einem speziellen Modus als zeitlich gemultiplexte, synchrone, serielle Schnittstelle konfiguriert wird. Wie bereits beschrieben, werden in diesem Modus Daten bzw. Kommandos innerhalb eines konstanten Zeitraums (Frame) zwischen dem DSP und allen angeschlossenen Bausteinen über getrennte Leitungen (MOSI und MISO) bidirektional ausgetauscht (vgl. Abbildung 4.2). Firmwareseitig wird diese Kommunikation auf zwei Ebenen vollzogen: Das Modul *asi* (s. Abbildung 5.1) liefert zum einen über den Driver den Zugang zur eigentlichen Schnittstelle und ist zum anderen für das Versenden und Empfangen der Datenpakete innerhalb eines Frames zuständig. Hierbei muss gegen Ende eines jeden Time-Slots (vgl. Abschnitt 4.3) der nächste zu sendende Wert an die Schnittstelle übergeben und der soeben empfangene Wert von der Schnittstelle ausgelesen werden.

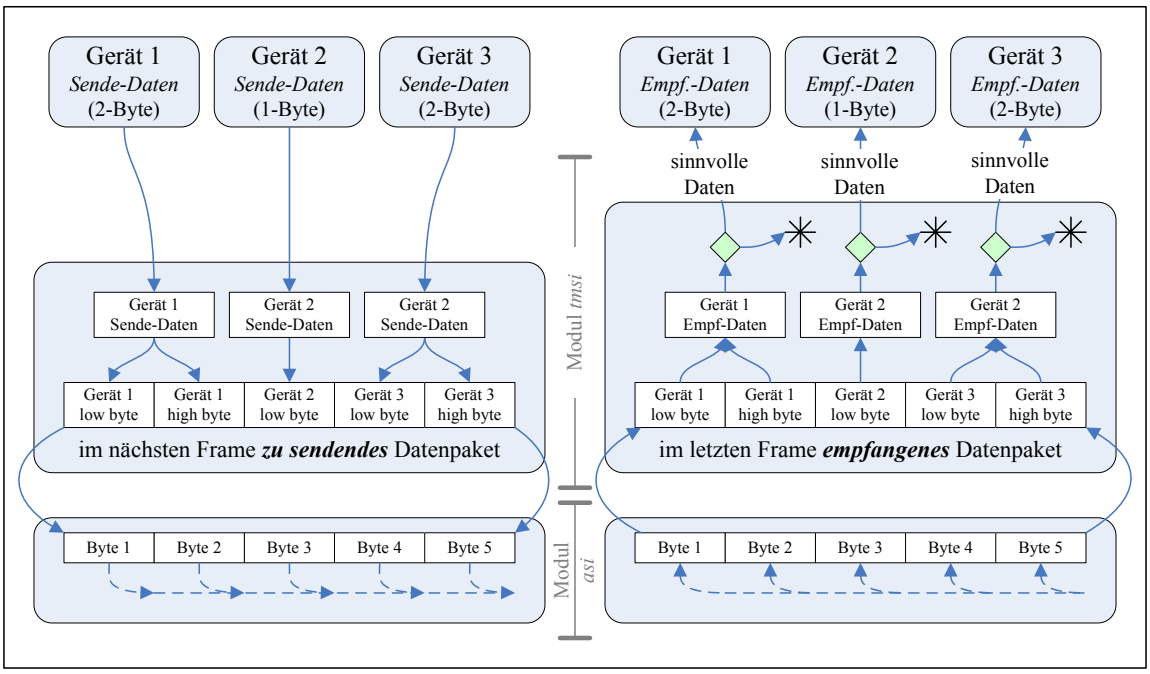


Abbildung 5.2: Aufbau und Funktionsweise der TMSI: Auf der Ebene des Moduls *asi* werden die Daten jeweils über eine DMA empfangen bzw. gesendet. Die Interrupt-Routine des Moduls *tmsi* konfektioniert das Paket der jeweils im nächsten Frame zu senden Daten und verarbeitet die im letzten Frame Empfangen Daten. Der Austausch der Pakete zwischen den beiden Modulen geschieht innerhalb der Interrupt-Routine am Ende eines jeden Frames.

Das übergeordnete Modul *tmsi* konfektioniert zum einen das jeweils im nächsten Frame zu versendende Datenpaket und bearbeitet zum anderen das Paket derjenigen Daten, die im jeweils letzten Frame empfangen wurden. Beim Konfektionieren des Sende-Pakets werden den einzelnen Bausteinmodulen zunächst die Daten entnommen, welche an die angeschlossenen Bausteine übertragen werden sollen. Anschließend werden diese Daten in das Sende-Paket eingeordnet. Die Position der jeweiligen Daten innerhalb des Pakets entspricht dabei dem/den für die Übertragung zugeordneten Time-Slot(s). Bei der Bearbeitung des zuletzt empfangenen Pakets werden zunächst die in den einzelnen Time-Slots empfangen Daten (jeweils ein Byte) ent-

sprechend der baussteinspezifischen Wortbreiten (Vielfache eines Bytes) zusammengesetzt. Im Anschluss werden diese Daten auf sinnvollen Inhalt überprüft und entweder verworfen oder an die Module der jeweiligen Bausteine übergeben. Am Ende eines jeden Frames wird das zu sendende Paket an das Modul *asi* übergeben und das soeben empfangene Paket abgeholt. Abbildung 5.2 veranschaulicht diese Vorgänge in einem Blockdiagramm.

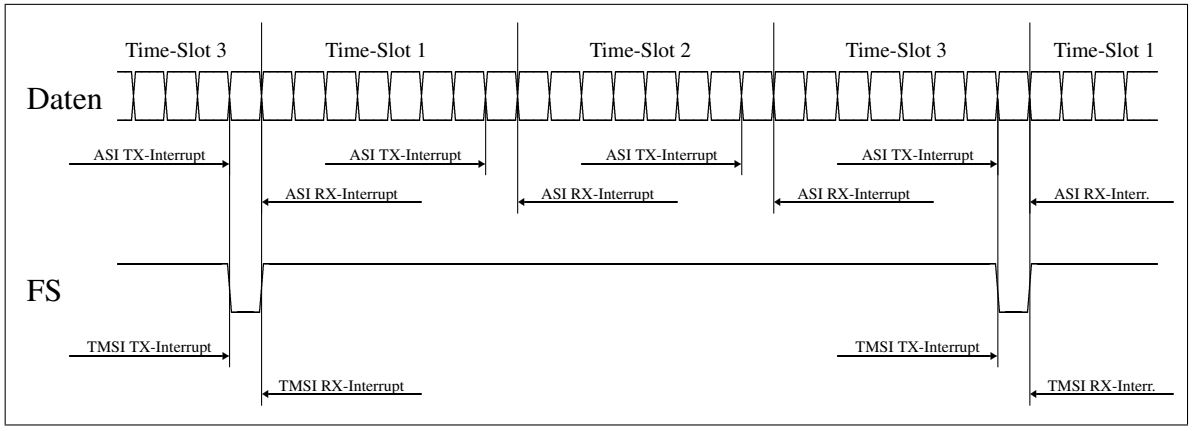


Abbildung 5.3: Vereinfachtes Zeitverlaufsdiagramm der TMSI inklusive Markierungen der Interrupts zur Koordination der Sende- und Empfangsvorgänge auf den beiden Ebenen *asi* und *tmsi*. TX-Interrupts leiten jeweils die Senderoutinen ein, RX-Interrupts starten die jeweiligen Empfangsroutinen.

Das Verfahren des Sendens und Empfangens sowohl auf der Ebene des Moduls *asi* als auch auf der Ebene des Moduls *tmsi* wurde in einer ersten Umsetzung dieses Konzepts jeweils über Interrupts gesteuert. Abbildung 5.3 veranschaulicht den zeitlichen Ablauf eines exemplarischen Frames mit drei Time-Slots und verdeutlicht die Zeitpunkte, an denen die einzelnen Teilfunktionen des Sendens und Empfangens - über Interrupts initiiert - ausgeführt werden. Aus dem Zeitverlaufsdiagramm ist ersichtlich, dass innerhalb eines Frames mit n Time-Slots auf der Ebene ASI $2 * n$ Interrupts und auf der Ebene TMSI 2 Interrupts ausgelöst werden, die jeweils zum Aufruf entsprechender Interrupt Routinen führen. Die Gesamtanzahl der Interrupt-Aufrufe beläuft sich somit auf $2 * n + 2 = 2 * (n + 1)$. Zur Optimierung des TMSI wurde die Schnittstelle auf der Ebene des Moduls *asi* vom Interrupt-basierten Betrieb auf die Verwendung jeweils einer DMA (**D**IRECT **M**EMORY **A**CCESS) für den Sende- und Empfangsteil umgestellt. Durch zusätzliche Optimierung konnte die Abhandlung des Sende- und des Empfangsteils auf TMSI-Ebene in einer Routine zusammengefasst werden. Damit entfallen $2 * n + 1$ Aufrufe von Interruptroutinen pro Frame, was zur Freisetzung der ansonsten hierfür notwendigen Rechenzeit des DSP Hauptprozessors führte. In der derzeitigen Implementierung wird somit die gesamte Schnittstelle vom DSP zur restlichen ICU Hardware über eine einzige Interrupt-Routine bedient.

5.1.2 Boot-Loader

Basierend auf dem Betriebssystem SEROS wurde im Rahmen der oben erwähnten Gemeinschaftsprojekte als erste Anwendung ein BOOT-LOADER programmiert. Neben der Hauptaufgabe, nämlich dem Laden der eigentlichen Anwendung aus dem Flashspeicher in den Arbeitsspeicher des DSPs, ermöglicht der Boot-Loader zusätzlich die Aktualisierung der Anwendungs-Firmware über die EIA232-Verbindung zu einem PC. Verwendet wird hierfür das auf der EIA-232-Schnittstelle aufsetzende XMODEM PROTOKOLL (s. Modul *xmodem* in Abbildung 5.1),

das von allen gängigen TERMINALPROGRAMMEN wie z. B. dem Microsoft® „HyperTerminal“ unterstützt wird. Ein Aktualisieren der Firmware ist bei einer bestehenden EIA-232 Verbindung zwischen der ICU und einem PC somit ohne weitere Utensilien, wie z. B. speziellen Programmierkabeln, möglich und kann mit entsprechender Anleitung sogar von Laien durchgeführt werden.

Die DSP-Platine mit dem Betriebssystem SEROS und dessen Boot-Loader stellt somit ein komplettes, schlüsselfertiges System dar, das als Ausgangspunkt für Firmwareprojekte wie dem der ICU genutzt werden kann.

5.1.3 Anforderungen an die ICU-Firmware

Viele der in Abschnitt 4.1 beschriebenen Anforderungen an die Hardware können in ähnlicher Weise im Bezug auf die Firmware der ICU übernommen werden, da die Gestaltung der Hardware selbstverständlich einen unmittelbaren Einfluss auf die dafür konzipierte Firmware hat. So muss die Firmware den strikt modularen Aufbau der Hardware berücksichtigen sowie eine einfache Einbindung neuer Schnittstellenbausteine und deren Verwendung innerhalb der Ablaufsteuerung ermöglichen. Das geforderte Echtzeitverhalten mit einem angestrebten Zeitraster von 10 µs ist durch die Eigenschaften der Hardwareschnittstelle des DSPs prinzipiell realisierbar und gilt daher ebenfalls als Richtlinie für die Firmware. Ob, bedingt durch die weiteren Anforderungen bezüglich der Firmwareeigenschaften, dieses hochgesteckte Ziel von 10 µs erreicht werden kann, ist anhand von ersten Abschätzungen allerdings fraglich. Weitere Anforderungen orientieren sich an dem in Kapitel 3 vorgestellten Konzept und der daraus abgeleiteten Funktionalität der ICU-Firmware: Alle innerhalb der ICU zur Verfügung stehenden Schnittstellenbausteine sollen einerseits durch unmittelbaren Zugriff und andererseits aus einem Ablaufprotokoll heraus verwendbar sein. Bei letzterem soll der Ausführzeitpunkt eines Kommandos entweder anhand von im Protokoll vorgegebenen Zeitpunkten (zeitlich deterministisch), in Abhängigkeit von externen Einflüssen (z. B. Triggersignale von weiteren Systemkomponenten) oder als Reaktion auf firmwareinterne Ereignisse bzw. Zustände bestimmt werden können. Protokollgesteuerte Abläufe sollen zudem durch die Möglichkeit der Verwendung von Elementen, die den Ablauf beeinflussen, möglichst flexibel gestaltbar sein. In Anlehnung an imperative Programmiersprachen wurde daher angedacht, Kontrollstrukturen wie Schleifenkonstrukte oder von Bedingung abhängige Verzweigungen sowie die Verwendung von ICU-internen Variablen und die Anwendung mathematischer Operationen innerhalb der Protokolle zu ermöglichen. Soweit möglich sollten die verschiedenen Firmwaremodule unabhängig von der Hardware der ICU programmiert werden, damit sie ggf. in weiteren Projekten bzw. in Nachfolgeprojekten wiederverwendet werden können. Generell gilt hierbei allerdings, dass eine Optimierung der Firmware gleichzeitig immer eine Abhängigkeit von der jeweils eingesetzten Hardware bedeutet. Die folgende Liste fasst die Anforderungen an die Firmware der ICU zusammen:

- Strikt modularer Aufbau,
- Echtzeitverhalten,
- Bereitstellung der Funktionen aller Schnittstellenbausteine,
- Unterstützung einfacher Einbindung neuer Hardwarekomponenten,
- unmittelbare Ausführung von Kommandos,

- zeitlich deterministische sowie zustandsabhängige, protokollgesteuerte Ausführung von Kommandos,
- ablaufbeeinflussende Elemente (Schleifen, Verzweigungen, Variablen, mathematische Operationen),
- möglichst hardwareunabhängige Programmierung.

5.2 Schnittstelle zur Hardware

Auf der Basis des in Abschnitt 5.1.1 beschriebenen Moduls *tmsi* wurde ein Mechanismus zur Verwaltung der ICU Hardwarekomponenten aufgesetzt, der die flexible Erweiterung bzw. den Austausch von Komponenten ermöglicht. Zudem wurden Module mit generischen C-Schnittstellen für alle derzeit in der ICU eingesetzten Schnittstellenbausteine geschaffen. Abbildung 5.4 zeigt diese verschiedenen Module und deren Verknüpfungen.

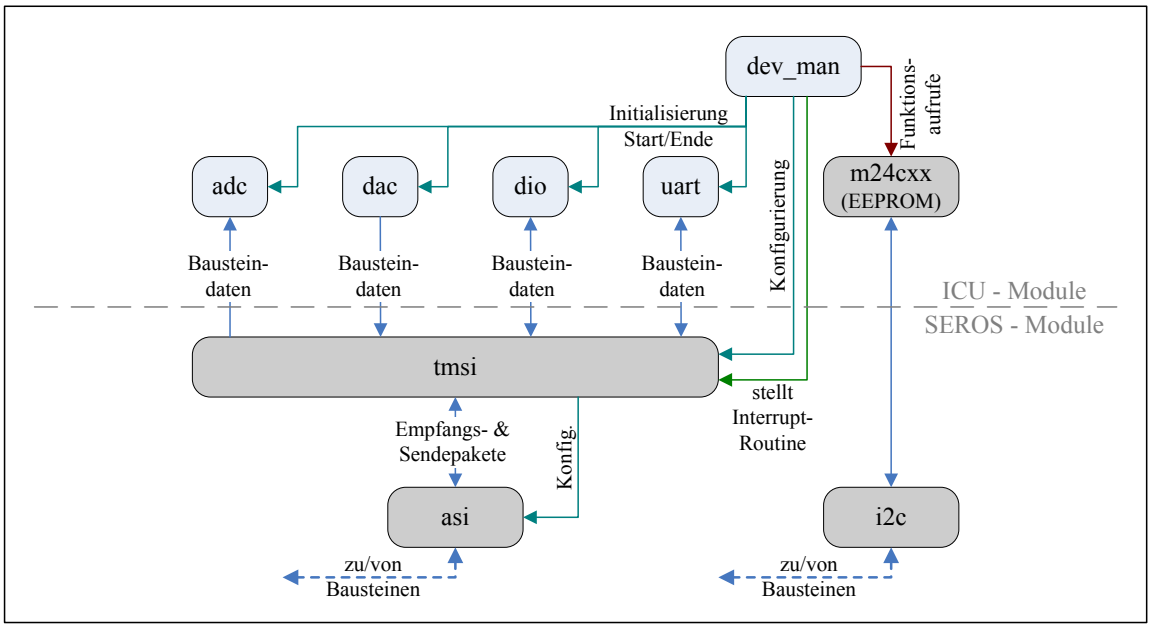


Abbildung 5.4: ICU-Firmware Schnittstelle zur Hardware: Module und deren Verknüpfung (vereinfachte Darstellung). Blau hinterlegte Modulblöcke bieten zusätzliche Schnittstellen, die im Weiteren in anderen ICU Modulen noch verwendet werden. Grau hinterlegte Blöcke kennzeichnen vollständig beschriebene Module.

Ein „Geräte Manager“ (Modul *dev_man*) überprüft nach dem Laden und Starten der ICU-Anwendung zunächst die jeweils aktuelle Hardwarekonfiguration: Hierfür befinden sich Speicherbausteine (EEPROM) auf allen ICU-Einschüben und Aufsteckplatinen, die über einen I^2C-Bus mit dem DSP verbunden sind. Auf den Speicherbausteinen sind Informationen über die Art der jeweiligen Platine, deren Version und Bestückung abgelegt. Nach Auslesen und Auswerten dieser Informationen (Modul *m24cxx*) initialisiert der Geräte Manager die Module der detektierten Schnittstellenbausteine (Module *adc*, *dac*, etc.). Zusätzlich stellt das Modul *dev_man* die Interrupt Routine für die Abhandlung des Sende- und Empfangsteils der TMSI (vgl. Abschnitt 5.1.1) zur Verfügung und sorgt für eine der Hardwarekonstellation entsprechenden Konfigurierung der TMSI. Die derzeit durch die ICU-Hardware zur Verfügung stehenden Schnittstellenbausteine werden entsprechend dem Flussdiagramm in Abbildung 5.5 eingebunden. Die aktuelle Nutzung der TMSI schöpft noch nicht die komplette Sende- und Empfangsbandbreite

aus. Die Anzahl der Time-Slots kann bis auf eine maximale Anzahl von 32 erhöht werden. Die flexible Konfigurierung der TMSI ermöglicht zudem, dass der Sende- und der Empfangskanal eines Time-Slots zwei unterschiedlichen Geräten bzw. Schnittstellenbausteinen zugeordnet werden kann. Da z. B. die beiden Bausteine für analoge Ausgänge lediglich den Sendekanal benötigen, könnte der Empfangskanal somit für die Übertragung von Daten zwischen weiteren Bausteinen und dem DSP genutzt werden.

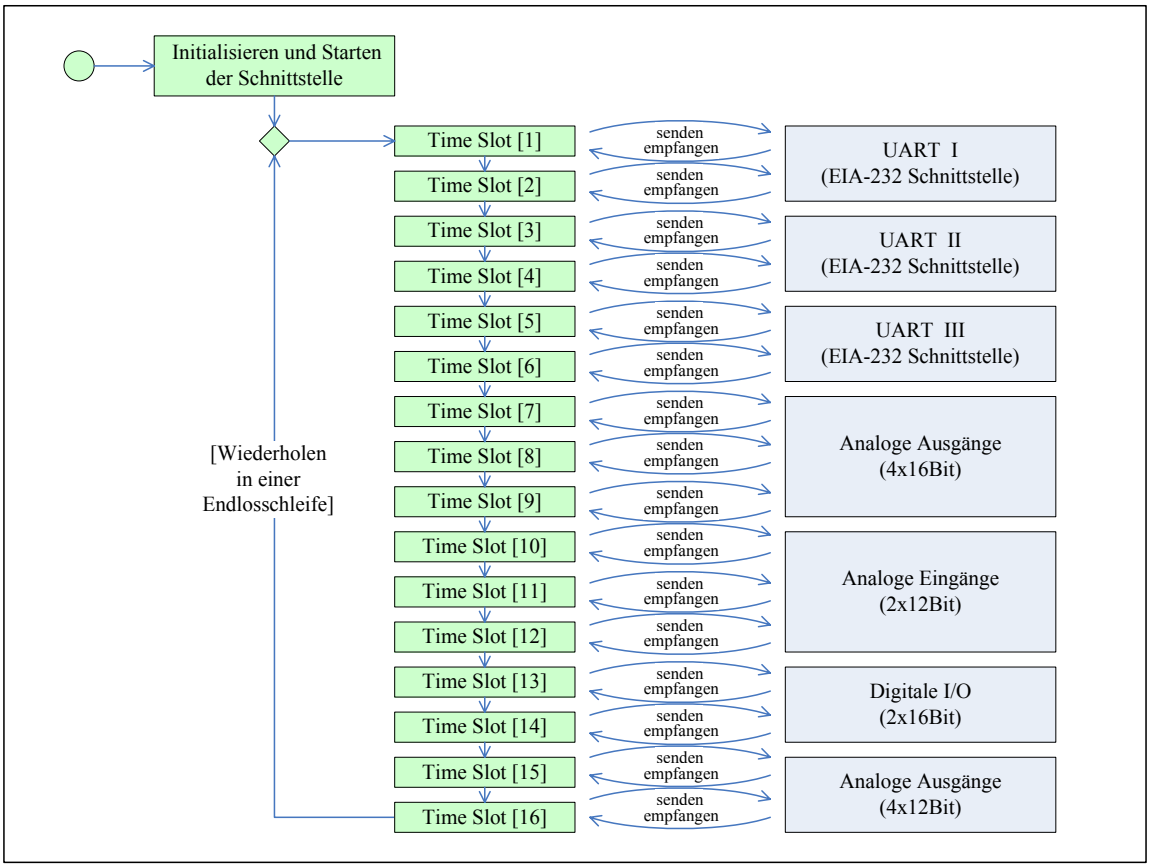


Abbildung 5.5: Flussdiagramm der ICU TMSI mit Zuordnung der Time-Slots zu den entsprechenden Schnittstellenbausteinen.

5.2.1 Schnittstellenmodule

Die derzeit über den Geräte Manager verwalteten Schnittstellenbausteine können aus der ICU-Firmware bequem über ein C-API (APPLICATION PROGRAM INTERFACE) angesprochen und/oder ausgelesen werden. Die Module *adc*, *dac*, *dio* und *uart* enthalten hierfür jeweils Funktionen zur Initialisierung, zum Starten und Beenden der Verbindung zur Hardware. Zusätzlich stehen je nach Funktionalität des Moduls generische Funktionen zum Senden von Daten oder Kommandos sowie zum Auslesen der Schnittstellenbausteine zur Verfügung. Teile der Firmware, die auf diesen Modulen aufsetzen, sind somit unabhängig von der verwendeten Hardwarearchitektur.

So bietet das Modul *adc* eine C-Funktion, mittels derer unter Angabe der Nummer des auszulesenden, analogen Eingangs der Wert des somit spezifizierten Wandlers an die aufrufende Funktion zurückgegeben wird. Das Modul *dac* bietet Funktionen zum absoluten und zum inkrementellen Setzen eines analogen Ausgangs unter Angabe der entsprechenden Ausgang-Nummer und des gewünschten absoluten bzw. relativen Wertes. Sind mehrere Ausgänge physikalisch

über einen Chip (Mehrkanalwandler) realisiert, kann pro TMSI-Zyklus jeweils nur ein Ausgang gesetzt werden. Um ein quasi gleichzeitiges Aktualisieren mehrerer analogen Ausgänge zu ermöglichen wird eine FIFO-Struktur verwendet, die alle zu setzenden Werte so schnell wie möglich an den Mehrkanalwandler übergibt. Die Dauer bis zum Aktualisieren des zuletzt gesetzten Kanals entspricht damit dem Produkt aus der Zykluszeit und der Anzahl der zu setzenden analogen Ausgänge. Zudem kann über eine weitere Funktion der aktuelle Wert eines über die Ausgang-Nummer spezifizierten analogen Ausgangs abgefragt werden. Nach einem Neustart der PC-Software oder nach einer unbekannten Anzahl von inkrementellen Änderungen des zu wandelnden Wertes kann hierüber der jeweils aktuelle Wert ermittelt werden.

Die digitalen I/O der ICU sind in Bänken zu jeweils 8 Bit organisiert. Das Modul *dio* enthält C-Funktionen zum Setzen bzw. zum Auslesen von digitalen I/O unter Angabe der Bank und einer Bit-Maske, die mit 8 Bit alle zu setzenden bzw. auszulesenden I/O markiert. Die Funktion zum Setzen der digitalen Ausgänge benötigt zusätzlich eine Bit-Maske, welche die logischen Zustände der zu setzenden Ausgänge definiert. Eine solche Bit-Maske wird als Ergebnis der Funktion zum Auslesen der digitalen Eingänge an die aufrufende Funktion zurückgeliefert.

Die ICU-eigenen seriellen EIA-Schnittstellen können über Funktionen des Moduls *uart* betrieben werden. Zum Senden und Auslesen stehen jeweils Funktionen zur Verfügung, die unter Angabe der Schnittstellennummer entweder ein einzelnes Zeichen oder einen gesamten Block senden bzw. auslesen. Jede Schnittstelle kann über eine weitere Funktion unter Angabe der Schnittstellennummer individuell konfiguriert werden. Neben den generellen Schnittstellenparametern (Baudrate, Wortlänge, Anzahl der Stoppbits und Verwendung der Paritätsüberprüfung) kann zusätzlich ein Lese-Modus ausgewählt werden, der das Verfahren zur Behandlung der von angeschlossenen Endgeräten empfangenen Daten festlegt: Im Modus „deaktiviert“ werden alle ankommenden Daten verworfen. Im Modus „abfragen“ werden ankommende Daten in einem Speicher abgelegt. Über aktives Auslesen (s. o.) durch die PC-Software werden diese Daten zum PC übertragen. Im Modus „Markierung des Endes der Übertragung“ wird in der Konfiguration ein weiterer Parameter angegeben, den das an die jeweilige EIA-Schnittstelle angeschlossene Endgerät sendet, um das Ende einer Übertragung zu markieren. Beim Auslesen der Schnittstelle werden alle im Speicher befindlichen Daten bis zu dem angegebenen Zeichen an den PC übermittelt. Der Modus „Anzahl der Zeichen“ funktioniert prinzipiell wie der zuletzt vorgestellte Modus. Das Ende einer Rückmeldung eines Endgeräts wird hier allerdings nicht durch ein bestimmtes Zeichen, sondern durch eine definierte Länge der Meldungen angegeben. Das Verfahren der letzten beiden Modi wurde automatisiert und in Form zweier weiterer Modi zur Verfügung gestellt. Der Unterschied ist hierbei, dass vom Endgerät empfangene Daten automatisch auf das Vollständigkeitskriterium geprüft und ggf. ohne aktive Abfrage an den PC geschickt werden. Bei Verwendung des Modus „Timeout“ werden auf eine Abfrage der Schnittstelle hin nach Ablauf einer in der Konfiguration angegebene Zeitspanne alle im Speicher befindlichen Daten an den PC übertragen. Im letzten Modus wurde dieses Verfahren wiederum automatisiert, indem der Empfang eines Zeichens den Timeout automatisch startet. Nach Auslaufen des Timeouts durch längere Inaktivität der Schnittstelle führt automatisch dazu, dass alle im Empfangsspeicher befindlichen Daten an den PC gesendet werden.

5.3 Protokollbasierte Ausführung

Die Hauptfunktion der ICU ist die protokollbasierte Ausführung von im Vorfeld definierten Abläufen in Echtzeit. Dabei soll das Echtzeitverhalten u. a. dadurch erreicht werden, dass Kommandos, die im Protokoll eingetragen werden, atomare Eigenschaften besitzen. Dies bedeutet einerseits, dass die maximale Ausführdauer der einzelnen Kommandos im Vorfeld eindeutig bestimmbar ist und andererseits, dass die Summe aller Ausführzeiten im Rahmen des angestrebten Zeitrasters liegt.

Um den Einsatz der erarbeiteten Funktionalität innerhalb unterschiedlicher Anwendungen zu ermöglichen, wurde der hardwareunabhängige Teil von Beginn an separiert. Das Ergebnis ist eine Zweiteilung des Protokolls in einen hardwareunabhängigen Teil (Modul *prot*) und in einen Teil (Modul *prot_ext*), der die Protokollkommandos zur Steuerung der Hardware und anwendungsspezifische Protokollfunktionen enthält. Der hardwareunabhängige Teil umfasst die eigentliche Protokoll-Struktur, Funktionen zur Aufzeichnung und Ausführung eines Protokolls sowie generische Protokollkommandos z. B. für die Realisierung von Schleifen und bedingungsabhängigen Verzweigungen.

5.3.1 Hardwareunabhängiger Protokollteil

Die firmwareseitige Umsetzung der Protokollfähigkeit sieht als Basis eine Protokollliste vor, deren Einträge jeweils *ein* Kommando beschreiben und den Ausführzeitpunkt, einen Verweis auf die auszuführende Funktion sowie einen Verweis auf die zu übergebenden Funktionsparameter und deren Anzahl enthalten. Dabei ist die Protokollliste durch einen zyklischen Ringspeicher realisiert, der bei optimaler Ausnutzung des Speicherplatzes ein gleichzeitiges Abarbeiten und Definieren des Protokolls erlaubt (s. Abbildung 5.6).

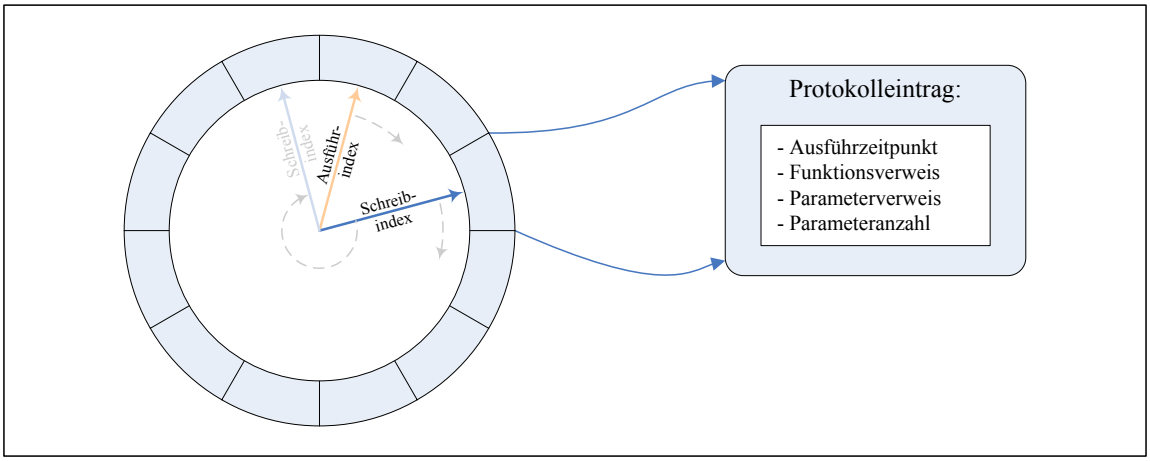


Abbildung 5.6: Protokollliste in Form eines zyklischen Ringspeichers, der gleichzeitiges Schreiben und Abarbeiten von Protokollen ermöglicht. Einträge in die Liste enthalten die jeweils zur Ausführung eines Kommandos notwendigen Informationen.

Nach der Initialisierung oder dem Zurücksetzen des Moduls verweisen sowohl der „Schreib-Index“ als auch der „Ausführindex“ gemeinsam auf eine anfängliche Position innerhalb des Ringspeichers. Beim Hinzufügen und Abarbeiten eines Protokolleintrags erhöht sich der Schreib-Index bzw. der Ausführindex. Neue Einträge können in die Protokollliste übernommen werden, solange der Schreib-Index den Ausführindex nicht „überholt“ (s. Abbildung 5.6 - blass

gezeichneter Pfeil des Schreibindex). Die Abarbeitung der Protokollliste erfolgt nun umgekehrt solange, bis der Ausführindex den Schreib-Index „eingeholt hat“. Bereits abgearbeitete Protokolleinträge können durch Hinzufügen neuer Einträge überschrieben werden, so dass die Kapazität des Ringspeichers zu jedem Zeitpunkt maximal ausgenutzt werden kann.

Neben der soeben beschriebenen Struktur und einer Funktion zum Hinzufügen von neuen Einträgen in die Protokollliste gehört der entwickelte Mechanismus zur Ausführung eines Ablaufprotokolls zum Kern des Moduls *prot*. Der Zeitpunkt der Ausführung eines Protokolleintrags wird abhängig von der weiteren Firmwarearchitektur und den Anforderungen der Anwendung durch unterschiedliche Ereignisse bestimmt. So könnte die Protokollausführung z. B. abhängig von einer Zählervariable, relativ zur Systemzeit oder Interrupt-betrieben stattfinden. In adaptiven oder Multi-Master-Systemen hängt die protokollbasierte Ausführung von Kommandos oder Kommandoblöcken zudem von externen Ereignissen wie z. B. digitalen oder analogen Steuersignalen ab. Da für die Bestimmung des Ausführzeitpunkts kein allgemein gültiges Verfahren festgelegt werden kann, wurde ein Mechanismus erarbeitet, der zur Laufzeit erlaubt das jeweils für den nächsten Protokolleintrag gültige Verfahren festzulegen. Dem entsprechend codierten Parameter „Ausführzeitpunkt“ eines Protokolleintrags (vgl. Abbildung 5.6) wird entnommen, welches Verfahren verwendet werden soll. Damit können Bedingungen für die Ausführung einzelner Kommandos individuell definiert und zur Laufzeit entsprechend angewendet werden. Bezüglich der Codierung des „Ausführzeitpunkts“ hat der Wert 0 eine gesonderte Bedeutung, da er für alle denkbaren Ausführmechanismen lediglich eine sinnvolle Bedeutung hat, nämlich unmittelbare Ausführung ohne zusätzliche Verzögerung. Auf die weitere Codierung dieses Parameters im Falle der ICU wird im folgenden Abschnitt (Abschnitt 5.3.2) detailliert eingegangen. Die Funktion zur Ausführung von Protokollen konnte unter den beschriebenen Voraussetzungen folgendermaßen optimiert werden (s. Abbildung 5.7):

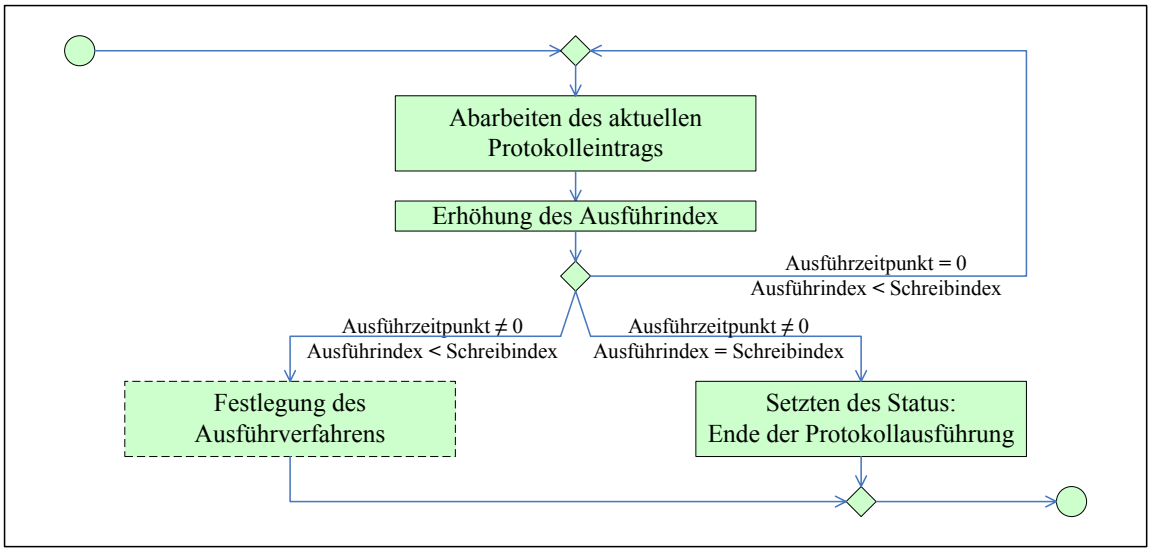


Abbildung 5.7: Vereinfachtes Flussdiagramm der Funktion zum Ausführen eines auf der Protokollstruktur (s. Abbildung 5.6) basierenden Ablaufprotokolls. Blöcke mit gestrichelten Rahmen markieren Aufrufe von modulfremden Funktionen.

Nach Aufruf der Ausführfunktion wird der aktuelle, durch den Ausführindex spezifizierte Protokolleintrag abgearbeitet. Im Anschluss daran wird der Ausführindex so lange erhöht und der jeweils durch den Ausführindex spezifizierte Protokolleintrag ausgeführt, bis der Parame-

ter „Ausführzeitpunkt“ des jeweiligen Eintrags einen von null verschiedenen Wert annimmt. Somit werden alle Kommandos so schnell wie möglich hintereinander ausgeführt, deren Parameter „Ausführzeitpunkt“ durch den Wert null eine unmittelbare Ausführung fordert. Wird beim Durchlaufen dieser Schleife das Ende des definierten Protokolls erreicht (Ausführindex = Schreib-Index), so wird durch entsprechendes Setzen der Statusvariable des Moduls das Ende der Ausführung festgelegt. Ist das Protokollende noch nicht erreicht, enthält der Parameter „Ausführzeitpunkt“ des aktuellen Eintrags aber einen von null verschiedenen Wert, so wird vor Beendigung der Ausführfunktion noch das Verfahren für die weitere Ausführung festgelegt. Hierfür wird eine externe Funktion aufgerufen, die durch die Angabe eines Verweises innerhalb der Initialisierung des Moduls *prot* angegeben werden muss. Über diesen Mechanismus kann die Ablaufkontrolle der Protokollausführung komplett auf einer übergeordneten Ebene von der jeweilige Anwendung angepasst erfolgen.

Die für die Firmware geforderte Flexibilität bezüglich der protokollbasierten Ablaufsteuerung wird u. a. durch Konstrukte wie Schleifen und bedingungsabhängigen Verzweigungen ähnlich denen einer höheren Programmiersprache wie z. B. C ermöglicht. Als idealerweise hardwareunabhängige Funktionalität wurden diese Konstrukte als weiterer Teil in das Modul *prot* übernommen. Sie liefern zugleich die ersten Funktionen, die einer Protokollliste als Einträge hinzugefügt werden können. Wiederum wurde sowohl im Falle von Schleifen als auch im Falle von Verzweigungen besonders auf die Trennung von allgemeinen und von anwendungsspezifischen Programmteilen geachtet. Das Modul stellt daher Funktionen für die Verwendung innerhalb eines Protokolls zur Verfügung, welche lediglich die entsprechende Grundfunktionalität bieten. Die Bedingung für die Anzahl der Schleifendurchläufe bzw. für die verzweigte Ausführung von Protokollblöcken wird beim Eintrag der entsprechenden Funktionen in die Protokollliste als Parameter in Form eines Verweises übergeben. Dieser Verweis gibt eine Funktion an, die durch ihren Rückgabewert der Form wahr (eins) oder falsch (null) über die weitere Ausführung entscheidet. Über diesen Mechanismus können in Anwendungen, in denen die Protokollfunktionalität verwendet wird, beliebige Bedingungen konstruiert und beim Definieren des Protokolls verwendet werden. Wie dies im Fall der ICU-Firmware genutzt wird, wird in einem späteren Abschnitt (Abschnitt 5.5) detailliert beschrieben.

5.3.2 ICU-spezifische Protokollfunktionen

Alle hardwareabhängigen und ICU-anwendungsspezifischen Funktionen, die für die protokollgesteuerte Ausführung von Abläufen benötigt werden, sind in das Modul *prot_ext* ausgelagert. Prinzipiell unterschiedlich sind hier zwei Gruppen von Funktionen: Wie im vorherigen Abschnitt beschrieben wurde, ist bereits für die Initialisierung des Moduls *prot* u. a. ein Verweis auf eine Funktion notwendig, die das Verfahren für die weitere Protokollausführung anhand des Parameters „Ausführzeitpunkts“ bestimmt. Zusätzlich werden Mechanismen und Funktionen für eben diese Verfahren und zur allgemeinen Ablaufkontrolle benötigt. In der zweiten Gruppe lassen sich alle hardwareabhängigen Funktionen zusammenfassen, welche die Funktionalität der vorhandenen Bausteine bzw. deren Module (s. Abschnitt 5.2) für die Verwendung innerhalb von Protokollen zur Verfügung stellen. Damit fungieren diese Funktionen als Bindeglied zwischen den Hardwareschnittstellen und der Protokollstruktur und ermöglichen den eigentlichen Zweck, nämlich eine protokollgesteuerte Hardwaresynchronisation von Systemkomponenten.

Zur Definition eines Ablaufprotokolls werden Funktionen bereitgestellt, über welche der Be-

ginn der Protokollausführung beeinflusst werden kann. Wie im Abschnitt über die Protokollstruktur und deren Speicherorganisation (Abschnitt 5.3.1) beschrieben wurde, besteht die Möglichkeit, ein Protokoll bereits auszuführen, während es noch definiert wird. Da hierbei jedoch bei sehr kleinen Verzögerungen der Kommandoausführungen ein Leerlaufen der Protokollliste möglich sein kann, ist der unmittelbare Start eines Protokolls nach Hinzufügen des ersten Eintrags in einigen Fällen nicht wünschenswert. Zur allgemeinen Ablaufkontrolle werden daher Funktionen bereitgestellt, welche die Aufnahme eines kompletten Protokolls über Angeben des Startpunkts und des Endes ermöglichen. Mit der Markierung des Protokollendes wird die Ausführung automatisch gestartet. Durch die Ringstruktur der Protokollliste ist es möglich, über den soeben beschriebenen Mechanismus mehrere Protokolle hintereinander zu definieren. Über die Markierung des jeweiligen Anfangs wird ein interner Haltepunkt gesetzt, der im Falle der Definition mehrerer Protokolle das Ende der Abarbeitung zur Folge hat, sobald diese an die durch den Haltepunkt markierte Listenposition gelangt. Erst durch Markieren des Protokollendes wird die weitere Ausführung ab der durch den Haltepunkt markierten Position wieder aufgenommen.

Die Bedingung für den Zeitpunkt der Ausführung eines Kommandos soll für jeden Protokolleintrag individuell anpassbar sein. Das Modul *prot_ext* enthält hierfür eine Funktion, die das Verfahren für die weitere Protokollausführung anhand des übergebenen Parameters „Ausführzeitpunkt“ festsetzt. Ein Verweis auf diese Funktion wird bei der Initialisierung des Moduls *prot* übergeben, so dass sie aus der Funktion zur Ausführung eines Protokolls (vgl. Flussdiagramm in Abbildung 5.7) aufgerufen werden kann. Innerhalb der ICU-Anwendung wird grundsätzlich zwischen zwei verschiedenen Verfahren zur Bestimmung des Ausführzeitpunkts unterschieden: Zeitabhängig oder ereignisabhängig. Der Parameter „Ausführzeitpunkt“ beschreibt eine positive, ganze Zahl mit einer Wortbreite von vier Byte (Bereich 0 bis $2^{32} - 1$ (= $4.294.967.295$) bzw. 0 bis 0xFFFFFFFF (hexadezimal)). Der komplette Wertebereich des Parameters ist entsprechend der verschiedenen Verfahren in zwei Intervalle unterteilt:

Im Intervall von 0 bis 0xFFFFFFCF ist der Parameter als Zeiteinheit auf der Basis von $10\,\mu s$ definiert und beschreibt die Ausführzeit jeweils relativ zum Ausführzeitpunkt des vorherigen Kommandos. In diesem Bereich können daher Verzögerungen von $0\,\mu s$ bis 0xFFFFFFCF$*10\,\mu s$ angegeben werden. Die maximale Verzögerung von einem Kommando zum nächsten entspricht damit $42.949.672.470\,\mu s \approx 11\,h\colon 56\,min$. Sollen größere Verzögerungen zwischen zwei Kommandos eingestellt werden, kann dies über das Einfügen eines oder mehrerer Dummykommandos (Kommandos ohne Funktion) mit entsprechenden Ausführzeitpunkten erfolgen. Die zeitliche Dynamik der Ablaufprotokolle kann sich somit bei einer Auflösung von $10\,\mu s$ flexibel bis über Stunden, Tage oder Wochen erstrecken, ohne nach oben beschränkt zu sein.

Das zweite Intervall erstreckt sich von 0xFFFFFFD0 bis 0xFFFFFFFF und ist für das Referenzieren von ereignisabhängigen Verfahren zur Bestimmung des Ausführzeitpunkts und einige Sonderfälle reserviert. Die Untermenge des Bereichs von 0xFFFFFFD0 bis 0xFFFFFFEF ist zur Spezifizierung von flexibel formulierbaren Ausführbedingungen vorgesehen, deren Programmierung und funktionale Möglichkeiten im Weiteren (Abschnitt 5.5) noch eingehend vorgestellt werden.

Da die verbleibenden Werte Sonderfälle beschreiben, die überwiegend für den firmwareinter-

nen Gebrauch reserviert sind, soll lediglich auf den Wert 0xFFFFFFF2 eingegangen werden: Kommandos, deren Ausführzeitpunkt mit diesen Wert angegeben ist, werden *nicht* der Protokollliste hinzugefügt, sondern unmittelbar und unabhängig von jeglicher Protokollaktivität ausgeführt. Hierdurch besteht die Möglichkeit, von einander entkoppelte Abläufe über die ICU parallel abarbeiten zu lassen. Allerdings erfolgt lediglich der durch das Protokoll beschriebene Ablauf zeitgenau. Durch Verwendung dieses Mechanismus besteht jedoch die Gefahr, dass unbeabsichtigt in einen vordefinierten, protokollgesteuerten Ablauf eingegriffen wird, indem z. B. ein Objekttisch zusätzlich zu den im Protokoll festgelegten relativen Bewegungen verschoben wird. Der weitere Verlauf des Protokolls wäre damit nicht mehr entsprechend der durch das Protokoll definierten Planung und würde sehr wahrscheinlich zu unbrauchbaren Ergebnissen führen. Unter Umständen könnte dies sogar zu Schäden an den eingesetzten Geräten führen, wenn z. B. ein protokollgesteuerter Pipettiervogang und ein automatischer Wechsel, initiiert durch ein protokollunabhängiges Kommando, gleichzeitig mechanisch auf eine Mikrotiterplatte zugreifen. Bei der Verwendung der Konstanten 0xFFFFFFF2 für unmittelbare, protokollunabhängige Ausführung ist daher äußerste Vorsicht geboten.

Beschreibt der Parameter „Ausführzeitpunkt“ eine Zeitabhängigkeit, wird ein entsprechendes Verfahren zur Bestimmung der Ausführung basierend auf der SEROS-Systemzeit verwendet. Dieses zeitabhängige Verfahren wird durch eine Funktion beschrieben, welche die durch den Parameter „Ausführzeitpunkt“ vorgegebenen Differenz mit der seit der Ausführung des letzten Befehls vergangenen Zeit vergleicht. Bei entsprechend fortgeschrittener Systemzeit wird die Ausführung des Protokolleintrags durch Aufrufen der Ausführ-Funktion (s. o.) eingeleitet. Um ein langfristiges Driften der Ausführzeiten einer Sequenz von zeitabhängigen Kommandos zu vermeiden, wird die neue Ausführzeit theoretisch über Addition des Parameters „Ausführzeitpunkt“ und des theoretischen Ausführzeitpunkts des letzten Kommandos ermittelt. Minimale Verzögerungen bedingt durch den Aufruf der Ausführ-Funktion bzw. den Aufruf der Kommandofunktionen führen somit im Weiteren Verlauf des Protokolls nicht zu einem schleppenden Zeitverzug.

Das Module *prot_ext* bietet zusätzlich Funktionen für Standardaufgaben innerhalb von Protokollen. In einer Funktion wird z. B. eine einfache Bedingung für Schleifen mit einer definierten Anzahl von Durchläufen formuliert. Beim Eintragen des Schleifenkommandos (s. Seite 79) in die Protokollliste wird ein Verweis auf die Funktion dieser Bedingung und durch einen Parameter die Anzahl der Schleifendurchläufe übergeben.

Der zweite Teil des Moduls *prot_ext* besteht aus den Funktionen zur Einbindung der Hardware in protokollbasierte Abläufe. Da die Funktionalität dieses Teils in einem späteren Abschnitt detailliert beschrieben wird (Abschnitt 5.6.3), soll an dieser Stelle hierauf nicht weiter eingegangen werden.

Nach der Beschreibung der beiden Module *prot* und *prot_ext*, die den Kern der protokollbasierten Ausführung von vordefinierten Abläufen bilden, veranschaulicht Abbildung 5.8 die Verknüpfung der beiden Module untereinander sowie die Verbindung zu den Modulen, welche die Schnittstellen zur ICU-Hardware liefern: Das Modul *prot_ext* bietet Funktionen, die über einen an späterer Stelle (Abschnitt 5.6) erklärten Mechanismus in die durch das Modul *prot* zur Verfügung gestellte Protokollliste eingetragen werden können. Über diese Funktionen wird u. a. die Funktionalität der Hardwareschnittstellen für die Verwendung innerhalb eines Protokolls

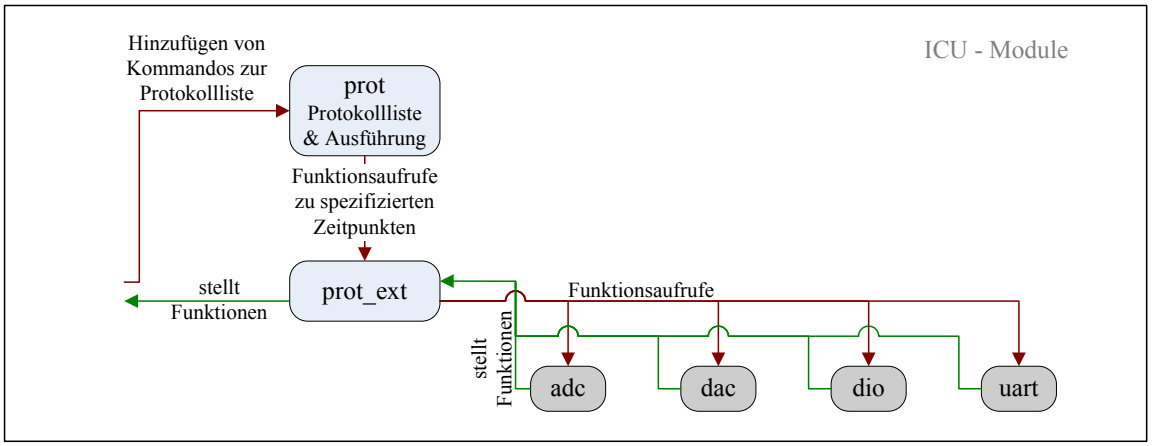


Abbildung 5.8: Module für die Protokollfunktionalität der Firmware und deren Verknüpfung. Blau hinterlegte Modulblöcke bieten zusätzliche Schnittstellen, die im Weiteren in anderen ICU Modulen noch verwendet werden. Grau hinterlegte Blöcke verweisen auf bereits beschriebene Module

ermöglicht. Während der Abarbeitung werden z. B. zu im Protokoll festgelegten Zeitpunkten die entsprechend eingetragenen Funktionen des Moduls *prot_ext* aufgerufen. Dies führt entweder zur Ausführung protokollinterner Funktionen wie z. B. Schleifen oder zum Bedienen der Hardwareschnittstellen.

5.4 Eingebettete Mathematik

Neben einer allgemeinen, protokollbasierten Ablaufsteuerung von sequenziellen Kommandos wird durch Schleifen und bedingungsabhängigen Verzweigungen die Definition komplex strukturierter Abläufe ermöglicht. Um die Flexibilität der Protokollgestaltung zusätzlich zu erhöhen, wurden die beiden soeben beschriebenen Module um ein weiteres Modul ergänzt, das die Verwendung mathematischer Funktionalität innerhalb von Protokollen ermöglicht: Modul *prot_math*. In Anlehnung an moderne Programmiersprachen können über dieses Modul Variablen und mathematische Operationen in der Definition von Protokollen verwendet werden.

Für Variablen und Konstanten werden bei der Initialisierung des Moduls intern Listen angelegt, in deren Elemente die entsprechenden Werte abgespeichert werden. Bei einer Deklaration muss jeweils der Typ der Variablen bzw. Konstanten (ganze Zahl, Fließkommazahl, etc.), ein Name in Form einer ein-Byte Zahl und ein Anfangswert übergeben werden. Der numerische Variablenname spezifiziert die Position in der entsprechenden Liste. Die Werte von Konstanten können nach der Deklaration nicht mehr verändert werden. Einer Variablen kann auf zwei verschiedene Weisen ein Wert zugewiesen werden: Entweder über eine entsprechende Funktion, der neben dem Typ und dem Namen der Variablen der zuzuweisende Wert übergeben wird, oder mittels eines mathematischen Ausdrucks.

Damit können als Variable nicht nur vorgegebene numerische Werte eingegeben werden, sondern auch die Ergebnisse mathematischer Berechnungen. Durch Angabe des Typs und des Namens können über eine weitere Funktion die Werte von Variablen und Konstanten für die Übergabe an Protokollkommandos ausgelesen werden. Weiterhin können Variablen und Konstanten unter Angabe des Namens in mathematischen Ausdrücken eingesetzt und somit für Berechnungen und/oder logische Operationen verwendet werden. Auf Letzteres wird im Fol-

genden noch näher eingegangen.

Das Modul *prot_math* bietet neben der Verwaltung von Variablen und Konstanten eine Funktion zur Evaluierung von mathematischen Ausdrücken. Ähnlich einer Programmiersprache wird ein mathematischer Ausdruck durch eine Sequenz von numerischen Werten, Variablen, Konstanten und mathematischen Operatoren gebildet, die in Form einer ASCII-Zeichenkette an die Evaluierungsfunktion übergeben wird. In einem rekursiven Verfahren wird ein solcher Ausdruck grammatikalisch bestimmt und mathematisch ausgewertet. Als Rückgabewert der Funktion wird nach Übergabe und Evaluierung eines syntaktisch und semantisch korrekten Ausdrucks dessen Ergebnis an die aufrufende Funktion zurückgegeben. Fehler innerhalb des an die Evaluierungsfunktion übergebenen Ausdrucks werden mit Angabe der Fehlerart anstelle eines Ergebnisses zurückgegeben. Eine Referenzierung von Variablen und Konstanten innerhalb von mathematischen Ausdrücken erfolgt über einen Bezeichner („v“ für Variable und „c“ für Konstante) mit unmittelbar anschließendem numerischen Wert. Die Kombination aus dem Bezeichner und der numerischen Zahl spezifiziert die entsprechende Liste und die Listenposition wodurch der gewünschte Wert eindeutig referenziert ist. Der Ausdruck „v5“ entspricht somit der Variablen, die innerhalb der Liste für Variablen auf Position 5 abgelegt ist. Variablen, die innerhalb von mathematischen Ausdrücken eingesetzt werden sollen, müssen im Vorfeld zunächst deklariert werden.

Neben der Verwendung von konstanten Zahlenwerten, die in ASCII ausgeschrieben werden, können innerhalb mathematischer Ausdrücke Wertzuweisungen (’=’), verschiedene arithmetische Rechenoperationen (’+’, ’-’, ’*’, ’/’, ’%’ (Modulo)) und logische Operationen (’$<$’ (kleiner), ’$>$’ (größer), ’==’ (gleich)) verwendet werden. Desweiteren können Teile von Ausdrücken durch entsprechende Klammerung zusammengefasst werden (’(’, ’)’). Komplexere mathematische, z. B. der Trigonometrie entstammende Operationen, können innerhalb der Firmware durch Angabe eines Bezeichners und eines Verweises auf die entsprechend programmierte Funktion einfach in das Modul aufgenommen werden und stehen damit für die Evaluierung zur Verfügung. Über diese Erweiterungsmöglichkeit wurden Funktionen zum Potenzieren (’^’; z. B. „5^2“ $\widehat{=} 5^2$) sowie Sinus („sin“) und Cosinus („cos“) zum Funktionsumfang des Moduls hinzugefügt. Variablen und vordefinierte Konstanten können über die bereits beschriebene Form in mathematischen Ausdrücken verwendet werden. Tabelle 5.1 verdeutlicht den Aufbau der Ausdrücke anhand einiger Beispiele.

Im Zuge der Optimierung des Moduls im Hinblick auf Geschwindigkeit wurde die interne Verwendung von rein ASCII basierten mathematischen Ausdrücke auf eine Mischung von ASCII und hexadezimal interpretierten Zeichen umgestellt. Um die einfache Eingabeschnittstelle für den Benutzer beizubehalten, werden als Zeichenkette übergebene numerische Werte bei der Definition von mathematischen Ausdrücken in entsprechende hexadezimale Ausdrücke umgewandelt. Beispielsweise wird eine als „v53“ übergebene Variable intern zu „v5“ (ASCII von ’5’ $\widehat{=} 0x35$(hexadezimal) = 53 (dezimal)) umgewandelt. Konstante Zahlenwerte werden nach ähnlichem Schema aus den Ausdrücken extrahiert, in numerische Werte umgewandelt und mit einer internen Referenz versehen, die zur Laufzeit von Protokollen verwendet wird.

Die mathematische Funktionalität des in diesem Abschnitt vorgestellten Moduls ist auf verschiedenste Weisen im Rahmen von Protokolldefinitionen einsetzbar. Einerseits können Berechnungen und einfache Zuweisungen zur Ausführzeit eine Manipulation und flexible Übergabe

Ausdruck	Mathematische Form	Rück-gabewert	Kommentar
„v1=5“	$v1 = 5$	5	einfache Wertzuweisung
„2*v1“	$2 * v1$	10	einfache Multiplikation
„v2=2*v1“	$v2 = 2 * v1$	10	Zuweisung des Ergebnisses einer einfachen Multiplikation
„v3=v1^2“	$v3 = v1^2$	25	Zuweisung des Ergebnisses einer Potenzierung
„v4=sin(3.1416)“	$v4 = sin(3.1416)$	~ 0	Zuweisung des Ergebnisses der Anwendung der Sinusfunktion
„v1==6“	$v1 == 6$	0	Logischer Vergleich der Variable v1 mit der Zahl 5; 0 ≙ falsch
„v1<v2“	$v1 < v2$	1	Logischer Vergleich der Variablen v1 & v2; 1 ≙ wahr

Tabelle 5.1: Beispiele für mathematische Ausdrücke in ASCII-Format, die durch das Modul *prot_math* evaluiert werden können.

von Parametern ermöglichen. Dies kann z. B. eingesetzt werden, um in einer Schleife von Bildaufnahmen die Spannung eines analogen Ausgangs inkrementell zu erhöhen. Wird die analoge Spannung zur Steuerung einer Fokussiereinrichtung verwendet, führt dies zum Erstellen eines Bildstapels. Über dieses Verfahren können somit Parameter zwischen unterschiedlichen Kommandos innerhalb des Protokolls unmittelbar, oder durch Berechnungen bearbeitet, ausgetauscht werden. Andererseits können Variablen und/oder die Ergebnisse von Berechnung sowie logischen Operationen als Bedingungen für Schleifen oder Verzweigungen eingesetzt werden, um somit einen direkten Einfluss auf den Protokollablauf zu nehmen. Desweiteren sind über den Einsatz von Variablen Anwendungen wie z. B. Zähler(-Variablen) zur Ermittelung der Häufigkeit von Vorgängen, Kurzspeicher zum Speichern von Zwischenergebnissen usw. denkbar. Die verschiedenen Mechanismen zur Einbindung der Funktionalität des Moduls *prot_math* in die Definition von protokollbasierten Abläufen werden in den folgenden Abschnitten beschrieben.

5.5 Ereignisse und Bedingungen

Durch die in Abschnitt 5.3 beschriebenen Module wird eine protokollbasierte Ausführung von Abläufen zu im Vorfeld spezifizierten Zeitpunkten ermöglicht. Zusätzlich zu dieser strikt deterministischen Charakteristik sollen flexible Protokolldefinitionen in Abhängigkeit von während der Ausführzeit auftretenden Ereignissen bzw. evaluierten Bedingungen möglich sein. Hierüber soll u. a. die geforderte Fähigkeit zur Adaption sowie die Multi-Master-Fähigkeit des Systems bewerkstelligt werden. Wie bereits in Verbindung mit Schleifen und verzweigter Ausführung von Protokollblöcken (Seite 79) sowie in Verbindung mit ereignisabhängiger Ausführung von Kommandos (s. Seite 80) angesprochen, wurden interne Schnittstellen über Verweise auf Funktionen geschaffen, die eine flexible Angabe der jeweiligen Bedingung bzw. des jeweiligen Ereignisses (im Folgenden *Events*) erlauben. Die Möglichkeit der flexiblen Konfiguration dieser Events und der Verwendung innerhalb der Definition von Protokollen darf sich dabei nicht

auf die Ebene der Firmware beschränken, sondern muss vielmehr durch den Benutzer möglich sein. Aus diesem Grund wurde eine Struktur zur Verwaltung von Events sowie Funktionen zur Konfiguration und zur Einbindung der Events in Protokolldefinitionen geschaffen (Modul *event_man*). Ähnlich der Implementierung der Protokollfunktionalität wurde aus Gründen der Portabilität des Quellcodes der anwendungsunabhängige Teil von ICU-spezifischen Komponenten getrennt und in einem separaten Modul(Modul *event_ext*) zusammengefasst.

Die Struktur zum Verwalten der Events besteht aus einer Liste, deren Einträge die jeweils notwendigen Daten *eines* Events enthalten:

- Typ
- Modus
- Verweis auf die Event-Funktion
- Verweis auf die Parameter
- Parameteranzahl

Der Event-Typ spezifiziert den Mechanismus, mit dessen Hilfe das Ergebnis der jeweiligen Eventabfrage ermittelt wird. Die Vielzahl der möglichen Mechanismen kann generell in drei verschiedene Gruppen unterteilt werden: Protokollbasierte „Soft-Events“ (z. B. eine logische Operation oder arithmetische Berechnung), Firmware-interne „Firm-Events“ (z. B. Empfang eines vordefinierten Zeichens an einer der EIA-232-Schnittstellen) oder hardwarebasierte „Hard-Events“ (z. B. Status von Trigger- oder Sync-Leitungen). Die derzeit implementierten Events werden in Abschnitt 5.5.1 vorgestellt.

Über den Modus kann für jedes Event individuell einer von drei möglichen Modi ausgewählt werden: Direkt, Sample&Hold oder Kumulativ. Der Unterschied der drei Modi soll anhand des Beispiels eines Events vom Typ „digitaler Eingang“ mit erwartetem Zustand „logisch eins“ veranschaulicht werden.

Ein im Direkt-Modus konfiguriertes Event überprüft zum Zeitpunkt der Event-Abfrage den Zustand des digitalen Eingangs. Abhängig vom Vergleich mit dem erwarteten Zustand wird der Wert „wahr“ oder „falsch“ als Ergebnis der Eventabfrage zurück gegeben. Im Modus Sample&Hold wird der Zustand des digitalen Eingangs nicht nur einmal zum Zeitpunkt der Event-Abfrage, sondern quasi-kontinuierlich überprüft. Wird der erwartete Zustand mindestens einmal registriert, so wird dies gespeichert und eine Abfrage des Events ergibt den Wert „wahr“, auch wenn der Zustand des digitalen Eingangs zum Zeitpunkt der Abfrage nicht mit dem erwarteten Zustand übereinstimmt. Durch die Abfrage des Events wird der gespeicherte Zustand wieder zurückgesetzt. Der Modus Kumulativ erweitert den zuletzt beschriebenen Modus Sample&Hold insofern, als zusätzlich die Anzahl der registrierten Übereinstimmungen aufaddiert und festgehalten wird. Die Abfrage des Events führt bei einer oder mehreren registrierten Übereinstimmungen zu einer positiven Rückantwort und zu der Verringerung der gezählten Anzahl von Übereinstimmung um einen Zähler. Mehrfach registrierte Übereinstimmungen zwischen zwei Event-Abfragen führen somit bei mehreren Abfragen des Events zu entsprechend vielen positiven Abfrageergebnissen.

Als notwendige Daten des Eintrags werden neben dem Typen und dem Modus des Events bei der Konfiguration zusätzlich ein Verweis auf die Event-Funktion, die beim Aufruf dieser Funk-

tion zu übergebenden Parameter sowie deren Anzahl in der Event-Liste hinterlegt. Da ein Event-Datensatz durch die Konfiguration an einer bestimmten Position innerhalb der Event-Liste abgelegt wird, kann das jeweilige Event durch Angabe dieser Listenposition eindeutig referenziert werden. Die Listenposition stellt damit eine abstrakte Referenz, ein so genanntes Handle, dar.

Nach entsprechender Definition kann eine mit dem Event-Handle korrespondierende Konstante (s. Seite 80) anstelle des Parameters Ausführzeitpunkt (s. Abschnitt 5.3.1) eingesetzt werden. Auf diesem Weg kann der Ablauf eines Protokolls in Abhängigkeit von äußeren Einflüssen gesteuert bzw. auf diese synchronisiert werden. Neben der Verwendung zur Bestimmung des Ausführzeitpunkts können Events durch Angabe des entsprechenden Handles zudem als Bedingungen für Schleifen oder verzweigter Ausführung eingesetzt werden und ermöglichen somit eine flexible Gestaltung des Protokollflusses.

Die Konfigurierung führt gleichzeitig zu einer Blockade des jeweiligen Events. Dies dient zum Schutz vor einer unbeabsichtigten Neukonfigurierung, da diese bei Verwendung des ursprünglichen Events innerhalb von bereits definierten Protokollen unweigerlich zu Fehlern führen würde. Events, die für die weiteren Abläufe nicht mehr benötigt werden, können allerdings freigegeben werden. Nach einer erfolgten Freigabe kann der damit wieder zu Verfügung stehende Eintrag der Event-Liste für eine erneute Konfigurierung verwendet werden.

Die Funktion der Event-Abfrage dient unter Angabe eines Handles zum einfachen Aufrufen von im Vorfeld konfigurierten Events. Der über das Handle referenzierte Listeneintrag enthält einen Verweis auf die entsprechende Event-Funktion. Beim Aufruf der so spezifizierte Funktion werden die hinterlegten Parameter mit übergeben. Der Rückgabewert der aufgerufenen Funktion wird als Ergebnis der Event-Abfrage an die aufrufende Funktion zurückgeliefert.

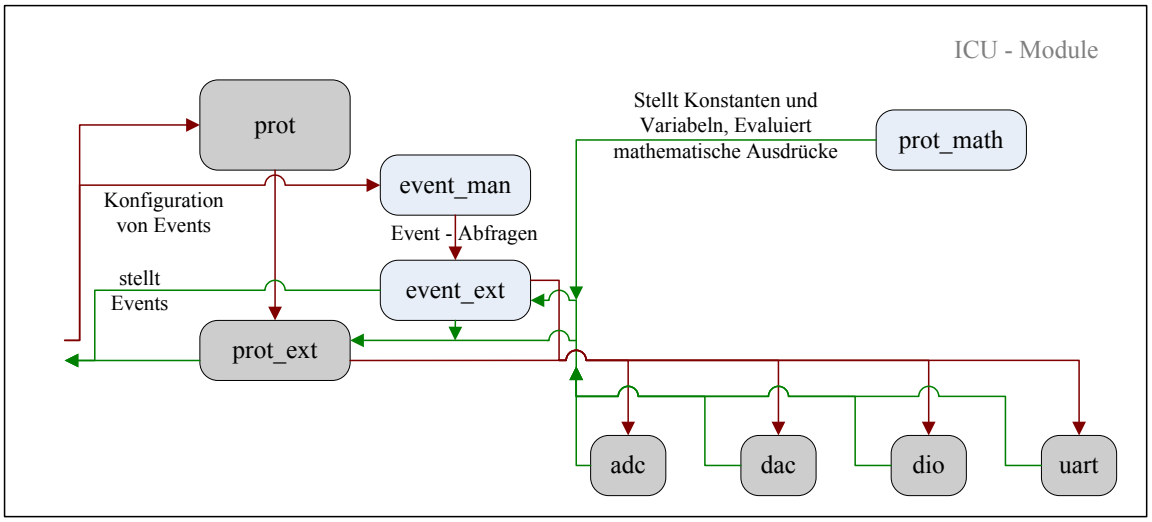


Abbildung 5.9: Die Module *event_man* und *event_ext* erweitern die Protokollfunktionalität um flexibel konfigurierbare Bedingungen für Schleifen und Verzweigungen sowie für bedingungsabhängige Ausführzeitpunkte von Protokollkommandos. Das Modul *prot_math* ermöglicht zudem die Verwendung mathematischer Funktionalitäten, die ähnlich denen von imperativen Programmiersprachen wie z. B. C sind. Diese können sowohl für mathematischen Anweisungen durch Einträge innerhalb von Protokollen als auch innerhalb der Beschreibung von Bedingungen für Schleifen bzw. bedingter Ausführung oder zur Spezifikation des Ausführzeitpunkts eines Kommandos verwendet werden.

Das vereinfachte Blockdiagramm in Abbildung 5.9 veranschaulicht die in diesem Abschnitt beschriebenen Zusammenhänge zwischen den Modulen *event_man* und *event_ext* sowie dem Modul *prot_math* und der in Abschnitt 5.3 vorgestellten Protokollfunktionalität.

5.5.1 Aktuelle Events

Als Erweiterung zur soeben beschriebenen, hardwareunabhängigen Struktur und den Funktionen zur Verwaltung und Verwendung dieser Struktur, wurden entsprechend den Anforderungen an die ICU-Firmware eine Reihe elementarer Events in einem separaten Modul (Modul *event_ext*) zusammengefasst. In der aktuellen Version stehen für die Konfigurierung somit folgende Event-Typen zur Verfügung:

- Digitaler Eingang Event
- Software Event
- Mathematisches Event
- EIA-232 Event
- Warten Event

Wie im Beispiel des vorherigen Abschnitts angesprochen, kann der Zustand eines digitalen Eingangs bzw. einer bidirektionalen Synchronisationsleitung über dieses Event mit einem erwarteten Wert verglichen werden. Der Event-Funktion müssen neben der eindeutigen Angabe des bzw. der zu vergleichenden digitalen Eingangs bzw. Eingänge (Bank, Leitungs-Bit-Maske) auch die erwarteten Zustände in Form einer Bitmaske übergeben werden. Durch Verwendung der Funktion des Moduls *dio* zum Auslesen von digitalen Eingängen wird daraufhin entsprechend dem konfigurierten Event-Modus die Antwort auf eine Anfrage ermittelt.

Durch Senden eines speziellen Kommandos (s. Abschnitt 5.6) kann vom PC aus ein so genanntes Software-Event entsprechend dem konfigurierten Modus ausgelöst werden. Über den Einsatz dieses Events anstelle eines Ausführzeitpunkts können Haltepunkte in ein Protokoll eingebaut werden, mittels derer die Ausführung von Echtzeitprotokollen und ggf. parallel laufende, PC-basierte Abläufe synchronisiert werden kann. Als Beispiel sei hier die Aufnahme schneller Bildsequenzen genannt, in denen die Kamerasynchronisation auf Hardwareebene lediglich zum Starten einer Belichtung verwendet werden kann. Da die Belichtung eines neuen Bildes erst dann beendet werden darf, wenn das vorausgegangene Bild komplett vom PC ausgelesen wurde, muss das erneute Auslösen einer Bildaufnahme verzögert werden. In diesem Fall gibt der PC nach erfolgreicher Übertragung der Bilddaten über ein Software-Event die protokollbasierten Vorgänge zum Auslösen einer neuen Beleuchtung bzw. Belichtung frei.

Über das mathematische Event besteht die Möglichkeit, die in Abschnitt 5.4 beschriebenen mannigfaltigen Möglichkeiten durch die Evaluierung mathematischer Ausdrücke als Bedingungen in einer Protokolldefinition zu verwenden. Dabei liegt der Schwerpunkt der Verwendung mathematischer Ausdrücke im Rahmen von Event-Definitionen auf der Formulierung von logischen Verknüpfungen als Bedingungen für Schleifen und Verzweigungen. Bei der Definition des mathematischen Events wird neben einem Verweis auf die Evaluierungsfunktion des Moduls *prot_math* ein durch den Benutzer definierter mathematischer Ausdruck der beschriebenen Form in die Event-Struktur eingetragen. Das Ergebnis der Event-Abfrage wird wiederum entsprechend dem Event-Modus ermittelt.

Zur optimalen Synchronisation von Geräten, die über eine der EIA-232-Schnittstellen an die ICU angeschlossen sind, steht ein EIA-232-Event zu Verfügung. Dieses berücksichtigt, dass die Kommunikationsstruktur vieler über EIA-232 angeschlossener Endgeräte entweder eine feste Anzahl von Zeichen oder ein definiertes Zeichen für die Markierung des Endes einer Übertragung vorsieht. Über die Konfiguration des Events wird neben der EIA-232-Schnittstelle, über welche das Endgerät an die ICU angeschlossen ist, auch bestimmt, ob das Event aufgrund einer bestimmten Anzahl empfangener Zeichen oder aufgrund eines definierten „Ende-der-Übertragung"-Zeichens ausgelöst wird. Für größtmögliche Flexibilität kann die Anzahl der Zeichen bzw. das Zeichen, welches das Ende einer Übertragung signalisiert, bei der Konfiguration bestimmt werden.

Wird beim Hinzufügen eines Kommandos in die Protokollliste anstelle einer zeitlichen Verzögerung ein Event durch Einsetzen der entsprechenden Konstante spezifiziert und kommt es während der Ausführung des Protokolls aufgrund dieses Events zu einer Verzögerung der Abläufe, so wird firmwareintern das „Warten Event" aktiviert. Eingesetzt als Bedingung einer Schleife oder einer verzweigten Ausführung kann somit auf unvorhergesehene Verzögerungen reagiert werden und z. B. eine Echtzeitschleife abgebrochen oder die Verzögerung durch eine Meldung an den PC signalisiert werden. Das „Warten-Event" ist aufgrund der Aktivierungsbedingung auf die Verwendung im Sinne einer Bedingung für Schleifen und Verzweigungen beschränkt. Anstelle einer zeitlichen Verzögerung eingesetzt, würde während der Protokollausführung der Event-Status auf aktiv gesetzt werden, sobald das Event abgefragt wird. Dies hätte zur Folge, dass das Kommando ohne weitere Verzögerung ausgeführt wird. Der Einsatz des „Warten-Event" zur Bestimmung des Ausführzeitpunkt eines Kommandos ist daher nicht sinnvoll.

5.6 Definition der Ablaufsteuerung

Um dem Benutzer die bisher beschriebene Funktionalität der ICU zur Verfügung zu stellen, bedarf es einer Schnittstelle, mittels derer Grundeinstellungen der ICU konfiguriert, Protokolle, Events und mathematische Ausdrücke definiert oder die unmittelbare Ausführung von Kommandos initiiert werden kann. Gemäß dem in Kapitel 3 beschriebenen Konzept reiht sich die ICU in die Gruppe der intelligenten Geräte ein, die durch den PC für Echtzeit-Abläufe des Systems konfiguriert werden können, um diese anschließend - über Hardwaresynchronisation optimiert - auszuführen. Die durch die ICU gewährleistete Synchronisation der Systemkomponenten ermöglicht zusätzlich zu zeitlich deterministischen Abläufen auch interaktives sowie adaptives Eingreifen in die zuvor festgelegten Abläufe. Der Schnittstelle zum PC kommt damit eine besondere Bedeutung zu, da sie nicht nur für zeitunkritische Konfigurierungsvorgänge, sondern auch für möglichst schnelle, latenzfreie Kommunikation zwecks Umkonfigurierung der Abläufe z. B. aufgrund von Ergebnissen PC-basierter Bildanalyse verwendet wird.

5.6.1 Schnittstelle zum PC

Der verwendete DSP besitzt keine ausdrücklich für die Verbindung zu einem PC vorgesehene Schnittstelle. Allerdings besteht die Möglichkeit, eine der seriellen Schnittstellen in einem Modus zu betreiben, der dem EIA-232 Protokoll entspricht. Nach der Pegelanpassung über ein entsprechendes elektronisches Bauteil verfügt die DSP-Platine somit über eine vollwertige

EIA-232-Schnittstelle. Da diese sich bereits durch den Einsatz als Schnittstelle zur Firmwareaktualisierung etabliert hat, wird sie derzeit ebenfalls für die Kommunikation zwischen dem Anwender-PC und der ICU eingesetzt.

Firmwareseitig basiert die Schnittstelle zum PC auf dem Modul *si* (vgl. Abbildung 5.1). Entsprechend konfiguriert emuliert die Schnittstelle das EIA-232 Protokoll mit einer Baudrate von 57.600 Baud, einer Bitlänge von 8 Bit, mit einem Stoppbit und ohne Verwendung der Paritätsüberprüfung. Das auf dem Modul *si* aufsitzende Modul *rs232* (vgl. Abbildung 5.1) bietet neben der entsprechenden Konfigurierung der Schnittstelle zwei Datenspeicher (FIFOs) für empfangene und zu sendende Daten, sowie Funktionen zum Zugriff auf diese Datenspeicher. Die asynchrone Datenübertragung (voneinander unabhängiges und ggf. zeitlich versetztes Empfangen und Senden) erfolgt auf DSP-Seite interruptgesteuert. In Empfangsrichtung bedeutet dies, dass ein empfangenes Zeichen (1 Byte) über eine Interrupt Routine automatisch in den Empfangs-Datenspeicher geschrieben wird. Aus der jeweiligen Anwendung können die empfangenen Daten über eine Funktion entweder blockweise oder Byte für Byte ausgelesen werden. In Senderichtung werden Daten ebenfalls über eine Interrupt Routine aus dem entsprechenden Datenspeicher entnommen und an die Schnittstelle übergeben, sobald diese für das Senden bereit ist. Um unnötiges Aufrufen der Interrupt Routine zu vermeiden, wird die Sendeseite der asynchronen EIA-Schnittstelle deaktiviert, sobald keine weiteren zu sendenden Zeichen im Datenspeicher vorliegen. Werden aus der Anwendung heraus neue Daten (Byte oder blockweise) in den Sende-FIFO geschrieben, wird die Sendeseite der Schnittstelle wieder aktiviert.

Durch die interruptgesteuerte Übertragung ist garantiert, dass empfangene Daten durch die Ablage im Empfangs-FIFO nicht verloren gehen, und dass das Versenden eines Datenblocks mit maximaler Geschwindigkeit der Schnittstelle erfolgt. Über die logische Zwischenschaltung der FIFOs wird eine zeitliche Entkopplung der Schnittstellenaktivität und der Verarbeitung der Daten durch die Firmware erreicht. Dadurch kann, wie im Weiteren (Abschnitt 5.7) gezeigt wird, die zeitliche Abfolge der firmwareinternen Vorgänge unabhängig von der Schnittstelle zum PC koordiniert werden.

Mittelfristig ist geplant, die Schnittstelle zwischen ICU und PC auf eine USB-Verbindung umzustellen. Der firmwareinterne Mechanismus über die FIFOs soll dabei erhalten bleiben, um die Entkopplung der Schnittstellenaktivität von den Vorgängen innerhalb der Firmware weiterhin zu gewährleisten.

5.6.2 Kommando-Dekodierung

Mit dem Hintergedanken, die Schnittstelle zum PC mittelfristig auf USB umzustellen, wurde eine hardwareunabhängige Kommandostruktur mit dazugehörigem Verfahren zur Dekodierung von Kommandos aus einem quasi-kontinuierlichem Datenfluss entworfen (Modul *cmd_dec*). Im Hinblick auf die Verwendung der Schnittstelle sowohl zur Konfiguration als auch für schnelles, adaptives Eingreifen wurde die Kommandostruktur auf Übertragungssicherheit sowie minimales Übertragungsvolumen optimiert. Abweichend von der weit verbreiteten Verwendung von ASCII-basierten Übertragungsprotokollen setzt sich diese Kommandostruktur daher aus einer prinzipiell rein hexadezimal interpretierten Zeichenfolge zusammen.

Nach der in Abbildung 5.10 veranschaulichten Struktur besteht ein Kommando minimal aus fünf Zeichen bzw. Bytes: Einer Kommandonummer (1 Byte), der Länge des Kommandos in

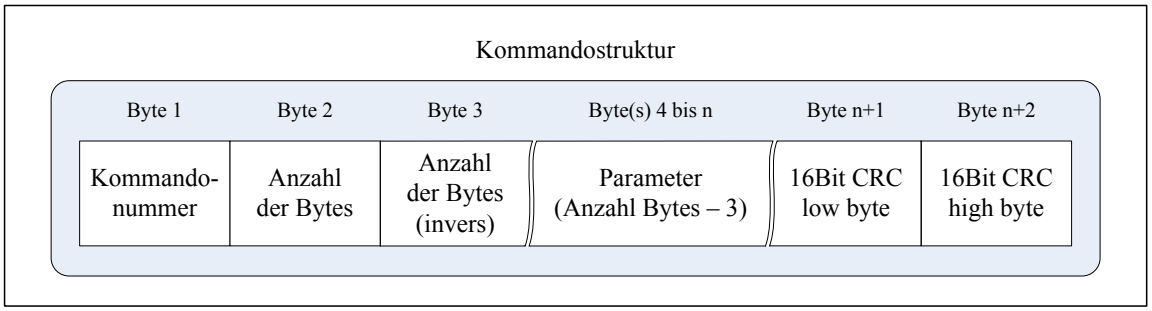


Abbildung 5.10: Kommandostruktur bestehend aus einer hexadezimal interpretierten Zeichenfolge. Die Position innerhalb der Übertragungssequenz kodiert die Bedeutung der einzelnen Bytes. Einem durch die Kommandonummer spezifizierten Kommando kann optional eine variable Anzahl von Parametern übergeben werden. Den Abschluss der Übertragung eines Kommandos bildet ein 16 Bit-CRC

Form der Byteanzahl (1 Byte), des invertierten Werts der Länge des Kommandos (1 Byte) und dem Wert einer abschließenden 16 Bit-CRC. Die maximale Gesamtlänge eines Kommandos ist daher durch die Angabe der in einem Byte kodierten Länge auf 255 Bytes reduziert. Optional können Kommandos somit Parameter von bis zu 252 Bytes (255 - 3 Bytes für Kommandonummer, Anzahl und inverse Anzahl der Bytes) enthalten. Die redundante Übertragung der Kommandolänge durch die Anzahl der Bytes und den inversen Wert der Byteanzahl dient zur Synchronisation der Übertragung. Da die Struktur Kommandos mit variierender Länge zulässt und da die Interpretation der einzelnen Bytes durch die Position innerhalb der Übertragungssequenz kodiert wird, ist die Synchronisation der Dekodierung eines kontinuierlichen Datenflusses von elementarer Bedeutung. Fehler in der Übertragung der Byteanzahl oder falsche Angabe der Kommandolänge würden dauerhaft zu Fehlinterpretationen der Dekodierung führen. Durch überprüfenden Vergleich der redundant übermittelten Kommandolänge werden Fehlinterpretationen nahezu ausgeschlossen. Durch das Dekodierverfahren ermittelte Fehler, wie z. B. der Empfang einer nicht registrierten Kommandonummer, nicht korrelierende Anzahl und inverse Anzahl der Bytes, oder ein negativer CRC führen zu Fehlermeldungen an den PC (vgl. Abschnitt 5.1.1).

Das in Abbildung 5.11 präsentierte Flussdiagramm veranschaulicht den Dekodierungsprozess: Aus einem Datenfluss werden nacheinander jeweils einzelne Bytes an die Dekodierungsfunktion übergeben. Nach der Initialisierung bzw. dem Zurücksetzen des Dekodierprozesses wird das zuerst übergebene Byte als Kommandonummer interpretiert und nach einer positiven Validierung als solches abgespeichert. Nach Umsetzen des Dekodierungsstatus auf das nächste zu erwartende Byte (Anzahl der Bytes) wird die Dekodierungsfunktion beendet. Ergibt die Validierung, dass eine Kommandonummer firmwareintern nicht registriert ist, wird eine Fehlermeldung an den PC gesendet und der Dekodierungsprozess zurückgesetzt. Beim nächsten Aufruf wird das an die Dekodierungsfunktion übergebene Byte als Kommandolänge interpretiert und als solche abgespeichert. Nach Umsetzen des Dekodierungsstatus auf das nächste zu erwartende Byte (die inverse Anzahl der Bytes) wird die Dekodierungsfunktion wiederum beendet. Das beim nächsten Aufruf übergeben Byte wird als inverse Anzahl der Kommandobytes interpretiert, invertiert und mit der bereits abgespeicherten Anzahl verglichen. Bei einem negativen Ergebnis des Vergleichs wird eine entsprechende Fehlermeldung an den PC gesendet und der Dekodierungsprozess erneut zurückgesetzt. Stimmen die als Anzahl und als inverse Anzahl

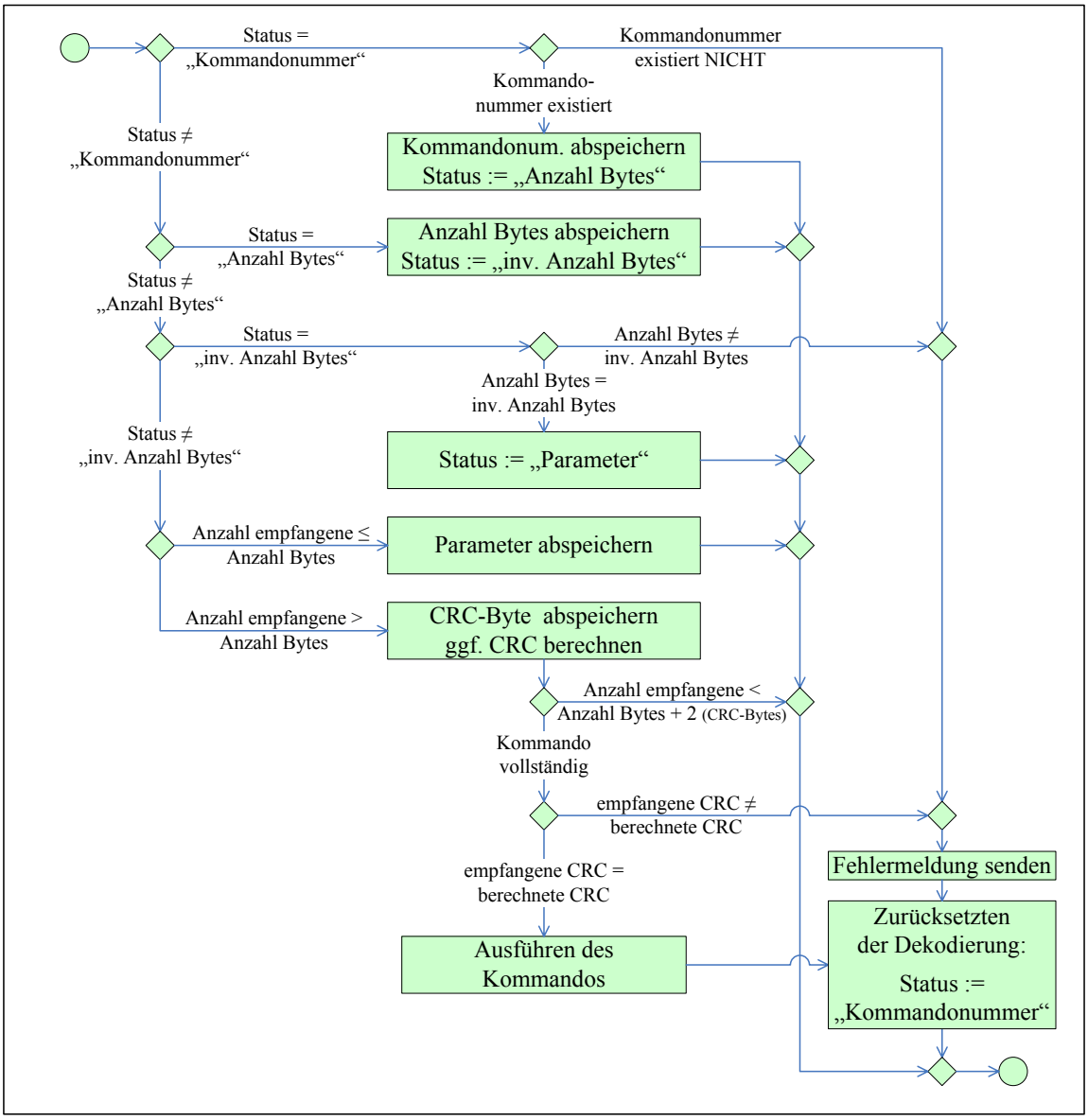


Abbildung 5.11: Flussdiagramm der Dekodierung von Kommandos: Die jeweils abhängig von der Position innerhalb der Übertragungssequenz erwarteten Bytes werden den entsprechenden Kommandoteilen zugeordnet und ggf. auf Stimmigkeit überprüft. Fehler führen zu einer Fehlermeldung und zum Zurücksetzen der Dekodierung.

übertragenen Kommandolängen überein, wird der Dekodierungsstatus auf das nächste zu erwartende Byte (Parameter) umgesetzt und die Dekodierungsfunktion beendet. Solange die Anzahl der empfangenen und an die Dekodierungsfunktion übergebenen Bytes kleiner als die durch das Kommando spezifizierte Kommandolänge ist, werden alle folgenden Zeichen als Parameter interpretiert und als solche abgespeichert. Die beiden im Anschluss daran übergebenen Bytes werden als nieder- bzw. höherwertiges Byte des 16 Bit-CRC interpretiert. Das Kommando ist erfolgreich dekodiert und wird ausgeführt, wenn der übertragene mit dem über das komplette Kommando berechnete CRC-Wert übereinstimmt. Stimmen die Werte nicht überein, wird wiederum eine Fehlermeldung an den PC gesendet und der Dekodierungsprozess zurückgesetzt.

Der Initialisierung des Moduls *cmd_dec* wird eine Liste übergeben, die für jedes mögliche Kommando eine eindeutige Kommandonummer sowie einen Verweis auf die bei erfolgreicher Deko-

dierung aufzurufende Funktion enthält. Die Definition der Kommandofunktionen kann auf diese Weise unabhängig von der Kodierungsstruktur und dem Dekodierungsverfahren in einem separaten Modul erfolgen. Zudem wird über diese Aufteilung wiederum eine Trennung der universell einsetzbaren und der anwendungsspezifischen Strukturen und Funktionen ermöglicht. Neben der Verwendung innerhalb der ICU-Firmware wird das Modul *cmd_dec* derzeit auch in der Firmware des Polychrome V (vgl. Abschnitt 1.2) eingesetzt. Auf der gleichen Kommandostruktur aufbauend und das gleiche Dekodierungsverfahren nutzend, werden neben einer Schnittmenge von gemeinsamen Kommandos wie (z. B. „Hardware-Reset“, Protokolldefinitions- oder Schleifenkommandos) individuelle, auf die jeweilige Anwendung zugeschnittene Kommandos eingebunden.

5.6.3 Kommando-Ausführung

Innerhalb der ICU-Firmware werden die Funktionen aller vom PC aus verwendbaren Kommandos im Modul *cmd_ext* zusammengefasst. Bei der Initialisierung des Moduls wird eine Liste aller Funktionen und der dazugehörigen Kommandonummern erstellt, die im Weiteren an die Initialisierung des im vorherigen Abschnitt beschriebenen Moduls (Modul *cmd_dec*) übergeben wird. Da nicht alle Kommandos für die Verwendung innerhalb von Protokolldefinition vorgesehen sind, ergibt sich automatisch eine grobe Zweiteilung.

Kommandos, die einem Protokoll hinzugefügt werden können, haben mit der Angabe des Ausführzeitpunkts mindestens einen Parameter. Der Aufbau dieser Kommandos folgt i. d. R. einem gemeinsamen Schema: Nach der Überprüfung, ob die korrekte Anzahl an Parameter übergeben wurde, wird der Ausführzeitpunkt aus der Parameterliste extrahiert und verifiziert (ist ein Event spezifiziert, wird z. B. überprüft, ob dieses Event bereits definiert ist). Spezifiziert der als Ausführzeitpunkt angegebene Parameter durch den Wert 0xFFFFFFF2 (vgl. Abschnitt 5.3.2, Seite 80) eine sofortige, protokollunabhängige Ausführung, so wird die mit dem Kommando verbundene Funktion unmittelbar aufgerufen. In allen anderen Fällen wird das Kommando mit dem Ausführzeitpunkt, einem Verweis auf die mit dem Kommando verknüpfte Funktion und einem Verweis auf die Parameter sowie deren Anzahl der Protokollliste hinzugefügt (vgl. Abbildung 5.6). Die erfolgreiche Aufnahme des Kommandos in die Protokollliste wird dem PC mit einer Rückmeldung bestätigt. Außerhalb der Aufnahme Protokollblocks wird die Ausführung nach Hinzufügen des Kommandos unmittelbar gestartet (vgl. Abschnitt 5.3.2). Etwaig auftretende Fehler, wie z. B. eine falsche Anzahl an Parametern, die Verwendung eines noch nicht definierten Events anstelle des Ausführzeitpunkts, oder Fehler beim Hinzufügen des Kommandos in die Protokollliste (z. B. Speicherüberlauf) werden durch entsprechende Meldungen an den PC weitergeleitet. Vereinfacht dargestellt tragen diese Art von Kommandos Funktionen des Moduls *prot_ext* mitsamt den beim Funktionsaufruf notwendigen Parameterdaten in die Protokollliste ein (s. Abbildung 5.12, Modulkette *prot_ext* → *cmd_ext* → *prot*).

In der zweiten Gruppe sind alle Kommandos zusammengefasst, die nicht unmittelbar in Verbindung mit der Definition von Protokollen stehen. Hierunter fallen z. B. Kommandos zum Auslesen der Systemkonfiguration oder zum firmwareseitigen Auslösen eines Hardware-Reset sowie Servicekommandos wie z. B. zum Schreiben der Systemkonfiguration oder zum Auslesen ICU-interner Zeitmessungen. Weiterhin sind hier die Kommandos zu nennen, die vorbereitende bzw. administrative Funktion haben, wie z. B. die Kommandos zu Starten und Beenden der Definition eines Protokollblocks, die Funktion zum Löschen der gesamten Protokollliste oder

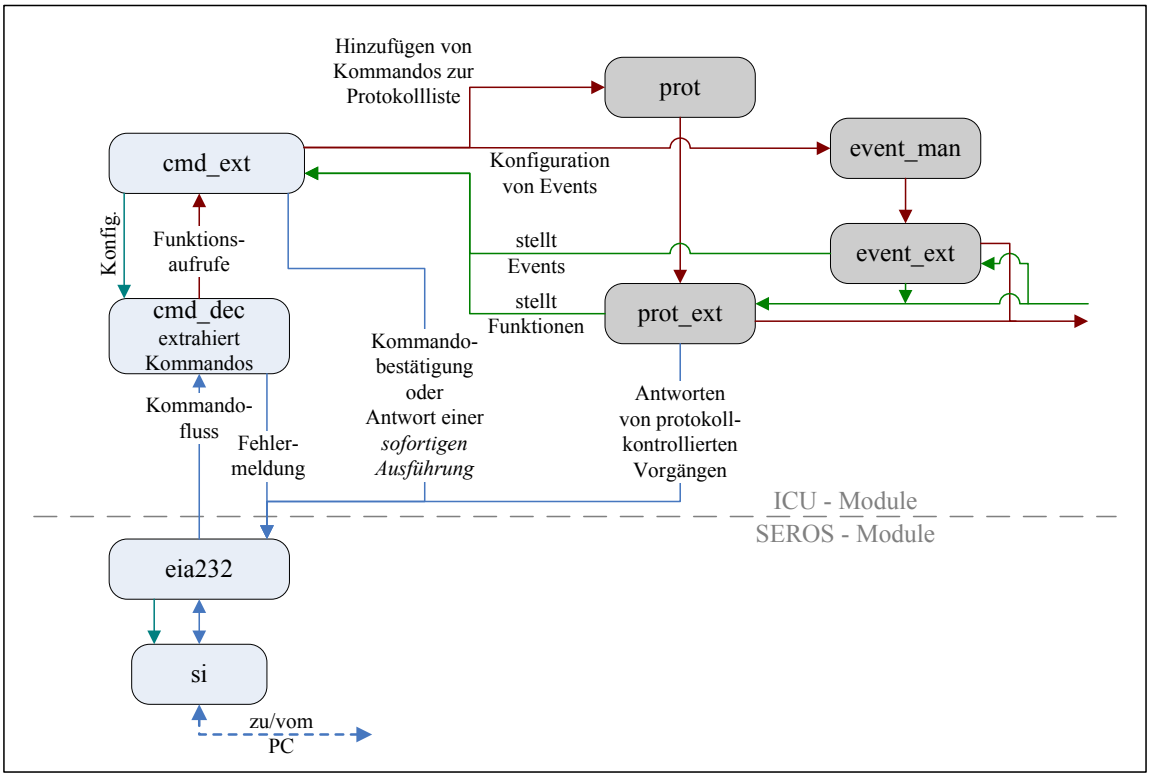


Abbildung 5.12: Blockdiagramm der für die Definition der Ablaufsteuerung zuständigen Module und deren Verknüpfung: Vom PC über die serielle EIA-232 Verbindung geschickte Kommandos werden aus dem Datenfluss extrahiert (dekodiert) und ausgeführt. Dabei werden Kommandos u. a. in die Protokollliste eingetragen und zu spezifizierten Zeiten ausgeführt.

die Kommandos zum Verwalten von Events. Vergleichbar mit dem Hinzufügen von Kommandos in die Protokollliste, werden bei der Definition von Events Funktionen, die vom Modul *event_ext* (vgl. Abschnitt 5.5.1) zur Verfügung gestellt werden, mitsamt den beim Funktionsaufruf notwendigen Parameterdaten in die Eventliste eingetragen (s. Abbildung 5.12, Modulkette *event_ext* → *cmd_ext* → *event_man*).

5.7 Zeitplanung und interne Ablaufkontrolle

Die beschriebenen Funktionsblöcke der ICU Hartware-Schnittstellen, der Kommandodekodierung und der Ausführung von vordefinierten Protokollen wurden nach den Anforderungen an die Firmware modular und somit komplett voneinander entkoppelt konzipiert und implementiert. Somit können die einzelnen Module in weiteren Firmware-Projekten wiederverwertet werden. Zudem kann die Reihenfolge und die Priorisierung der Ausführung einzelner Modulteile individuell auf die Bedeutung der einzelnen Funktionsblöcke im Gesamtkontext der Firmware angepasst werden. Hierfür bedarf es allerdings eines Mechanismus, der die Koordination der firmwareinternen Abläufe entsprechend einer festgelegten Gewichtung übernimmt. Dieser Mechanismus (im Folgenden Aufgabenplaner genannt) muss zum einen die strikte Einhaltung der Planung zeitkritischer Funktionen gewährleisten, und er muss zum anderen die Verteilung der übrig bleibenden zeitlichen Ressourcen abhängig von der Priorisierung übernehmen. Da der Aufgabenplaner nicht zur Funktionalität der Firmware beiträgt, sondern diese lediglich koordiniert, sollte er selbst möglichst wenig Rechenzeit in Anspruch nehmen. Im Rahmen des in

Abschnitt 5.1 erwähnten Gemeinschaftsprojekts ist ein Aufgabenplaner entstanden, der auf Basis der Interrupt-Struktur des verwendeten DSPs eine zyklische, zeitliche Anordnung von bis zu drei Aufgaben auf unterschiedlichen Prioritätsebenen ermöglicht (Modul *task_man*). Die Aufgabe mit der höchsten Priorität wird dabei streng zyklisch abgearbeitet. Die Abarbeitung der Aufgabe mit der zweithöchsten Priorität erfolgt quasi zyklisch, d. h. im Mittel genau so oft wie die Aufgabe mit der höchsten Priorität. Die verbleibende Rechenzeit wird der Aufgabe mit der niedrigsten Priorität zugewiesen.

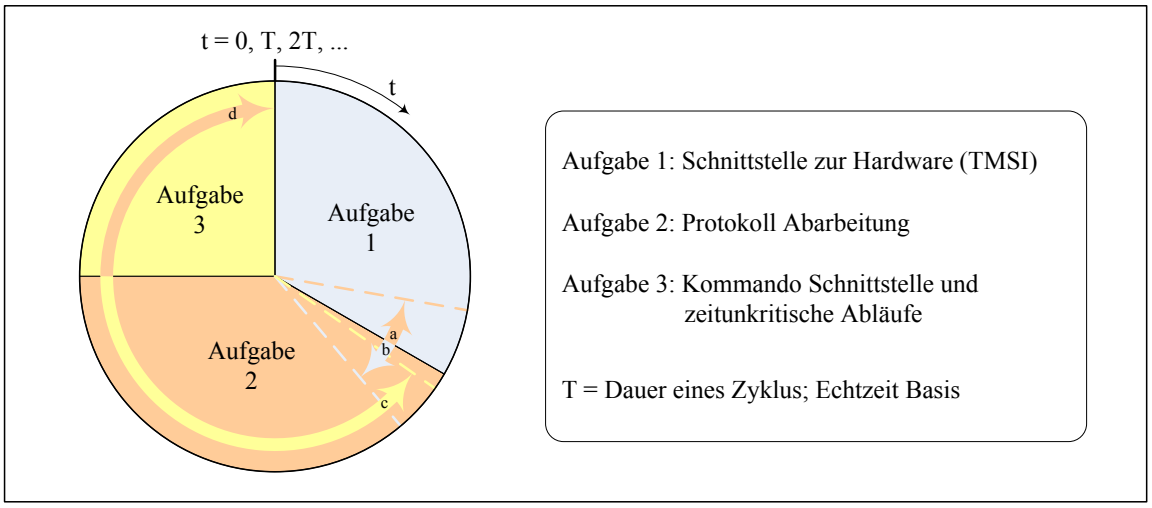


Abbildung 5.13: Zeitscheibe zur Veranschaulichung der Verteilung der zeitlichen Ressourcen durch den Aufgabenplaner: Die strikt zyklische Aufgabe 1 hat im Fall der ICU einen relativ konstanten Zeitbedarf. Die Dauer der Abarbeitung von Aufgabe 2 variiert dagegen stark und kann im äußersten Fall zusammen mit Aufgabe 1 den kompletten Zyklus beanspruchen. Verbleibende Rechenzeit wird Aufgabe 3 zugewiesen.

Anhand der Zeitscheibe in Abbildung 5.13 wird die Verteilung der zeitlichen Ressourcen durch den Aufgabenplaner veranschaulicht. Die strikt zyklische Aufgabe (Abbildung 5.13 - Aufgabe 1) ist die Datenübertragung zwischen dem ICU-DSP und allen Schnittstellenbausteinen über das TMSI (s. Abschnitt 5.2). Aufgabe 1 besteht ausschließlich aus der Abarbeitung der in Abschnitt 5.1.1 beschriebenen Interrupt-Routine, die das Empfangs-Paket des letzten Frames verarbeitet und das Sende-Paket für die Übertragung innerhalb des nächsten Frames vorbereitet. Die quasi zyklische Aufgabe (Abbildung 5.13 - Aufgabe 2) beschreibt die Ausführung der Protokollliste (s. Abschnitt 5.3.1). Aufgabe 2 besteht aus der Überprüfung der durch den Parameter Ausführzeitpunkt angegeben Ausführbedingung und der ggf. darauf folgenden Abarbeitung der Protokolleinträge (s. Abbildung 5.7). Die generelle Implementierung des Aufgabenplaners sieht vor, dass die quasi zyklische Aufgabe lediglich im Mittel genau so oft abgearbeitet wird, wie die strikt zyklische Aufgabe. Im Fall der ICU-Firmware würde dies bedeuten, dass die komplette Abarbeitung von Protokolleinträgen *und* die Übermittlung der Daten zu bzw. von den Schnittstellenbausteinen nur *im Mittel* und nicht entsprechend dem Begriff der Echtzeit *garantiert* innerhalb eines Zyklus erfolgt. Die Echtzeitfähigkeit wird allerdings automatisch erreicht, wenn der maximale Zeitbedarf von Aufgabe 1 und Aufgabe 2 in Summe nicht die Zykluszeit, und damit die Basis der Echtzeit, überschreitet. Die Dauer der Abarbeitung von Aufgabe 1, im Fall der ICU also der Interrupt-Routine zum Bedienen der Hardwareschnittstellen (vgl. Abbildung 5.2), ist aufgrund ihres Funktionsaufbaus mit einer Abweichung von weniger als

5 % relativ konstant (in Abbildung 5.13 wird die Abweichung durch die Pfeile *a* und *b* symbolisiert). Die Dauer der Abarbeitung von Aufgabe 2, also der Ausführung von Protokolleinträgen, kann abhängig von der Überprüfung der Ausführbedingung und von der Anzahl der auszuführenden Protokollkommandos stark variieren. Ist z. B. die Ausführung eines Kommandos an einen externen Trigger gebunden, so würde die über den Parameter Ausführzeitpunkt spezifizierte Überprüfung des entsprechenden Events bei negativem Ergebnis *nicht* zur Ausführen des Kommandos führen. In diesem Fall ist die komplette Aufgabe 2 mit der Event-Abfrage abgeschlossen und würde nahezu keine Rechenzeit in Anspruch nehmen. In Abbildung 5.13 wird dies durch den Pfeil *c* symbolisiert. Ein positives Ergebnis der Event-Abfrage führt hingegen zur Ausführung des Kommandos. Zudem würden durch den Ausführmechanismus des Protokolls (vgl. Abbildung 5.7) alle folgenden Kommandos, deren Parameter Ausführzeitpunkt durch den Wert 0 keine Verzögerung angibt, ebenfalls abgearbeitet. Die maximale Dauer von Aufgabe 2 ergibt sich aus dem ungünstigsten Fall, in dem innerhalb eines Zyklus alle Schnittstellen einmal beschrieben bzw. ausgelesen würden. In diesem Fall würde Aufgabe 3 demnach für diesen Zyklus keine Rechenzeit zugeteilt, wie in Abbildung 5.13 durch Pfeil *d* symbolisiert wird.

Als die Aufgabe mit der niedrigsten Priorität wird die nach der Erledigung der Aufgaben 1 und 2 verbleibende Rechenzeit des DSPs an Aufgabe 3 zugeteilt. Primär wird diese zur Dekodierung und Ausführung der vom PC an die ICU gesendeten Kommandos genutzt. Um darüber hinaus verfügbare Rechenzeit des DSPs zu nutzen, wurde der Aufgabenplaner speziell für die ICU-Anwendung so erweitert, dass auf der niedrigsten Prioritätsebene durch eine Pseudopriorisierung die Einbindung weiterer Aufgaben möglich ist. Dies geschieht über die Eintragung von Funktionsverweisen in eine Liste, die in der zur Verfügung stehenden Rechenzeit sequenziell abgearbeitet wird. Zum Ende des Zyklus wird diese Abarbeitung durch den Aufgabenplaner unterbrochen, der mit der Ausführung von Aufgabe 1 den neuen Zyklus beginnt. Bei der nächsten Aktivierung von Aufgabe 3 wird nach Beenden der im letzen Zyklus unterbrochenen Funktion wieder am Anfang der Prioritätsliste mit der Ausführung fortgefahren. Mit der höchsten Pseudopriorität versehen, wird so weiterhin primär die Kommandoschnittstelle bedient. Durch entsprechenden Eintrag in die sequenzielle Liste können theoretisch beliebig viele zusätzliche Aufgaben in Form von Funktionsaufrufen abgearbeitet werden. Die derzeit implementierten Aufgaben sind von den hardwareunabhängigen Strukturen und Funktionen des Aufgabenplaners getrennt in einem separatem Modul zusammengefasst (Modul *task_ext*).

Beispiele für derzeit bereits implementierte Aufgaben konzentrieren sich auf die verschiedenen Konfigurationsmöglichkeiten der ICU-eigenen EIA-232-Schnittstellen. Wie in Abschnitt 5.2.1 beschrieben wurde, kann das Verfahren zur Behandlung der von angeschlossenen Endgeräten gesendeten Rückmeldungen individuell konfiguriert werden. Abhängig vom eingestellten Modus werden bei der Konfigurierung der Schnittstelle automatisch Aufgaben in die oben beschriebene Liste für pseudopriorisierte Abarbeitung eingetragen. Der Modus wird z. B. zur automatischen Rücksendung einer Antwort, dessen Ende durch ein bestimmtes Zeichen markiert ist, durch eine solche Aufgabe realisiert: Vom Endgerät gesendeten Zeichen werden dabei innerhalb dieser Aufgabe automatisch auf Übereinstimmung mit dem das Ende markierende Zeichen überprüft. Diese Aufgabe wird ebenfalls zur Realisierung des in Abschnitt 5.5.1 beschriebenen EIA-232 Events eingesetzt.

Das Blockdiagramm in Abbildung 5.14 veranschaulicht abschließend noch einmal die Zu-

sammenhänge des Aufgabenplaners (Modul *task_ext*), der Bedienung der Hardwareschnittstellen (Modul *tmsi*), der Protokollausführung (Modul *prot*), der Kommandoschnittstelle (Modul *cmd_dec*) und der die restliche Rechenzeit ausnutzenden Aufgaben (Modul *task_ext*).

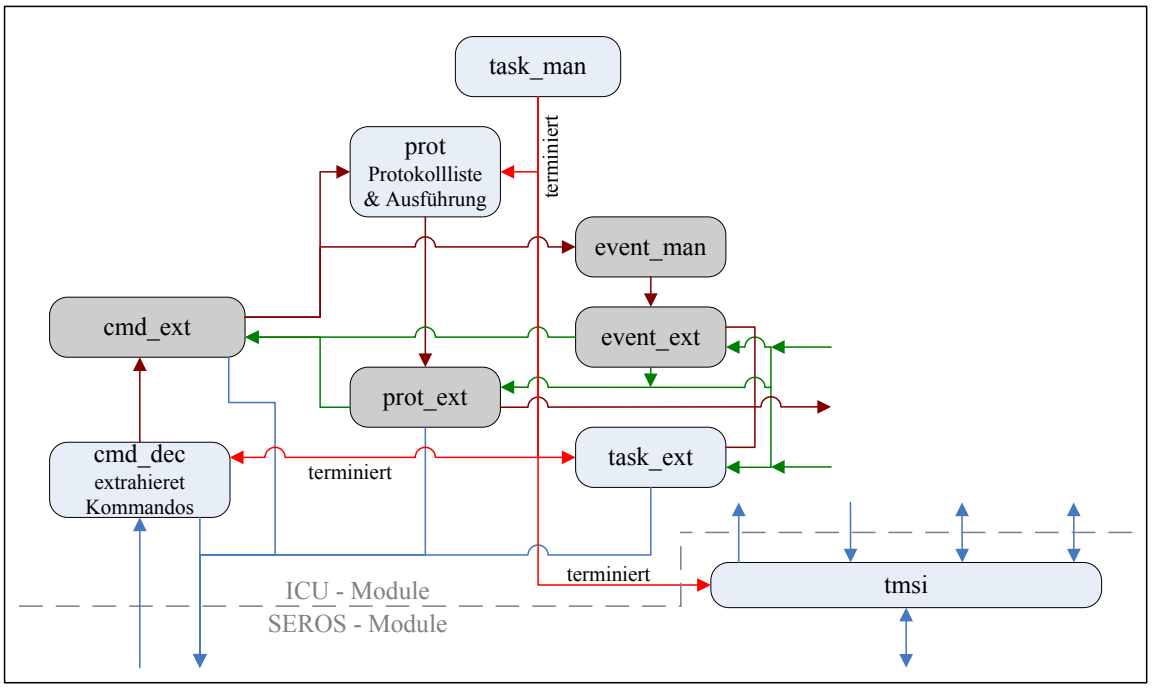


Abbildung 5.14: Aufgabenplaner der ICU: Die firmwareinternen Funktionsblöcke der Hardwareschnittstellen und der Protokollausführung - symbolisiert durch die Module *tmsi* und *prot* - werden durch den Aufgabenplaner (Modul *task_man*) anhand einer Priorisierung zyklisch abgearbeitet. In der hiernach verbleibenden Rechenzeit werden die Kommandoschnittstelle sowie benutzerdefinierte Aufgaben bedient - Module *cmd_dec* und *task_ext*.

5.8 Gesamtüberblick - Firmwaremodule und deren Verknüpfung

In diesem Kapitel wurden die verschiedenen Funktionsblöcke der ICU-Firmware detailliert vorgestellt. Diese setzen sich aus Modulen des rudimentären Betriebssystems SEROS sowie der darauf aufsetzenden Anwendung zusammen. Die logischen Zusammenhänge zwischen den einzelnen Modulen wurden für jeden Funktionsblock durch jeweils ein Blockdiagramm veranschaulicht. Abbildung 5.15 zeigt nun in einer Gesamtübersicht alle vorgestellten Module und deren Verknüpfung untereinander in einem ganzheitlichen Blockdiagramm. Der neu hinzugekommene Block des Moduls *icu_main* beschreibt die Haupt-Routine, die nach Booten des DSPs zunächst alle Module initialisiert bzw. konfiguriert. Im Anschluss daran ist die ICU durch Starten des Aufgabenplaners einsatzbereit.

Durch die strikte Trennung von allgemeinen und ICU-anwendungsspezifischen Funktionen in separate Module ergibt sich zum Einen die Möglichkeit der Verwendung der hardwareunabhängigen Module in weiteren Firmwareprojekten. Zum Anderen wird eine flexible Erweiterung der Funktionalität z. B. über einfaches Einbinden neuer Hardwarekomponenten ermöglicht. Zudem wurden alle Strukturen wie z. B. die Protokoll-, Variablen- oder Event-Listen flexibel angelegt, so dass i. d. R. über die Änderung von einer Konstanten bzw. Variablen z. B. die Listenlänge an veränderte Anforderungen angepasst werden kann. Würde ein

solcher Änderungsparameter in die Konfigurationsdatei der ICU (s. Abschnitt 5.9) aufgenommen, würde die Änderung durch Anpassen des Parameters und einen Neustart der Firmware unmittelbar gültig.

Das Blockdiagramms in Abbildung 5.15 zeigt auf den ersten Blick, dass die beschriebene Firmware aus vielen logischen Ebenen aufgebaut ist. Dies ist notwendig, um die geforderte Modularität und Erweiterbarkeit sowie die logische Trennung von Funktionen wie z. B. unmittelbarer und protokollbasierter Ausführung von Kommandos zu gewährleisten. Die Verbindung zwischen zwei übereinander angeordneten logischen Ebenen erfolgt über Funktionsaufrufe. Soll eine Verbindung über mehrere Ebenen erstellt werden, geschieht dies über entsprechend viele, geschachtelte Funktionsaufrufe. Da jeder dieser Aufrufe Rechenzeit benötigt, wirkt eine vielfache Schachtelung wie ein Multiplikator. Um die damit verbundenen Rechenzeitverluste zu minimieren, wurde - wenn möglich - mittels Durchreichen von Funktionsverweisen und deren Verwendung bei entsprechenden Aufrufen eine Überbrückung von Zwischenebenen erreicht.

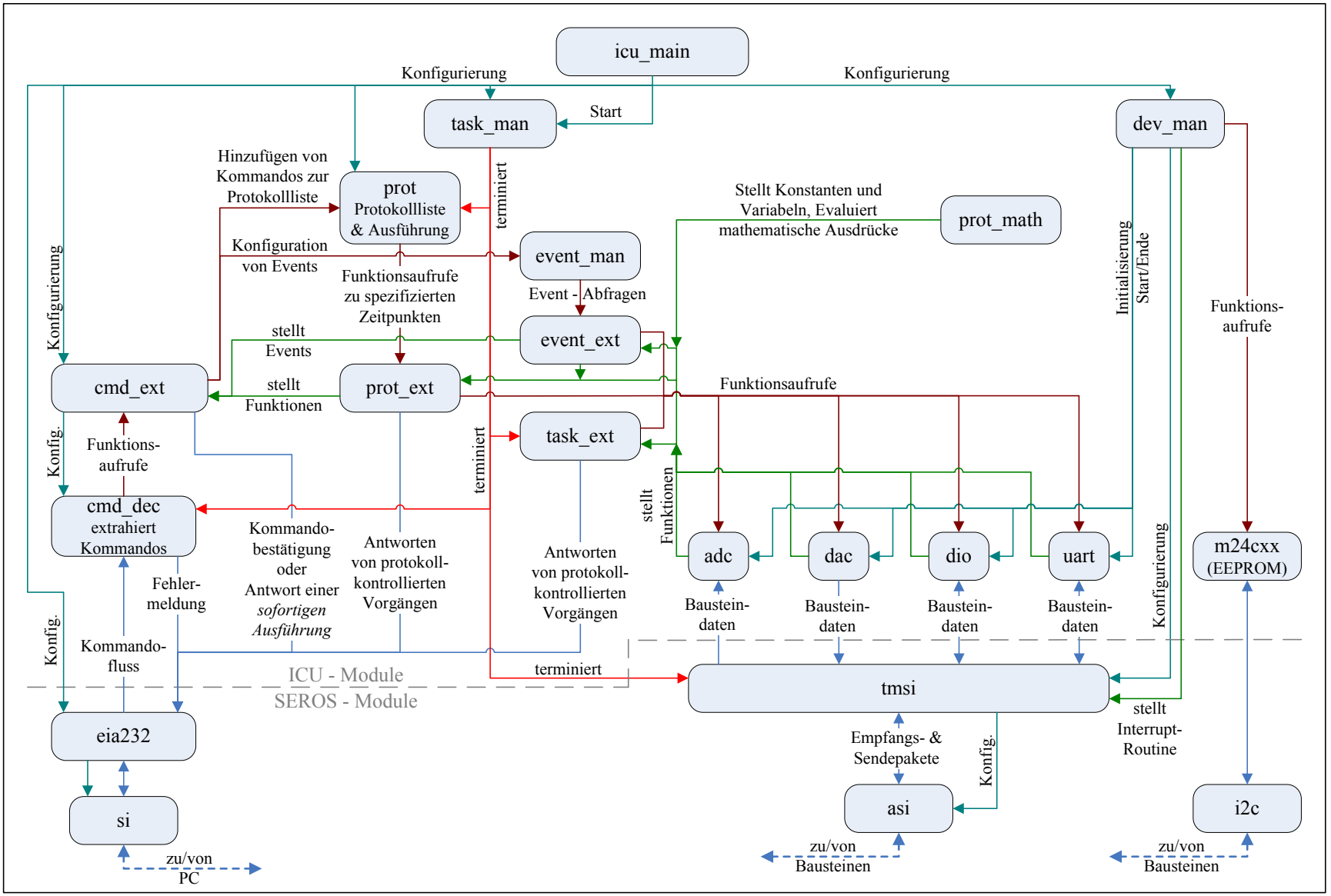


Abbildung 5.15: Zusammenfassung der Haupt-Firmwaremodule und deren Verknüpfung

5.9 Kommandoschnittstelle zur ICU

Die in Abschnitt 5.6.1 beschriebene Schnittstelle zum PC ermöglicht dem Benutzer den Zugang zu der in diesem Kapitel ausführlich vorgestellten ICU-Firmware, der hierüber zugänglich gemachten Funktionalität sowie den damit verbundenen Möglichkeiten der flexiblen Gestaltung von protokollbasierten Abläufen. Eine detaillierte Beschreibung des Aufbaus *aller* Kommandos dieser Schnittstelle ist für das Verständnis der ICU-Funktionalität im Einzelnen nicht notwendig. Eine kurze Beschreibung der implementierten Kommandos und deren Funktion stellt diese Kommandoschnittstelle im Folgenden allerdings grob vor. Eine umfangreiche Beschreibung aller ICU-Kommandos mitsamt der im Einzelnen notwendigen Parameter sowie Informationen über das Ausführverhalten und über zu erwartende Rückantworten bietet die Dokumentation „ICU v1.0 Firmware Interface Description“, die im Anhang dieser Arbeit abgedruckt ist (s. Seite 147 ff.). Die Menge der Kommandos wird im Folgenden in einen Block von administrativen bzw. Service-Kommandos und in einen Block protokollfähiger Kommandos unterteilt. Die in den folgenden Auflistungen angegebenen hexadezimalen Werte entsprechen der Kommandonummer, welche die einzelnen Kommandos eindeutig identifiziert (vgl. Abbildung 5.10 auf Seite 90).

Kommandos, die *nicht* für die Verwendung innerhalb von Protokolldefinitionen vorgesehen sind:

- $0x00$ - *NoOp*: Dieses Kommando hat buchstäblich keine Funktion. In Protokollen eingesetzt, kann es allerdings z. B. verwendet werden, um die Abarbeitung nachfolgender Kommandos länger als die maximal durch den Parameter Ausführzeitpunkt (0xFFFFFFCF ≈ 11 h: 56 min) einstellbare Verzögerung hinauszuschieben. Zudem kann über dieses Kommando die Kommunikation zwischen Anwender-PC und der ICU überprüft werden, da - wie bei allen Befehlen, die in Ihrer Antwort keine Daten übertragen - eine Empfangsbestätigung an den PC gesendet wird.
- $0x01$ - *WriteSystemConfig*: Über dieses Kommando können verschiedene Systemparameter der ICU gesetzt werden. Abhängig von der Art des jeweiligen Parameters, wird dieser entweder permanent im Flash-Speicher der DSP-Platine oder vergänglich, bis zum nächsten Start der Firmware, im Arbeitsspeicher des DSPs abgelegt. Folgende Parameter können in der aktuellen Version der Firmware verändert werden:
  - Seriennummer,
  - Hardware Konfiguration,
  - Kennzeichnung: wurde das System bereits verwendet (ja/nein)
  - Schalter: Kommando Dekodierung verwendet einen Timeout zwischen zwei zu dekodierenden Bytes (ja/nein),
  - Version der CPLD-Programmierung,
  - Schalter: Sende Startup Warnung (ja/nein).
- $0x02$ - *ReadSystemConfig*: Die soeben beschriebenen Parameter sowie zusätzliche, automatisch durch die Firmware generierte Parameter können über das Kommando „ReadSystemConfig“ ausgelesen werden. Die zusätzlichen Parameter sind:
  - Version der Firmware,
  - Version der Kommandoliste,

  - Datum der Firmwareerstellung (nur für Testversionen),
  - Version des SEROS.

- $0x07$ - *HardwareReset*: Durch Senden dieses Befehls kann firmwareseitig ein Hardware-Reset der DSP-Platine und hierüber aller elektronischen Bauteile innerhalb der ICU ausgelöst werden.
- $0x4D$ - *ConfigEvent*: Konfiguriert ein Event, wie in Abschnitt 5.5 beschrieben.
- $0x4E$ - *FreeEvent*: Gibt den durch eine Event-Konfiguration belegten Speicherplatz in der Event-Liste frei und ermöglicht damit eine Neukonfiguration (s. Abschnitt 5.5).
- $0x4F$ - *CancelProtocol*: Bricht die laufende Abarbeitung eines Protokolls ab und setzt die gesamte Protokollstruktur durch Löschen aller Einträge zurück.
- $0x50$ - *RecordProtocol*: Markiert den Anfang der Definition eines Protokollblocks. Eine laufende Abarbeitung von bereits vorher definierten Protokollblöcken wird hierdurch nicht beeinträchtigt. Die Abarbeitung wird gestoppt, sobald die als Beginn des neuen Protokollblocks markierte Listenposition erreicht wird (s. Abschnitt 5.3.2).
- $0x51$ - *EndRecordProtocol*: Markiert das Ende der Definition eines Protokollblocks und führt gleichzeitig zum Starten der Ausführung. In Verbindung mit dem Kommando „RecordProtocol“ ist somit überlappendes Definieren von Protokollen während gleichzeitiger Abarbeitung eines bereits vorher definierten Protokolls möglich (s. Abschnitt 5.3.2).

Kommandos die für die Verwendung innerhalb von Protokolldefinition vorgesehen sind:

- $0x52$ - *Loop*: Setzt den Anfang einer Schleife.
- $0x53$ - *EndLoop*: Setzt das Ende einer Schleife.
- $0x54$ - *If*: Leitet eine bedingungsabhängige, verzweigte Ausführung von Protokollblöcken ein.
- $0x55$ - *Else*: Markiert den Anfang eines alternativ auszuführenden Protokollblocks einer verzweigten Ausführung
- $0x56$ - *EndIf*: Markiert das Ende einer verzweigten Ausführung
- $0x57$ - *SendResponse*: Über den Einsatz dieses Befehls in Protokollen können beliebige, durch den Anwender bestimmte Rückmeldungen zu im Protokoll festgelegten Zeiten an den PC gesendet werden. Dies kann z. B. zur Synchronisation der PC-Software und den durch die ICU gesteuerten Echtzeitabläufen verwendet werden.
- $0x58$ - *ResetExpTimer*: Eine firmwareinterne „Stoppuhr“ kann über dieses Kommando zurückgesetzt werden. Das Kommando liefert den zu diesem Zeitpunkt aktuellen Wert als Antwort an den PC. Verwendet innerhalb von Protokollen kann somit z. B. die durch Warten auf ein externes Signal vergangene Zeit ermittelt und zur zeitlichen Rekonstruktion der Protokollabläufe verwendet werden.
- $0x59$ - *SendExpTimerValue*: Ähnlich der Beschreibung des vorangegangen Kommandos wird der zur Ausführzeit aktuelle Wert der firmwareinternen „Stoppuhr“ als Antwort an den PC gesendet, allerdings ohne die „Stoppuhr“ dabei zurückzusetzen.
- $0x5A$ - *SoftEvent*: Löst ein Soft-Event aus (s. Abschnitt 5.5.1)
- $0x5B$ - *DeclareVariable*: Deklariert eine Variable nach der in Abschnitt 5.4 beschriebenen Methode.

- $0x5C$ - *FreeVariable*: Gibt den Listeneintrag einer vorher deklarierten Variable frei und ermöglicht so eine Neudeklarierung (s. Abschnitt 5.4).
- $0x5D$ - *WriteVariable*: Weist einer vorher deklarierten Variablen einen Wert zu (s. Abschnitt 5.4).
- $0x5E$ - *ReadVariable*: Liest den Wert einer vorher deklarierten Variablen aus (s. Abschnitt 5.4) und sendet ihn an den PC.
- $0x60$ - *ConfigUART*: Konfiguriert die angegebene ICU EIA-232-Schnittstelle (s. Abschnitt 5.2)
- $0x61$ - *WriteUART*: Sendet einen Datenblock an ein über eine der EIA-232-Schnittstellen angeschlossenes Endgerät (s. Abschnitt 5.2).
- $0x62$ - *ReadUART*: Liest entsprechend dem konfigurierten Modus einen Datenblock von einem über eine der EIA-232-Schnittstellen angeschlossenes Endgerät (s. Abschnitt 5.2).
- $0x63$ - *WriteAnalogOut*: Setzt den spezifizierten analogen Ausgang der ICU auf den angegebenen Wert (s. Abschnitt 5.2). Allerdings ist bei der Verwendung in einem Protokoll zu beachten, dass je Zyklus (TMSI-Frame) nur der Wert eines Analogkanals geändert werden kann. Dies ist darin begründet, dass die Analogkanäle über einen analog-zu-digital-Wandler mit nachgeschaltetem Multiplexer realisiert sind. Das Ändern der Werte von mehreren analogen Ausgängen mit einer angegebenen Ausführverzögerung von null (Parameter „Ausführzeitpunkt“ = 0) wird durch die ICU-Firmware auf entsprechend viele Zyklen ausgedehnt.
- $0x64$ - *ReadAnalogOut*: Liest den Wert des spezifizierten analogen Ausgangs der ICU und sendet ihn an den PC (s. Abschnitt 5.2). Dieses Kommando hat somit lediglich Kontrollfunktion (s. Abschnitt 5.2.1).
- $0x65$ - *ReadAnalogIn*: Liest den Wert des spezifizierten analogen Eingangs der ICU und sendet ihn an den PC (s. Abschnitt 5.2).
- $0x66$ - *ConfigDigitalIO*: Konfigurierbare digitale I/O können über diesen Befehl bankweise entweder als Ein- oder als Ausgänge konfiguriert werden (s. Abschnitt 5.2).
- $0x66$ - *WriteDigitalOut*: Setzt einen oder mehrere digitale Ausgänge einer Bank (s. Abschnitt 5.2).
- $0x67$ - *ReadDigitalIn*: Liest einen oder mehrere digitale Eingänge einer Bank und schickt diese an den PC (s. Abschnitt 5.2).
- $0x70$ - *MathExpression*: Führt die Evaluierung eines mathematischen Ausdrucks aus (vgl. Abschnitt 5.4).
- $0x71$ - *MathExpressionDurationCheck*: Misst die für die Evaluierung eines mathematischen Ausdrucks benötigte Ausführdauer und sendet diese an den PC. Dieses Kommando dient zur Zeitplanung im Verlauf von Protokolldefinitionen auf PC-Software Ebene.

5.10 Testen der ICU-Firmware

Zum Testen der Schnittstelle zwischen ICU und PC, dem Verfahren der Kommandodekodierung sowie der sofortigen und protokollbasierten Kommandoausführung wurde eine PC-Testumgebung entwickelt: ICU_TestingControl. Basierend auf der in Microsoft® Office Excel 2003 eingebetteten Programmierumgebung **V**isual **B**asic for **A**pplications (VBA) wurden Funktionen programmiert, die den kompletten Umfang der ICU-Kommandos auf PC-Seite

widerspiegeln. Für die Kommunikation zwischen dem PC und der ICU über eine EIA-232-Schnittstelle wird eine von Dr. Ulrich Ruhnau (ehemals TILL I.D.) programmierte **D**ynamic **L**ink **L**ibrary (DLL) verwendet. Diese DLL stellt neben dem Zugriff auf die physikalische Schnittstelle einfache Funktionen zum Erstellen von Kommandos nach der in Abschnitt 5.6.2 beschriebenen Kommandostruktur bereit. Generell werden Kommandoparameter, wie z. B. der Ausführzeitpunkt, die Kanalnummer eines analogen Ausgangs und der zu setzende Wert, in entsprechend hierfür vorgesehene Zellen des Arbeitsblatts eingetragen. Über das Betätigen einer Befehlsschaltfläche wird nach den Vorgaben der Parameter das entsprechende Kommando zusammengesetzt und an die ICU gesendet. Die Excel-basierte Benutzeroberfläche der Testumgebung besteht aus mehreren Arbeitsblättern, die jeweils einen Aspekt der Testumgebung zusammenfassen. Das Arbeitsblatt mit dem Titel „Basic ICU“ bietet Eingabemasken zum individuellen Versenden aller protokollrelevanten Kommandos zum Bedienen der ICU-Hardware sowie der Protokollfunktionen. Hierüber wurden zum einen die Kommandoschnittstelle und die korrekte Funktion jedes einzelnen Kommandos überprüft und zum anderen einfache Testprotokolle aufgesetzt.

Für komplexere und wiederkehrende Protokolldefinitionen besteht zudem die Möglichkeit, ein VBA-Script zu erstellen, welches über das Betätigen einer Befehlsschaltfläche ausgeführt wird und damit die darin enthaltenen Kommandos an die ICU sendet (Arbeitsblatt „VBA-Script“). Innerhalb eines VBA-Scripts können alle ICU-Kommandos über Funktionsaufrufe eingebunden werden. Vom Kommando benötigte Parameter werden dabei beim Funktionsaufruf übergeben. Die Script-Funktionalität wurde u. a. zum Testen des zeitlichen Verhaltens der Ausführung von Protokollen verwendet.

Auf einem weiteren Arbeitsblatt wurde ähnlich dem Aufbau des Arbeitsblatts „Basic ICU“ die Möglichkeit geschaffen, die Systemkonfiguration auszulesen bzw. veränderliche Elemente zu beschreiben. Neben der Überprüfung der Kommandos *ReadSystemConfig* und *WriteSystemConfig* (s. Seite 99) und den mit Lesen und Schreiben verbundenen Operationen wie z. B. dem Zugriff auf den Flash-Speicher kann dieser Teil der Testumgebung zudem innerhalb der Produktion zum Setzen von Parametern wie der Seriennummer eingesetzt werden.

5.11 Firmware auf weiteren ICU-Einschüben und deren Vernetzung

Neben der Firmware für den DSP-Einschub der ICU wird am BIZ an weiteren Firmwareprojekten gearbeitet: Firmware für die Digitale Scanner Kontrollsteuerung (DSC) und das **A**nalog-**S**ignal **A**cquisition **M**odule (ASAM). Da innerhalb dieser Projekte die gleiche Prozessorarchitektur (eine DSP-Platine der Firma SmartMove) als Hardwarebasis verwendet wird, bauen die jeweiligen Firmwareanwendungen ebenfalls auf dem in Abschnitt 5.1.1 vorgestellten Betriebssystem SEROS auf. Zudem kommen einzelne, anwendungsunabhängige Module, die im Rahmen der Beschreibungen der ICU-Firmware vorgestellten wurden, innerhalb dieser Projekte zum Einsatz.

Die Firmware für den in Abschnitt 4.6.2 beschriebenen Einschub DSC setzt als Haupthardwareschnittstelle ebenfalls das TMSI ein. Neben den digitalen Reglern für galvanometrische Motoren, analogen und digitalen Ausgängen zum Steuern von z. B. Laser Combinern und/oder zur Synchronisierung von Detektionsgeräten soll das TMSI mittelfristig auch als Schnittstelle zwischen den DSPs verwendet werden. Die aktuelle Konfiguration des TMSIs der DSC und die

Zuordnung der Time-Slots zu den einzelnen Geräten (Slaves) ist in Abbildung 5.16 dargestellt.

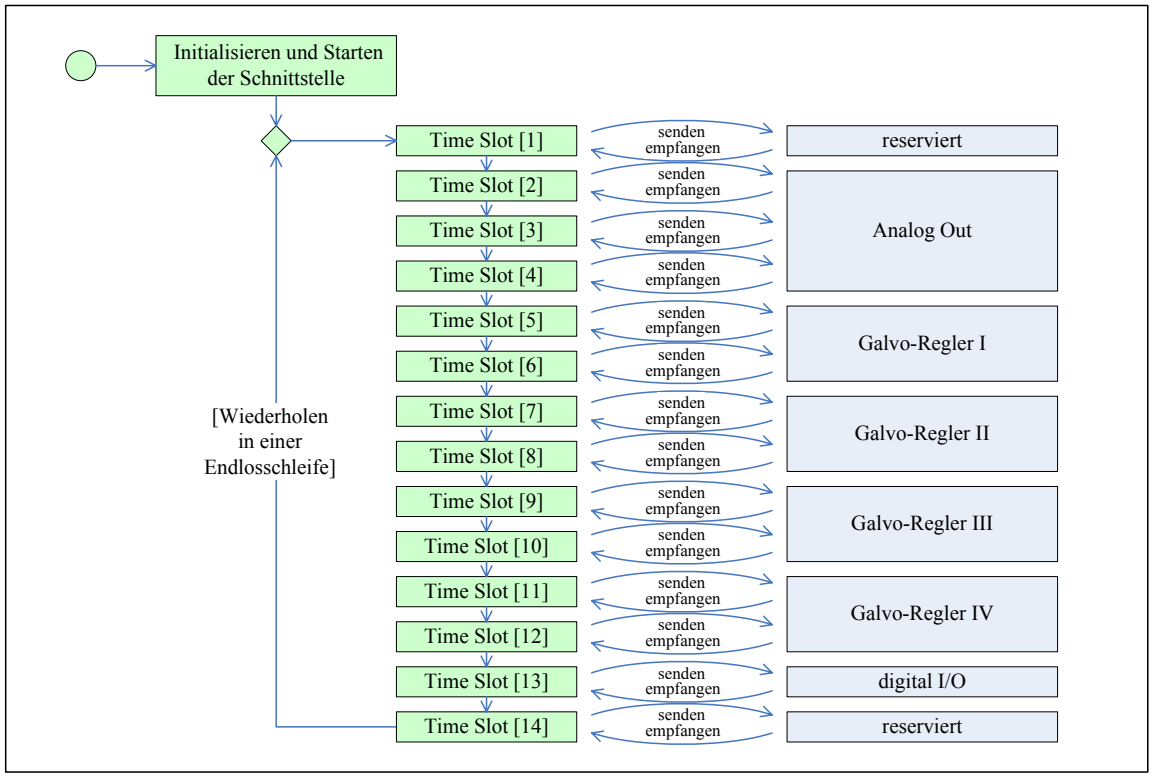


Abbildung 5.16: Flussdiagramm des DSC TMSIs mit Zuordnung der Time-Slots zu den entsprechenden Geräten

Am Beispiel der Zuordnung der Time-Slots im Fall der DSC ist der Time-Slot mit der Nummer Eins für die geplante Verknüpfung der DSPs untereinander reserviert. Durch diese Verknüpfung der auf den verschiedenen Einschüben der ICU eingesetzten DSPs besteht zum einen die Möglichkeit des Austauschs von Daten bzw. Kommandos. Zum anderen ist über diese dauerhaft betriebene, serielle Schnittstelle eine Synchronisation aller DSPs auf einen Master hin möglich. Die Genauigkeit dieser Synchronisation beläuft sich auf die Zykluszeit (Dauer eines Frames) der als Master konfigurierten TMSI.

6 PC-Software-Konzept

Innerhalb der in Kapitel 2 vorgestellten Anwendungsgebiete nimmt der PC nach dem erläuterten Steuerungskonzept (s. Kapitel 3) neben der ICU eine zentrale Rolle ein. Als Eingabeschnittstelle ermöglicht er dem Benutzer u. a. den Zugang zur Systemhardware durch unmittelbare, interaktive Bedienung oder über das Definieren von zeitlich vorherbestimmten Abläufen. Zugleich werden die Rechenleistung und die Speicherkapazität der PCs für die Auswertung bzw. Analyse der durch die Systeme aufgenommenen Daten sowie deren Archivierung verwendet. Die Aspekte Hardware, Firmware und Software als „Komponenten“ von komplexen, computergestützten Systemen sind selbstverständlich unmittelbar miteinander verknüpft. Ohne ein entsprechendes Software-Front-End sind solche Systeme unvollständig und nicht zu bedienen. Zum Komplettieren der Beschreibungen aller Aspekte der vorgestellten Systeme soll das Softwarekonzept an dieser Stelle lediglich kurz vorgestellt werden, da das Erstellen der PC-Software nicht Teil der hier vorgestellten Arbeiten war.

Das generelle Softwarekonzept sieht ähnlich wie das der ICU-Firmware einen modularen Aufbau vor. Dies dient zum einen der internen Strukturierung der Software und bedient zum anderen das stark variierende Kundenklientel der vorgestellten Anwendungsgebiete vom „Bastler-Anwender“, der durch ständig veränderte Versuchsaufbauen möglichst viel Flexibilität benötigt, bis zum reinen Anwender, der schlüsselfertige Lösungen verlangt. Abbildung 6.1 zeigt den groben Aufbau des Softwarekonzepts anhand eines Blockdiagramms.

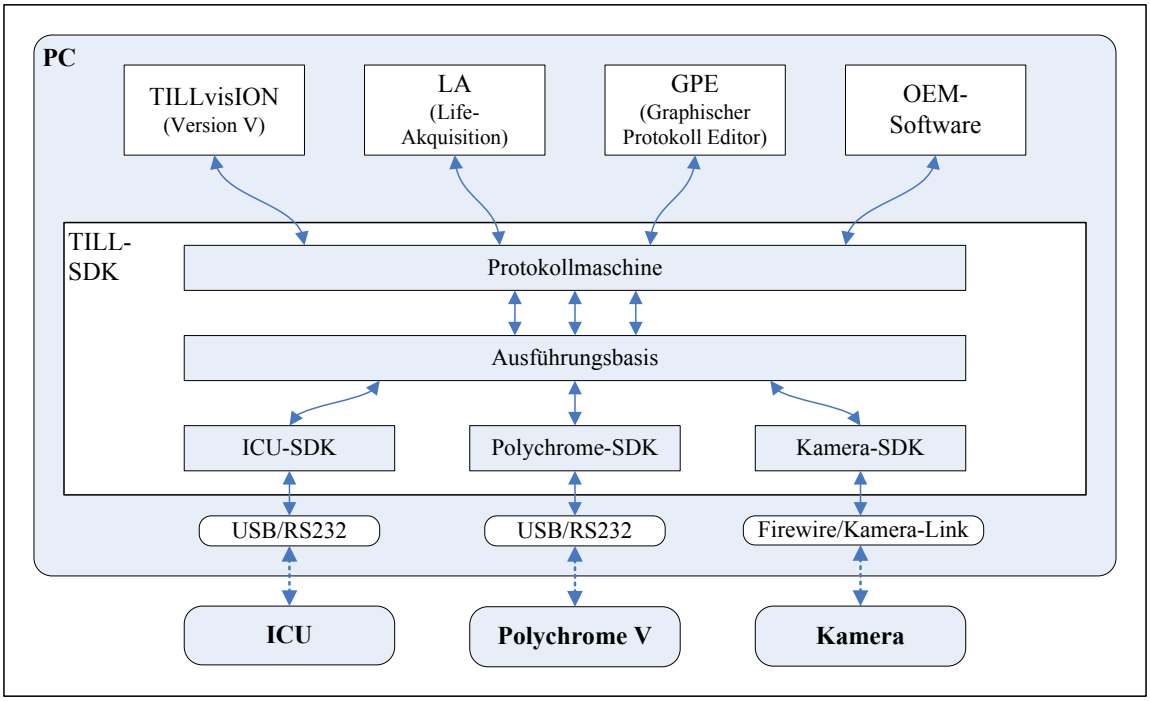


Abbildung 6.1: Softwarekonzept: Jegliche Software wie z. B. TILLvisION V basiert auf dem TILL-SDK. Dieses besteht aus der Protokollmaschine (Validierung, Konkretisierung und Teilung von Protokollen in gerätespezifische Teilprotokolle), der Ausführbasis (Extrahieren und Ausführen von Protokollkommandos) und den konkreten SDKs der integrierten Geräte. Softwaremodule wie z. B. das Modul zur Life-Akquisition oder der GPE können als Komponente in weiterer Software wie z. B. TILLvisION oder OEM-Software eingebunden werden; Informationsquelle: TILL Photonics

Für die Integration der ICU, der Mikroskop-Plattform iMIC und/oder eines Imaging-Systems bestehend aus einer Kamera und der Lichtquelle Polychrome V, soll ein SDK (TILL-SDK) zur Ansteuerung der Hardware zur Verfügung stehen. Bei grober Betrachtung ist dieses SDK in drei verschiedene logische Ebenen unterteilt. Auf der untersten, hardwarenähesten Ebene werden die Systemkomponenten über die Verwendung der Geräte SDKs (ICU-SDK, Polychorme-SDK und Kamera-SDK) und der jeweiligen Hardwareschnittstelle eingebunden. Zudem bieten die Geräte-SDKs durch ein entsprechendes API den Übergang von den aus Optimierungsgründen relativ kryptischen Kommunikationsprotokollen (wie z. B. im Fall der ICU; s. Abschnitt 5.6.2) hin zu einer allgemein verständlichen Form. Hierdurch wird die Funktionalität der jeweiligen Geräte auf eine Sammlung hardwarespezifischer Funktionen abgebildet. Diese werden innerhalb der Programmierung auf der nächsten, übergeordneten Ebene verwendet. Änderungen z. B. in den Kommunikationsprotokollen oder in der Hardwarefunktionalität können somit i. d. R. bereits auf der Ebene der Geräte-SDKs abgefangen werden, ohne das übergeordnete Softwaremodule davon betroffen sind. Beispielsweise würde eine Umstellung der Verbindung zwischen der ICU und dem PC von der derzeit verwendeten EIA-232-Schnittstelle auf USB eine Anpassung des Moduls ICU-SDK erfordern. Alle weiteren Module des TILL-SDKs und der hierauf aufbauenden Softwaremodule wären von diese Umstellung nicht betroffen.

Auf den Geräte-SDKs aufbauend wird die Funktionalität der einzelnen Systemkomponenten auf der direkt übergeordneten logischen Ebene, der „Ausführungsbasis“, zusammengeführt. Dies geschieht ausgehend von hardwarenahen XML (E**x**tensible **M**arkup **L**anguage)-Beschreibungen der Abläufe des Gesamtsystems, in denen gerätespezifische Kommandos bereits in separaten Protokollen getrennt nach den einzelnen Systemkomponenten vorliegen. Im Funktionsblock der Ausführungsbasis werden die Kommandos aus der XML-Beschreibung extrahiert und über den Aufruf der jeweiligen API-Funktionen der Geräte-SDKs ausgeführt. Als einfaches Beispiel sei hier das Ablaufprotokoll der Aufnahme eines Fluoreszenzbildes vorgestellt, das aus Einzelprotokollen mit Kommandos zur Beleuchtung durch die Lichtquelle Polychrome V, der Belichtung durch eine CCD-Kamera und der Synchronisation dieser beiden Prozesse durch die ICU besteht. Die Kommandos werden aus dem entsprechenden Ablaufprotokoll des Zielgerätes extrahiert und führen zum Aufruf der jeweiligen Funktionen der Geräte-SDKs.

Die „Protokollmaschine“ bildet die oberste logische Ebene des TILL-SDKs und liefert die Schnittstelle zu übergeordneter Software. Protokolle werden in Form von abstrakten XML-Beschreibungen an die Protokollmaschine übergeben. Innerhalb dieser Beschreibungen können Kommandos u. a. abstrakte Funktionen des Gesamtsystems repräsentieren, die bei der Ausführung z. B. zu geräteübergreifenden Operationen führen. Komplexe Abläufe werden hierüber zu einfachen Einheiten zusammengefasst. In Anlehnung an das vorangegangene Beispiel würde lediglich ein Eintrag im abstrakten XML-Protokoll die Aufnahme eines Fluoreszenzbildes beschreiben. Mit entsprechenden Angaben zum Ausführzeitpunkt des Kommandos, der Belichtungsdauer und der Anregungswellenlänge ist die Anzahl der benötigten Parameter auf ein Minimum reduziert. Nach der Übergabe einer solchen XML-Beschreibung an die Protokollmaschine überprüft diese die Syntax, löst die abstrakten Protokollkommandos auf, formt sie in konkrete Kommandos um und ordnet sie individuellen XML-Protokollen der verschiedenen Systemkomponenten zu. Zusätzlich validiert die Protokollmaschine die angegebenen Ausführzeiten und kontrolliert etwaige Überschneidungen von restriktiven Abläufen, die nicht gleichzeitig ausgeführt werden dürfen. Auftretende Fehler führen zu einer entsprechenden Fehlermeldung

an die übergeordnete Software. Überdies wird verhindert, dass ungültige Protokolle an die Ausführbasis übergeben und abgearbeitet werden. Abstrakte XML-Beschreibungen, die durch die Protokollmaschine erfolgreich überprüft und in konkrete, gerätespezifische Einzelprotokolle umgeformt wurden, werden für deren Abarbeitung an die Ausführungsbasis übergeben.

Das Ausnutzen der durch Hardwaresynchronisation optimierten, zeitlichen Kontrolle von deterministischen und adaptiven Abläufen wird über Beschreibungen der Vorgänge in Form von Protokollen erlangt. Das Erstellen dieser Protokolle kann zum einen auf der Programmierebene über die soeben beschriebenen XML-Protokolle erfolgen. Zum anderen bietet ein auf dem TILL-SDK aufsetzender graphischer Protokoll Editor (GPE) die Möglichkeit abstrakte XML-Protokolle über ein GUI mittels der Eingabe per PC-Maus zu erstellen, indem Funktionsblöcke auf einer Zeitachse angeordnet werden. Abbildung 6.2 zeigt den schematischen Aufbau der Benutzeroberfläche nach dem Konzeptentwurf des GPE.

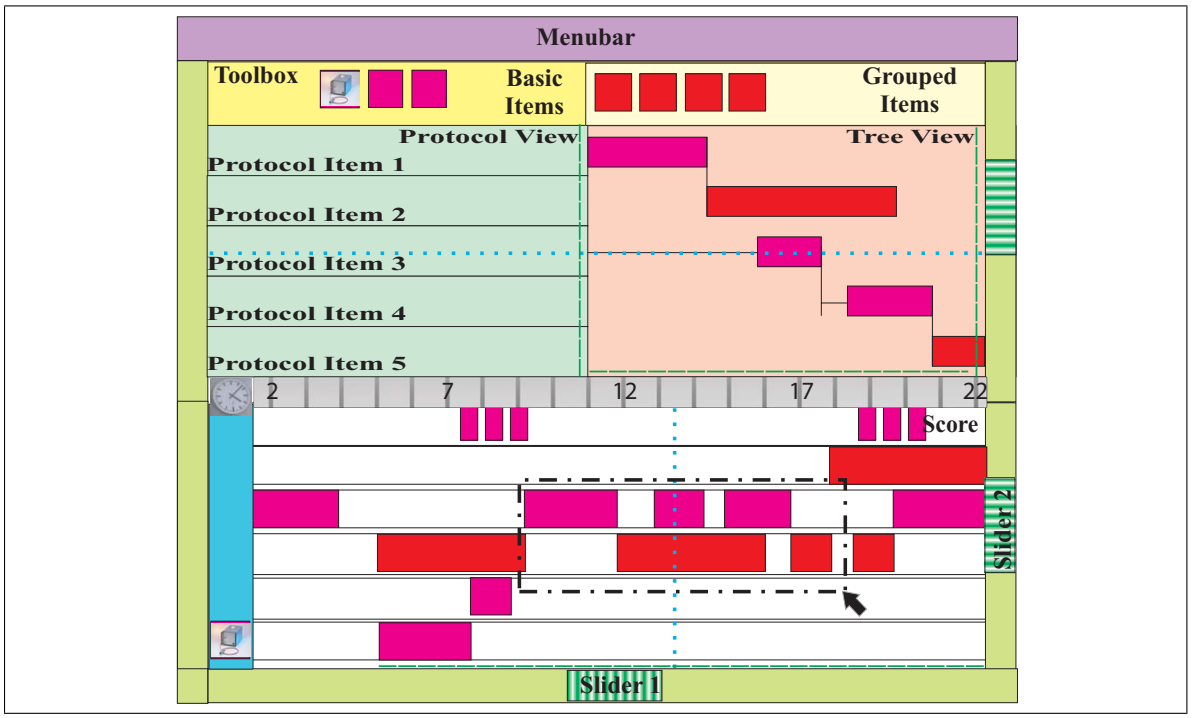


Abbildung 6.2: **G**raphischer **P**rotokoll **E**ditor (GPE) - Konzeptentwurf des Layouts. Der GPE ermöglicht das Erstellen von Protokollen über eine intuitive graphische Benutzeroberfläche. Quelle: TILL Photonics

Die Breite eines Blocks repräsentiert die Dauer der damit verbundenen Funktion. Seine Position in Bezug auf die Zeitachse bestimmt den Ausführzeitpunkt innerhalb des Protokolls. Ähnlich einer Partitur in der Musik ist es möglich mehrere Blöcke parallel anzuordnen, was einer gleichzeitigen Ausführung der durch die Blöcke beschriebenen Funktionen entspricht. Über die Verwendung der Protokollmaschine des TILL-SDKs (s. o.) wird die physikalische Durchführbarkeit der erstellten Protokolle durch eine Simulation vor der eigentlichen Ausführung überprüft. Dies dient zum einen der Gerätesicherheit, indem einander ausschließende Operationen wie z. B. das Fokussieren und gleichzeitiges Wechseln des Objektiv bereits zum Zeitpunkt der Protokollerstellung abgefangen werden. Zum anderen kann durch die Überprüfung der Zeitmodelle eine Optimierung der durch das Protokoll beschriebenen Abläufe erfolgen. Gruppen von Befehlsblöcken bis hin zu kompletten Protokollen können über

den GPE in XML-Dateien abgespeichert und innerhalb späterer Kompositionen wiederum als Funktionsblöcke geladen werden. Als autarke Software konzipiert besteht die Möglichkeit, den GPE entweder als eigenständiges Programm auszuführen, oder als Komponente aus einer übergeordneten Software wie z. B. TILLvisION heraus aufzurufen.

Ähnlich dem GPE setzt das Softwaremodul zur Life-Akquisition (vgl. Abbildung 6.1, Modul *LA*) auf dem TILL-SDK auf. Über die beschriebenen Mechanismen bietet diese Software bei kontinuierlicher, sequenzieller Bildaufnahme die Möglichkeit zur Online-Ansicht und Analyse der mikroskopierten Objekte. Auch das Modul zur Life-Akquisition wurde als eigenständige Software konzipiert, so dass sie entweder autark oder aus einer weiteren Software heraus aufgerufen und ausgeführt werden kann.

Die Software TILLvisION V bildet innerhalb des Softwarekonzepts das integrierte Front-End für Systeme, die sich aus aus den TILL Photonics Produkten zusammensetzen. Ebenfalls auf dem TILL-SDK aufsetzend bindet es zusätzlich den soeben beschriebenen GPE und das Modul zur Life-Akquisition ein. Neben der interaktiven Bedienung der Hardware und automatisierter, protokollbasierter Durchführung von Experimenten bietet TILLvisION Möglichkeiten zur Bearbeitung und Auswertung der gemessenen Bilddaten. Zudem können Ablaufprotokolle sowie die über die Ausführung der Protokolle ermittelten Daten ganzheitlich archiviert werden.

Integrieren OEM-Kunden einzelne Komponenten oder das komplette System als Teil(e) in ein übergeordnetes Gerät, besteht die Möglichkeit, die zugekaufte Funktionalität über das TILL-SDK, den GPE und/oder die Software für Life-Akquisition in die OEM-Software einzubinden.

7 Spezifikation des Echtzeitverhaltens und Überprüfung der ICU-Funktionalität

In den vorangegangenen Kapiteln wurde ein Konzept für die zeitoptimierte Steuerung modularer Mess- und Analysesysteme vorgestellt und die Echtzeitplattform ICU dabei als zentraler Bestandteil einer flexiblen Hardwaresynchronisation zwischen Systemkomponenten eingeführt. Die Funktionalität der ICU wurde anhand der Firmware-Beschreibung in Kapitel 5 bereits detailliert veranschaulicht. Schwerpunkte lagen dabei auf den Konzepten zur Gewährleistung des gewünschten Echtzeitverhaltens auf Hardwareebene durch die zeitlich gemultiplexte, serielle Schnittstelle TMSI (s. Abschnitt 4.3) und der flexiblen Programmierung von Steuerungsabläufen durch flankierende Firmware. Hierunter fallen u. a. der Aufgabenplaner (Abschnitt 5.7) mit den von ihm koordinierten Funktionsblöcken (Aufgaben), nämlich der Schnittstelle zur Hardware (s. Abschnitt 5.2), der Abarbeitung von Protokollkommandos (s. Abschnitt 5.3) sowie der Schnittstelle zum PC (s. Abschnitt 5.6). Die Verteilung der innerhalb eines Zyklus zur Verfügung stehenden zeitlichen Ressourcen durch den Aufgabenplaner wurde bereits in Abbildung 5.13 durch das Model einer Zeitscheibe veranschaulicht. Die Dauer eines durch die Zeitscheibe beschriebenen Zyklus wird durch die Frame-Rate der TMSI bestimmt (vgl. Abschnitt 4.3). Durch die zyklisch laufende TMSI und der höchstpriorisierten Verarbeitung der zu sendenden bzw. der empfangenen Daten erfolgt die Abhandlung der Hardwareschnittstelle in harter Echtzeit.

Die Echtzeitfähigkeit des Systems hängt zudem von der Summe der Ausführzeiten aller in einem Protokoll potenziell gleichzeitig verwendbaren Kommandos ab. Zur Spezifikation des ICU-Echtzeitverhaltens sind daher die Ausführzeiten aller relevanten Firmwareblöcke sowie die der Protokollkommandos ermittelt worden. Im Folgenden werden die Ergebnisse dieser Zeitmessungen, deren Auswertung und das daraus resultierende, derzeit mögliche Zeitraster vorgestellt.

7.1 Messung der Ausführzeiten

Um die Dauer aller relevanten Funktionen auf der Basis der ICU-Systemzeit messen zu können, wurde die Firmware um eine entsprechende Funktionalität erweitert. Nach dem Prinzip einer Stoppuhr ergibt sich die Dauer der Ausführung einer Funktion dabei aus der Differenz der Systemzeiten, die jeweils zu Beginn und zum Ende eines Funktionsaufrufs notiert werden. Um den realen Zeitbedarf der Funktionen zu erhalten, wird die für das Messverfahren benötigte Rechenzeit vom Wert der ermittelten Ausführdauer abgezogen. Um eine Verfälschung der Messungen durch im Verlauf der Ausführung einer Funktion aufgerufene, höher priorisierte Interrupt-Routinen zu vermeiden, werden systemweit alle Interrupt-Mechanismen zu Beginn einer Messung ausgeschaltet und erst nach Abschluss der Messung wieder aktiviert. Da der Aufgabenplaner Interrupt-basiert aufgebaut ist, führt dies zum Verlust der Echtzeitfähigkeit für die Dauer der Zeitmessungen. Die entsprechenden Messfunktionen müssen deshalb für den Normalbetrieb der ICU aus der Firmware entfernt werden. Dazu wurde die komplette Funktionalität zur Überprüfung der Ausführzeiten über PRÄPROZESSOR-SWITCHES in die bestehende Firmware integriert und damit über das Ändern *einer* Einstellung innerhalb der Firmware zwischen Normalbetrieb und Messbetrieb umschaltbar gemacht. Im Messbetrieb wird die Anzahl der Zeitmessungen für jede Funktion notiert und über alle gemessenen Ausführzeiten jeweils der

minimale und der maximale Wert der Dauer ermittelt. Zusätzlich wird das arithmetische Mittel der letzten 128 Ausführzeiten berechnet. Die Messdaten wurden über die Schnittstelle zum PC (vgl. Abschnitt 5.6) in die Testumgebung ICU_TestingControl (vgl. Abschnitt 5.10) eingelesen und ausgewertet.

Für die Durchführung der Zeitmessungen wurden Protokolle erstellt, die aus den in Kapitel 2 vorgestellten Anwendungsbeispielen abgeleitet wurden. Da durch die Große Anzahl von mindestens 1.000 Messwerten zu jeder einzelnen Funktion die statistische Signifikanz der Zeitangaben für die minimale und die maximale Ausführdauer gewährleistet ist, wird die exakte Anzahl der Messungen in den folgenden Tabellen nicht explizit angegeben. Die Ergebnisse der Zeitmessungen werden im Folgenden jeweils zu Funktionsblöcken zusammengefasst und präsentiert.

7.1.1 TMSI Interrupt-Routine

Aufgabe 1 der im Rahmen der Beschreibungen des Aufgabenplaners präsentierten Zeitscheibe (s. Abbildung 5.13) umfasst die auf Seite 71 beschriebene TMSI Interrupt-Routine, mittels der die Schnittstelle zwischen der Hauptplatine und der weiteren ICU-Hardware bedient wird. Zu Beginn eines jeden Zyklus wird zunächst die TMSI Interrupt-Routine nach dem beschriebenen Prinzip der Aufgabenplanung mit der höchsten Priorität ausgeführt. Die Ausführdauer dieser Routine hat daher einen wesentlichen Einfluss auf das Zeitraster der Echtzeit des Systems.

Test Nr.	Funktions-Name	Ausführdauer [µs] min.	Durchschnitt	max.
1	TMSI Interrupt-Routine	12,534	12,694	13,077

Tabelle 7.1: Zeitbedarf der TMSI Interrupt-Routine. Diese Routine stellt die komplette Aufgabe 1 der in Abbildung 5.13 vorgestellten Zeitscheibe dar.

TMSI Interrupt-Routine: Abhängig von der Anzahl der neu zu sendenden sowie der sinnvollen, empfangen Daten variiert der Zeitbedarf für die Ausführung der Interrupt-Routine. Die Abweichung beläuft sich allerdings auf weniger als 5 %.

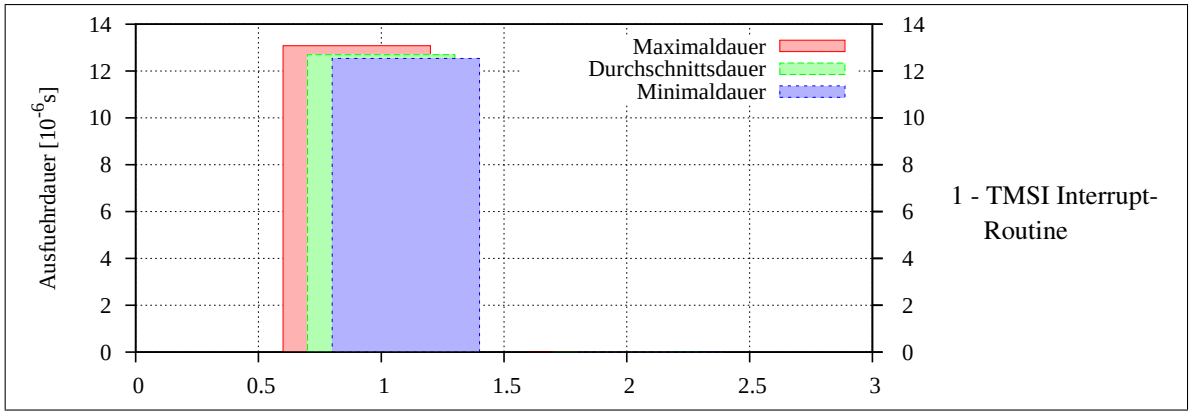


Abbildung 7.1: Übersicht der Ausführdauer der TMSI Interrupt-Routine.

7.1.2 Hardwareunabhängige Protokollfunktionen

Das Firmwaremodul *prot* bietet neben der Protokollstruktur bereits allgemeine, hardware- und anwendungsunabhängige Kommandos für die Verwendung innerhalb von Protokolldefinitionen

(s. Abschnitt 5.3.1). Der Zeitbedarf der mit diesen Kommandos verbundenen Funktionen ist in Tabelle 7.2 zusammengefasst.

Test Nr.	Funktions-Name	Ausführdauer [µs]		
		min.	Durchschnitt	max.
2	Execute*	2,049	2,049	2,076
3	NoOp	0,258	0,258	0,258
4	Loop	0,748	0,718	0,748
5	EndLoop	0,410	0,436	0,775
6	StdLoopCond	0,383	0,385	0,392
7	If	0,534	0,534	0,534
8	Else	0,445	0,445	0,445
9	EndIf	0,258	0,258	0,258

Tabelle 7.2: Zeitbedarf der hardware- und anwendungsunabhängigen Protokollfunktionen. Abhängig von der Protokolldefinition werden diese Funktionen jeweils als Teil von Aufgabe 2 (vgl. Abbildung 5.13) abgearbeitet. Die Ausführdauer der mit * markierten Funktion hängt von der Protokolldefinition ab. Die angegebenen Werte beziehen sich daher auf die kürzest mögliche Definition.

Execute: Die Funktion *Execute* stellt den in Abschnitt 5.3.1 beschriebenen Mechanismus zum Ausführen eines bzw. mehrerer ohne Verzögerung aufeinander folgender Protokollkommandos dar. Da alle in ein Protokoll eingetragenen Kommandos aus der Funktion *Execute* heraus ausgeführt werden, muss ihre Ausführdauer bei der Abarbeitung eines oder mehrerer aufeinander folgender Kommandos lediglich *einmalig* berücksichtigt werden. Nach erfolgreicher Ausführung der Protokollkommandos wird durch eine externe Funktion die Bedingung für die Ausführung des nächsten Kommandos festgelegt (vgl. Abbildung 5.7). Da der Aufruf der Funktion zum Festlegen der Ausführbedingung für das nächste Kommando einen festen Bestandteil der Funktion *Execute* darstellt, ist der hierfür notwendige Zeitbedarf in den angegebenen Ausführzeiten berücksichtigt. Da die Anzahl und die Art der ausgeführten Protokollkommandos variiert, muss die Ausführdauer der etwaig aufgerufenen Funktionen zu der Ausführzeit der Funktion *Execute* addiert werden.
NoOp: Sinnvolle Möglichkeiten der Verwendung des Befehls *NoOp* wurden bereits an früherer Stelle beschrieben (s. Abschnitt 5.9). Obwohl das Kommando keine Funktionalität hat, führt ein entsprechender Protokolleintrag dennoch zum Aufruf einer - wenn auch leeren - Funktion, die selbstverständlich Rechenzeit bedarf.
Loop: Bei der Ausführung des Kommandos *Loop* wird die für wiederholte Abarbeitung benötigte Datenstruktur einer Schleife initialisiert. Hierbei wird u. a. der Verweis auf die Funktion gesetzt, welche die Schleifenbedingung formuliert. Im Anschluss wird abhängig von der angegeben Bedingung entschieden, ob der Kommandoblock innerhalb der Schleife ausgeführt wird, oder ob die Ausführung ab dem Ende der Schleife fortgesetzt wird. Da die Bedingungsfunktion durch den Mechanismus der „Events“ individuell formuliert werden kann, ist die Dauer der Bedingungsüberprüfung nicht in der angegebenen Ausführdauer enthalten und muss individuell hinzuaddiert werden.
EndLoop: Das durch das Kommando *EndLoop* markierte Ende einer Schleife führt zu einer erneuten Prüfung der Schleifenbedingung. Bei negativem Ergebnis wird regulär mit der

Ausführung des Protokolls fortgefahren. Ist bei ineinander verschachtelten Schleifen das Ende der Schleife auf oberster Ebene erreicht, sind zuvor einige interne Vorgänge zum Zurücksetzen der Strukturen für Schleifen und Verzweigungen notwendig. Ein positives Ergebnis führt zu einem Sprung in der Protokollliste an die Position des ersten Kommandos innerhalb des Blocks, der in der Schleife abgearbeitet wird. Die mit dem Kommando *Loop* verknüpfte Funktion wird daher nicht erneut aufgerufen. Auch hier ist die angegebene Ausführdauer exklusive der für die Bedingungsüberprüfung benötigte Zeit zu verstehen.
StdLoopCond: Die Verwendung von Schleifen wird in vielen Fällen lediglich für die Ausführung eines Kommandoblocks mit fester Anzahl der Wiederholungen und ohne weitere Abbruchbedingungen verwendet. Für diesen Standardfall bietet das Modul *prot* eine Funktion, die eine Bedingung in Abhängigkeit einer Zählervariablen und der angegeben Gesamtanzahl der Schleifendurchläufe beschreibt: *StdLoopCond*. Wie bereits beschrieben wurde, muss die Ausführdauer bei der Verwendung dieser Bedingungsfunktion zu denen der Funktionen *Loop* und *EndLoop* addiert werden.
If: Innerhalb der mit dem Kommando *If* verknüpften Funktion wird erst nach der Überprüfung der zu verwendenden Bedingung über die weitere Ausführung entschieden. Bei negativem Ergebnis wird entweder unmittelbar an den Anfang des durch das Kommando *Else* markierten, alternativen Kommandoblocks oder unmittelbar an den durch das Kommando *EndIf* angegebene Abschluss der verzweigten Ausführung gewechselt. In beiden Fällen wird mit der Ausführung des Kommandos, das unmittelbar nach den Kommandos *Else* bzw. *EndIf* in der Protokollliste eingetragen ist, fortgefahren. Da die Bedingungsfunktion auch hier individuell angegeben werden kann, beziehen sich die aufgelisteten Ausführzeiten wiederum lediglich auf die mit dem Kommando verknüpfte Funktion ohne die Dauer der Bedingungsüberprüfung.
Else: Das Kommando *Else* wird nur abgearbeitet, wenn die Bedingungsüberprüfung innerhalb der mit dem Kommando *If* verknüpften Funktion negativ verläuft. Es bewirkt einen Wechsel zum Kommando, das unmittelbar nach dem durch das Kommando ***EndIf*** markierten Ende der verzweigten Ausführung in der Protokollliste eingetragen ist. Da die entsprechende Funktion eine eindeutige und in sich abgeschlossene Funktionalität hat, ist die angegebene Ausführzeit als absolut anzusehen.
EndIf: Im Fall der Ausführung des alternativen Kommandoblocks wird zum Ende der verzweigten Abarbeitung das Kommando *EndIf* ausgeführt. Obwohl es lediglich den Abschluss markiert und keine weitere Funktionalität besitzt, kann über den Parameter „Ausführzeitpunkt“ der zeitlichen Ablauf von Protokollen beeinflusst werden. Die angegebene Ausführzeit der mit dem Kommando verknüpften Funktion ist ebenfalls als absolut anzusehen.

Die Messergebnisse des Zeitbedarfs aller hardware- und anwendungsunabhängigen Protokollkommandos sind in Abbildung 7.2 in einem Säulendiagramm graphisch zusammengefasst.

7.1.3 Anwendungsabhängige Protokollfunktionen

Der Zeitbedarf der in Abschnitt 5.3.2 beschriebenen ICU-spezifischen Protokollfunktionen des Moduls *prot_ext* wurde nach der gleichen, bereits beschriebenen Messmethode ermittelt. Einer übersichtlichen Gliederung wegen sind die Funktionen und Messergebnisse in zwei Blöcke unterteilt. Der erste Block beschreibt anwendungsabhängige Funktionen der Protokoll-“Infrastruktur“, der Zweite überwiegend hardwarespezifische Funktionen. Tabelle 7.3 fasst den Zeitbedarf der mit den Kommandos des ersten Blocks verbundenen Funktionen zusammen.

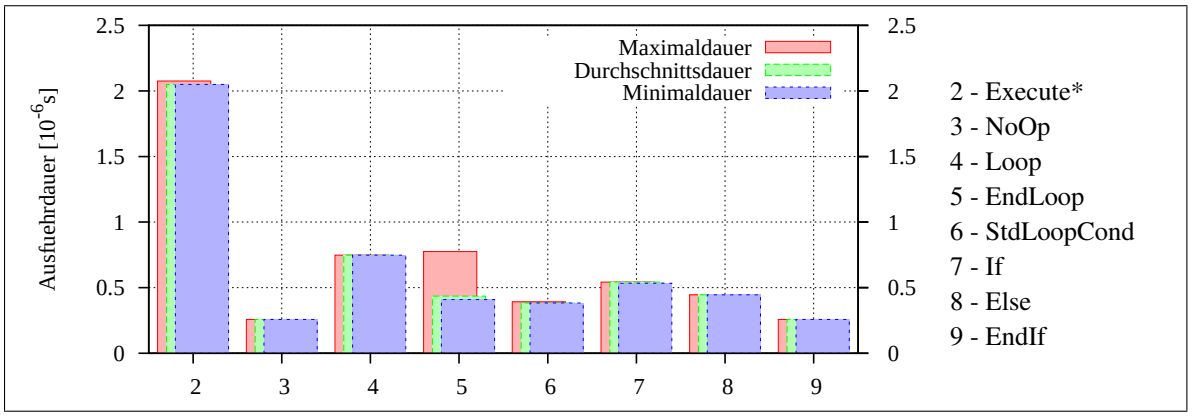


Abbildung 7.2: Graphische Übersicht der Ausführdauer der vom Modul *prot* für die Verwendung innerhalb von Protokollen zur Verfügung gestellten Funktionen.

Test Nr.	Funktions-Name	Ausführdauer [µs] min.	Durchschnitt	max.
10	CheckTimer	0,606	0,704	0,793
11	SetNextEvent	0,775	2,352	3,634
12	SendResponse*	9,184	9,184	9,184
13	SendResponse°	1,399	1,577	1,871
14	WriteVariable	6,120	6,120	6,120
15	ReadVariable	18,40	18,40	18,41
16	ResetExpTimer	18,21	18,37	18,67
17	SendExpTimerValue	17,74	18,15	18,55

Tabelle 7.3: Zeitbedarf anwendungsabhängiger Protokollfunktionen. Die Ausführdauer von mit * markierten Funktionen variiert abhängig von der Protokolldefinition. Die angegebenen Werte der mit ° markierte Funktion beziehen sich auf das Versenden jedes weiteren Zeichens/Bytes.

CheckTimer: Protokollkommandos werden entsprechend der Protokolldefinition abgearbeitet, wenn die Ausführbedingung erfüllt ist (vgl. Aufgabe 2 der Zeitscheibe in Abbildung 5.13). Für eine deterministische Protokollabarbeitung basiert die Ausführbedingung auf der Überprüfung der seit der Ausführung des letzten Kommandos vergangen Systemzeit. Spezifiziert der Parameter Ausführzeitpunkt eines Kommandos eine zeitliche Verzögerung zur Abarbeitung des vorherigen Kommandos, wird zu Beginn der Abarbeitung von Aufgabe 2 die Funktion *CheckTimer* aufgerufen. Bei positivem Ergebnis der Überprüfung wird im Anschluss die Protokollfunktion *Execute* (vgl. Abschnitt 5.3.1) zur Ausführung des Protokollkommandos aufgerufen. Die Ausführzeit der Funkton *CheckTimer* ist bei deterministischen Abläufen daher bezüglich der Ermittelung des Zeitbedarfs von Aufgabe 2 immer zu berücksichtigen.
SetNextEvent: Wie im Flussdiagramm in Abbildung 5.7 veranschaulicht, wird die Funktion *SetNextEvent* am Ende jeder Abarbeitung der Protokollfunktion *Execute* aufgerufen, um das Ausführverfahren für das nächste Kommando festzulegen. Deswegen sind die Ausführdauern (Min., Max. und Durchnitt) der Funktion *SetNextEvent* bereits in der der Protokollfunktion *Execute* enthalten (s. Tabelle 7.2). Die für die Funktion *SetNextEvent* angegeben Zeiten sind lediglich der Vollständigkeit halber angegeben und müssen ansonsten nicht weiter berücksichtigt werden.

SendResponse: Wie auf Seite 100 beschrieben wurde, können über diese Funktion benutzerdefinierte Rückmeldungen zur Laufzeit aus einem Protokoll heraus an den PC gesendet werden. Der Zeitbedarf dieser Funktion ist dabei abhängig von der Länge der spezifizierten Antwort. Um die Planung zeitkritischer Protokollteile optimal gestalten zu können, beziehen sich die in Tabelle 7.4 angegebenen Ausführzeiten auf das Senden einzelner Zeichen/Bytes als kleinste zu übertragende Nachricht. Zur Ermittelung des Zeitbedarfs längerer Antworten wird die für jedes weitere Zeichen/Byte zusätzlich benötigte Zeit in der Zeile SendResponse° angegeben.
WriteVariable und ***ReadVariable***: Prinzipiell ist das Schreiben und Lesen einer Variable vom Typ der Variablen abhängig. Da bisher lediglich der Typ „Integer“ unterstützt wird, sind die angegeben Ausführdauern jeweils als absolut anzusehen.
Dies gilt auch für die Ausführzeiten der mit den Kommandos ***ResetExpTimer*** und ***SendExpTimerValue*** verbundenen Funktionen zum Zurücksetzen und Senden der ICU-internen „Stoppuhr“ (s. Seite 100).

Abbildung 7.3 fasst den gemessenen Zeitbedarf aller anwendungsabhängigen Protokollkommandos graphisch zusammen.

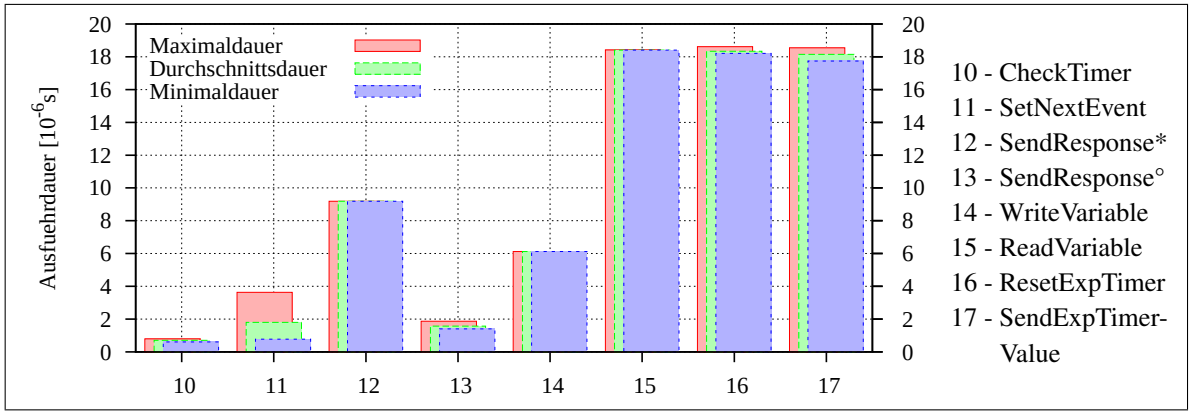


Abbildung 7.3: Übersicht der Ausführdauern anwendungsabhängiger Protokollfunktionen. Die Ausführdauer von mit * markierten Funktionen variiert abhängig von der Protokolldefinition. Die angegebenen Werte der mit ° markierte Funktion beziehen sich auf das Versenden jedes weiteren Zeichens/Bytes.

7.1.4 Hardwareabhängige Protokollfunktionen

Ein zweiter Block fasst die derzeit verfügbaren Kommandos zur Kontrolle der ICU-Hardware zusammen. Hierunter fallen alle digitalen und analogen I/Os sowie die EIA-232-Schnittstellen. Tabelle 7.4 präsentiert den Zeitbedarf der mit diesen Kommandos verbundenen Funktionen.

WriteDigitalOut: Die Ausführzeit der mit dem Kommando *WriteDigitalOut* verknüpften Funktion ist unabhängig von den zu setzenden digitalen Ausgängen (Bank, Line-Maske, Wert-Maske) und damit als absolut anzusehen.
ReadDigitalIn: Wie in Fall des Kommandos *WriteDigitalOut* ist die hiermit verbundene Funktion ebenfalls unabhängig von den zu lesenden digitalen Eingängen (Bank, Line-Maske, Wert-Maske) und damit als absolut anzusehen.
WriteAnalogOut: Die Dauer der Abarbeitung der mit dem Kommando *WriteAnalogOut* verbundenen Funktion ist ähnlich der des Kommandos *WriteDigitalOut* unabhängig von dem zu setzenden Wert und dem ausgewählten analogen Kanal für jeden Aufruf identisch. Sollen in einem

Test Nr.	Funktions-Name	Ausführdauer [µs] min.	Durchschnitt	max.
18	WriteDigitalOut	1,416	1,416	1,416
19	ReadDigitalIn	11,39	11,39	11,46
20	WriteAnalogOut	1,791	1,817	1,844
21	ReadAnalogOut	13,23	13,23	13,23
22	ReadAnalogIn	13,23	13,23	13,23
23	WriteUART*	1,547	1,659	1,666
24	WriteUART°	0,704	0,704	0,704
25	ReadUART*	10,11	11,41	13,47
26	ReadUART°	0,695	0,713	0,784

Tabelle 7.4: Zeitbedarf der Funktionen hardwarebezogener Protokollkommandos. Die Ausführdauer von mit * markierten Funktionen variiert abhängig von der Protokolldefinition. Die angegebenen Werte der mit ° markierte Funktionen beziehen sich auf das Versenden jedes weiteren Zeichens/Bytes.

Protokoll mehrere analoge Ausgänge gleichzeitig (Parameter Ausführzeitpunkt = 0) geändert werden muss allerdings das in Abschnitt 5.2.1 beschriebene Verhalten berücksichtigt werden.
ReadAnalogOut: Das Kommando *ReadAnalogOut* dient lediglich zur Überprüfung der Zustände der analogen Ausgänge durch die PC-Software und ist daher nicht für die Verwendung innerhalb von Protokollen vorgesehen (s. Abschnitt 5.2.1). Der Vollständigkeit halber sind die Ausführzeiten der entsprechenden Funktion an dieser Stelle dennoch mit aufgeführt.
ReadAnalogIn: Die Ausführzeit der mit dem Kommando *ReadAnalogIn* verknüpften Funktion entspricht exakt derer der Funktion ReadAnalogOut. Allerdings muss hierbei berücksichtigt werden, dass der entsprechende Eingang durch eine vorherige Abfrage mittels des Kommandos ReadAnalogIn selektiert wurde.
WriteUART: Im Gegensatz zu den Funktionen zum Setzen digitaler oder analoger Ausgänge besteht bei der Ausführdauer der mit dem Kommando *WriteUART* (s. Seite 101) verknüpften Funktion eine Abhängigkeit von der Anzahl der zu sendenden Zeichen/Bytes. Um die Planung zeitkritischer Protokollteile optimal gestalten zu können, beziehen sich die in Tabelle 7.4 in Zeile WriteUART* angegebenen Ausführzeiten daher auf das Senden lediglich eines Zeichens/Bytes als kleinste zu übertragende Nachricht. Für die Zeitplanung zum Senden größerer Daten- bzw. Kommandosequenzen wird die zusätzlich benötigte Zeit für jedes weitere Zeichen/Bytes in der Zeile WriteUART° angegeben.
ReadUART: Ähnlich wie beim Kommando *WriteUART* ist die Ausführdauer von der Anzahl der an den PC zu senden Zeichen/Bytes abhängig. Es wurde wieder die minimal für das Senden der kleinsten Nachrichteneinheit benötigte Zeit (Zeile ReadUART*) sowie die zusätzliche Dauer für jedes weitere Zeichen/Bytes (Zeile ReadUART°) angegeben.

7.1.5 Event-Funktionen

Event-Funktionen können sowohl als Bedingung für Schleifen oder verzweigte Ausführung, als auch anstelle einer zeitlichen Angabe der Ausführung verwendet werden (Details s. Abschnitt 5.5). Im ersten Fall beeinflussen die Events den Ablaufffluss von Protokollteilen, im zweiten den Ausführzeitpunkt eines Kommandos und aufgrund der relativ angegebenen

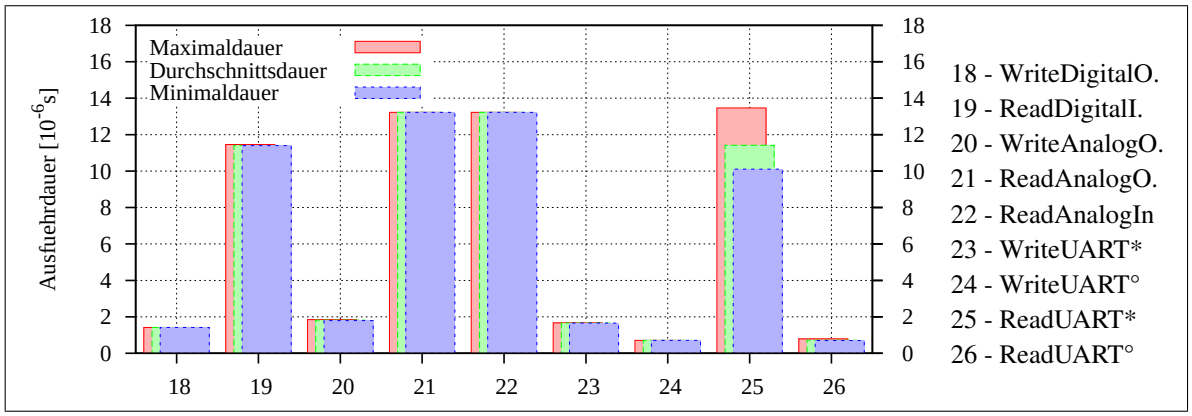


Abbildung 7.4: Übersicht der Ausführdauern hardwareabhängiger Protokollfunktionen. Die Ausführdauer von mit * markierten Funktionen variiert abhängig von der Protokolldefinition. Die angegebenen Werte der mit ° markierte Funktionen beziehen sich auf das Versenden jedes weiteren Zeichens/Bytes.

Verzögerungszeiten auch aller folgenden Kommandos. Wie weiter oben beschrieben wurde, muss bei der Verwendung eines Events in Verbindung mit den Kommandos *Loop*, *EndLoop* und *If* die für die Überprüfung benötigte Dauer zu der Ausführzeit der mit den aufrufenden Kommandos verbundenen Funktionen hinzuaddiert werden. Die Dauer der Eventfunktionen sowie der Funktion, über welche die Eventabfrage aufgerufen wird, wurde ermittelt. Die Ergebnisse sind in Tabelle 7.5 zusammengefasst.

Test Nr.	Funktions-Name	Ausführdauer [µs] min.	Durchschnitt	max.
27	CheckEvent	0,276	0,276	0,276
28	DIn	1,452	1,452	1,461
29	Soft	0,347	0,347	0,347
30	Wait	0,303	0,303	0,303
31	Expression*	16,355	16,355	16,364

Tabelle 7.5: Zeitbedarf der Event-Funktionen sowie der Funktion zur Event-Abfrage. Die Ausführdauer von mit * markierten Funktionen variiert abhängig von der Protokolldefinition.

CheckEvent: Jede Überprüfung eines Events, ob anstelle einer Ausführzeit oder als Bedingung für Schleifen oder Verzweigungen eingesetzt, erfolgt über den Aufruf der Funktion *CheckEvent*. Durch Angabe des Event-Handles wird die in der Eventliste eingetragene Funktion mit den zu übergebenden Parametern aufgerufen. Die Ausführdauer der Funktion ist damit bei jeder Event-Abfrage zu berücksichtigen.
DIn, ***Soft*** und ***Wait***: Die Ausführdauer der in Abschnitt 5.5.1 detailliert beschriebenen Eventfunktionen Digitaler Eingang Event (*DIn*), Software Event (*Soft*) und Warten Event (*Wait*) ist jeweils absolut anzusehen und zur Dauer der Funktion *CheckEvent* hinzuzuaddieren.
Expression: Da durch die eingebettete Mathematik (vgl. Abschnitt 5.4) die Möglichkeit besteht, beliebige mathematische Ausdrücke zu erstellen, lässt sich keine allgemeingültige Methode zur Bestimmung der Ausführdauer des mathematischen Events angeben. Die Dauer der Abfrage eines solchen Events hängt selbstverständlich von der Komplexität der auszuführenden mathematischen Operationen ab. Die in Tabelle 7.5 angegebenen Ausführdauer wurden daher exem-

plarisch mit einer häufig auftretenden Standardbedingung für verzweigte Ausführung ermittelt („l1==5“). In diesem Beispiel wird bei der Ausführung innerhalb einer Schleife der durch die Kommandos *If* und *EndIf* begrenzte Kommandoblock nur beim 5-ten Schleifendurchlauf ausgeführt. „l1“ beschreibt die ICU-interne Zählervariable der jeweils aktuell ausgeführten Schleife auf der obersten Ebene (bei ineinander geschachtelten Schleifen würde „l2“ die Zählervariable auf zweithöchster Ebene beschreiben, usw.). Zur Bestimmung der Ausführzeit von benutzerdefinierten mathematischen Ausdrücken kann das auf Seite 101 beschriebene Kommando *MathExpressionDurationCheck* verwendet werden.

Der gemessene Zeitbedarf der mit der Event-Funktionalität verbundenen Funktionen ist in Abbildung 7.5 durch ein Säulendiagramm graphisch zusammengefasst.

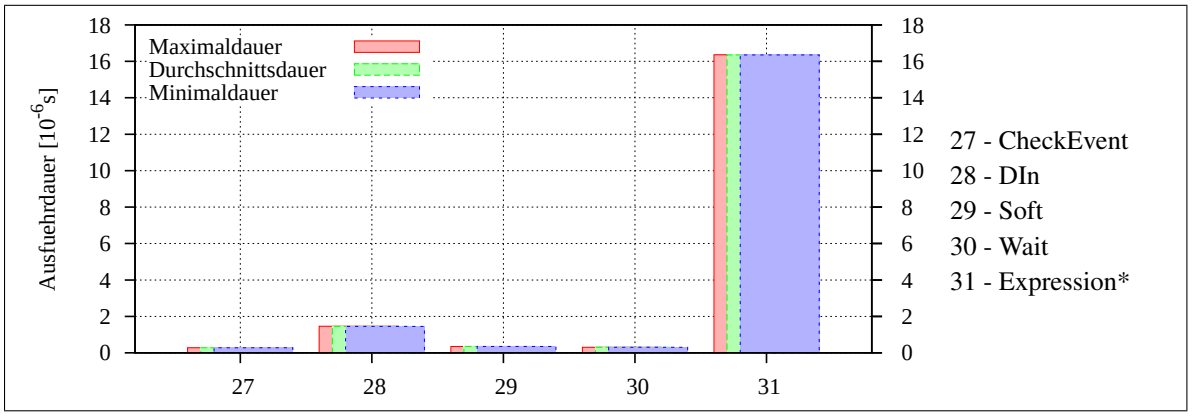


Abbildung 7.5: Übersicht der Ausführdauer der Event-Funktionen und der zum Überprüfen von Events notwendigen Funktion *CheckEvent*. Die Ausführdauer von mit * markierten Funktionen variiert abhängig von der Protokolldefinition.

7.2 Zeitverhalten der ICU-Hardware

Durch die beschriebenen Hardware- und Firmwaremechanismen werden alle ICU-Hardwareschnittstellen (digital I/O, analog I/O, serielle Schnittstellen wie z. B. die EIA-232-Schnittstellen) gleichwertig behandelt. Das aus den vorgestellten Konzepten resultierende Zeitverhalten der ICU-Hardware ist daher für alle Schnittstellen identisch. Die Betrachtung des in den Abbildungen 7.6 und 7.7 exemplarisch an digitalen Ein- bzw. Ausgängen veranschaulichten Verhaltens ist somit allgemein gültig. Werte für die in den Verlaufsdiagrammen markierten Zeiten sind in der jeweils entsprechenden Tabelle 7.6 bzw. 7.7 zu finden.

Abbildung 7.6 veranschaulicht die zeitlichen Zusammenhänge zwischen der Protokollausführung (vgl. Aufgabe 2 in Abbildung 5.13) und dem Zeitverhalten der in harter Echtzeit betriebenen Hardwareschnittstellen (vgl. Aufgabe 1 in Abbildung 5.13) am Beispiel eines digitalen Ausgangs. Über die Ausführung eines entsprechend definierten Protokolls (s. Auflistungen B.1) wird der Pegel eines digitalen Ausgangs der ICU in einer Endlosschleife im Abstand von 40 µs invertiert (s. Abbildung 7.6: t1 bzw. t2). Der theoretische Zeitverlauf des so spezifizierten Signals ist in Abbildung 7.6 in der mit „TrgOut (Soll)“ beschrifteten Zeile, der real gemessene Zeitverlauf in der mit „TrgOut (Ist)“ beschrifteten Zeile angegeben. In der mit „TMSI-Zyklen“ beschrifteten Zeile wurden die Zyklen des die Hardware bedienenden TMSIs aus dem beobachteten Verhalten des digitalen Ausgangs rekonstruiert.

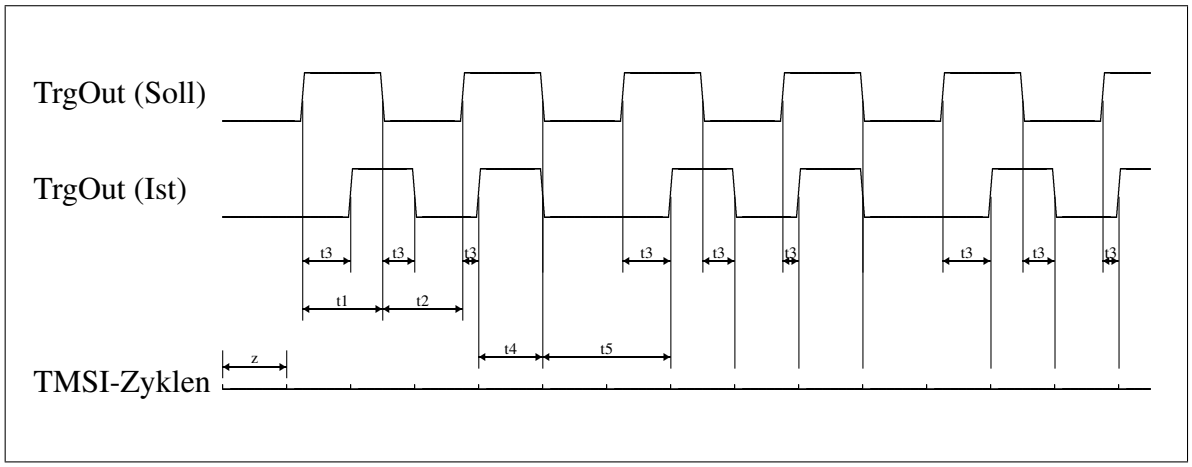


Abbildung 7.6: Zeitverlaufsdiagramm des Pegels eines digitalen Ausgangs der ICU. Die Soll-Werte werden durch ein entsprechend programmiertes Protokoll deterministisch vorgegeben. Zeitwerte für z bzw. t1-t5: s. Tabelle 7.6

Symbol	Beschreibung	Dauer [µs]		
		min.	typisch*	max.
z	TMSI Zyklusdauer	–	34	–
t1	Soll HIGH	–	40	–
t2	Soll LOW	–	40	–
t3	Soll-Flanke zu Ist-Flanke	0	–	z
t4	Ist HIGH	z	–	2*z
t5	Ist LOW	z	–	2*z

Tabelle 7.6: Zeitwerte zum Zeitverlaufsdiagramm in Abbildung 7.6: ICU-erzeugter Trigger Puls. Bei der mit * markierten Spalte gilt: Sind weder min. noch max. Werte angegeben, so ist die genannte Dauer als exakt anzusehen.

Ein Pegelwechsel wird durch Setzen des entsprechenden digitalen Ausgangs zu dem im Protokoll angegebenen Zeitpunkt initiiert. Diese zunächst lediglich lokal auf dem DSP vorhandene Änderung wird innerhalb des nächsten TMSI-Zyklus auf die Hardware - in diesem Beispiel das CPLD - übertragen. Nach dem im Abschnitt 5.7 beschriebenen Aufbau des Aufgabenplaners sind die Bedienung der Hardwareschnittstellen (Aufgabe 1) und die Abarbeitung von Protokollen (Aufgabe 2) prinzipiell synchron zu einander. Aufgrund der unterschiedlichen Zeitbasen der Protokolldefinitionen (10 µs; vgl. Abschnitt 5.3.2) und der TMSI (derzeit 34 µs), kommt es allerdings bei der Umsetzung eines Soll-Wertes zum entsprechenden Ist-Wert zu einer Verzögerung, die maximal der Dauer eines TMSI-Zyklus entspricht (s. Abbildung 7.6: t3). Wird der lokal aktualisierte Wert im Verlauf dieser Verzögerung erneut verändert, so wird die jeweils zuletzt vorgenommene Änderung auf die Hardware übertragen. Bei der Protokolldefinition von beispielsweise eines Triggerpulses, ist daher darauf zu achten, dass die Breite des Pulses mindestens der Dauer eines TMSI-Zyklus entspricht, da der Puls ansonsten ggf. lediglich DSP-intern registriert, aber nicht an die Hardware übertragen wird. Diese Eigenschaft wurde für das in Abbildung 7.6 dargestellte Beispiel berücksichtigt. Zur Veranschaulichung des Verhaltens ist zudem die derzeit für das Erzeugen eines Triggerpulses bzw. des dargestellten Taktsignals kleinste sinnvolle Verzögerung von 40 µs gewählt worden. Da die Aktualisierung der Hardwareschnittstellen stets im festen Raster der TMSI-Zyklen erfolgt, äußert sich die flie-

ßende Verzögerung auf der Hardwareseite durch einen diskreten Jitter von 0 µs bzw. der Dauer eines TMSI-Zyklus (vgl. Abbildung 7.6). Dieses Verhalten ist in Abbildung 7.6 exemplarisch durch die Zeiten t4 und t5 verdeutlicht. Der Jitter ist unabhängig vom Abstand zweier in einem Protokoll definierten Änderungen einer Schnittstelle und muss daher lediglich bei Protokollsequenzen zu berücksichtigt werden, in denen zwei aufeinander folgende Veränderungen an derselben Schnittstelle in der Größenordnung der Dauer eines TMSI-Zyklus liegen.

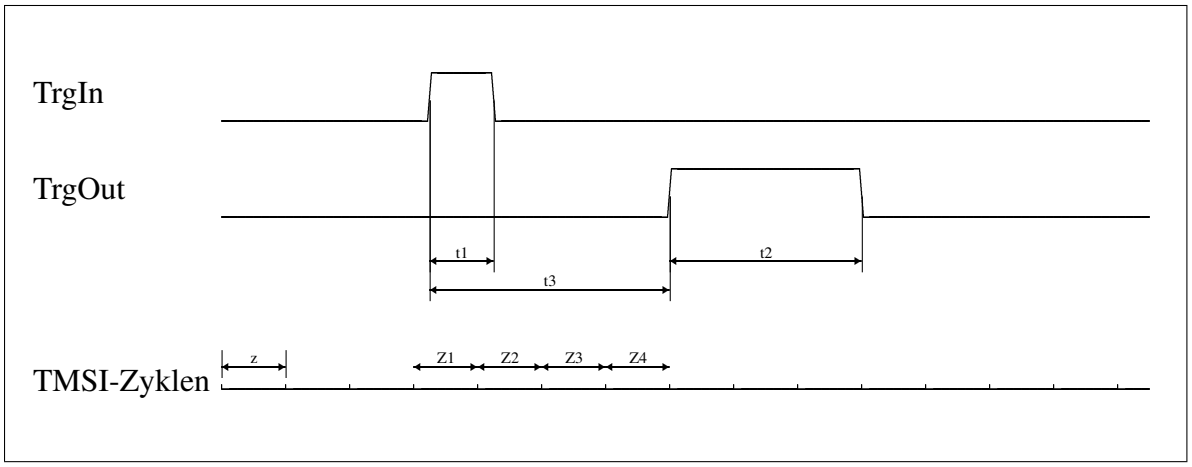


Abbildung 7.7: Zeitverlaufsdiagramm des Ablaufs eines entsprechend programmierten ICU Protokolls: Ausgabe eines Trigger Pulses an einem digitalen Ausgang als Reaktion auf den Pegelwechsel eines digitalen-Eingangs. Zeitwerte für z bzw. t1-t3: s. Tabelle 7.7.

Abbildung 7.7 veranschaulicht die zeitlichen Zusammenhänge zwischen eingehenden und ausgehenden Signalen am Beispiel eines protokollgenerierten Pulses an einem digitalen Ausgang als Reaktion auf einen detektierten Pegelwechsel (Flanken-Trigger) eines digitalen Eingangs. Über die Ausführung eines entsprechend definierten Protokolls (s. Auflistungen B.2) wird abhängig vom Pegel eines digitalen Eingangs der Pegel eines digitalen Ausgangs auf den Wert logisch 1 gesetzt. Nach einer Verzögerung von 110 µs wird der Pegel des digitalen Ausgangs wieder auf den Wert logisch 0 zurückgesetzt. Der Zeitverlauf der mit „TrgIn“ beschrifteten Zeile in Abbildung 7.6 zeigt das an den digitalen Eingang angeschlossene Signal eines Pulsgenerators. Das von der ICU anhand des Protokolls generierte Signal des digitalen Ausgangs findet sich in der mit „TrgOut“ beschrifteten Zeile. In der mit „TMSI-Zyklen“ beschrifteten Zeile wurden für ein besseres Verständnis wiederum die Zyklen des die Hardware bedienenden TMSIs aus dem beobachteten Verhalten rekonstruiert.

Symbol	Beschreibung	Dauer [µs]		
		min.	typisch*	max.
z	TMSI Zyklusdauer	–	34	–
t1	Pulsbreite Trigger In	z	–	–
t2	Pulsbreite Trigger Out	z	100	–
t3	Verzögerung In zu Out	3*z	–	4*z

Tabelle 7.7: Zeitwerte zum Zeitverlaufsdiagramm in Abbildung 7.7: ICU-erzeugter Trigger Puls als Reaktion auf den Pegelwechsel an einem digitalen Eingang. Bei der mit * markierten Spalte gilt: Sind weder min. noch max. Werte angegeben, so ist die genannte Dauer als exakt anzusehen.

Um die Verzögerung sowie die einzelnen notwendigen Prozesse von der Detektion des Pegelwechsels bis zur Ausgabe des im Protokoll spezifizierten Triggerspulses im Folgenden beschreiben zu können, wurden die entsprechenden TMSI-Zyklen beschriftet (s. Abbildung 7.7: Z1 bis Z4). Eine Veränderung des am digitalen Eingang anliegenden Pegels während eines laufenden Zyklus (Z1) wird im darauf folgenden Zyklus (Z2) über das TMSI an den DSP übertragen. Nach der Übertragung wird der Wert lokal im DSP aktualisiert (vgl. Abschnitt 5.7 Aufgabe 1) und führt in der darauf folgenden Abarbeitung des Protokolls (Aufgabe 2) zur im Protokoll spezifizierten Reaktion - im gezeigten Beispiel dem Setzen des Pegels des digitalen Ausgangs auf den Wert logisch 1. Die Abarbeitung des Protokolls im laufenden Zyklus (Z3) führt zunächst ebenfalls lediglich zu einer Änderung des lokalen Wertes. Nach der Übertragung dieses Wertes im folgenden Zyklus (Z4) wird die Hardware mit dem neuen Wert aktualisiert. Da die jeweils am Ende eines TMSI-Zyklus an den digitalen Eingängen anliegenden Pegel im darauffolgenden Zyklus an den DSP übertragen werden, muss die Breite eines eingehenden Triggers mindestens der eines TMSI-Zyklus entsprechen (s. Abbildung B.2: t1), um definitiv vom DSP wahrgenommen zu werden. Die Verzögerung von der Detektion einer Veränderung des Eingangssignals bis zur protokollgemäßen Reaktion (s. Abbildung B.2: t3) besteht aus einer flexiblen Verzögerung vom Pegelwechsel bis zum Start der Übertragung des neuen Wertes an den DSP (0 µs bis zur Dauer eines kompletten Zyklus: 34 µs), sowie einer darauf folgenden konstanten Verzögerung (1 Zyklus: Übertragung Hardware → DSP; 1 Zyklus: Abarbeitung des Protokolls, 1 Zyklus: Übertragung DSP → Hardware). Die Pulsbreite des durch das Protokoll spezifizierten, ausgehenden Triggers (s. Abbildung B.2: t2) muss aufgrund des bereits im vorherigen Beispiels beschriebenen Verhaltens mindestens der Dauer eines TMSI-Zyklus entsprechen.

7.3 Funktionsüberprüfung anhand von Beispielanwendungen

Die ICU wird bereits seit Mitte des Jahres 2005 in TILL Imaging-Systemen sowie innerhalb der automatisierten Mikroskopie auf Basis des iMIC eingesetzt. Die PC-Software TILLvisION bildet für beide Systeme die Benutzerschnittstelle und dient zum Konfigurieren der Abläufe sowie zur Anzeige, Analyse und Archivierung der Messdaten. Im Zuge der Einführung der ICU wurde die Software auf die Kommandoschnittstelle angepasst und bereits auf den überwiegenden Teil der zur Verfügung stehenden Funktionalität erweitert. Im Weiteren werden konkrete, über diese PC-Software aufgesetzte Beispielanwendungen vorgestellt, die zur Verifizierung der ICU-Funktionalität durchgeführt wurden. Die Beispiele basieren auf dem Einsatz eines Aufbaus bestehend aus einem Imaging-System (CCD-Kamera und Polychrome V), dem iMIC, der ICU sowie einem PC unter Verwendung der Software TILLvisION (Version IV).

7.3.1 TILLvisION IV - Schnappschuss und Live-Modus

Beim Einsatz des Mikroskopiesystems in alltäglichen biologischen Anwendungen, wird ein Präparat zu Beginn eines Versuchs i. d. R. auf interessante und aussagekräftige Regionen hin untersucht. Dies kann z. B. ein Blickfeld sein, in dem sich besonders viele bzw. gut entwickelte Zellen befinden. Für diesen Suchvorgang bietet TILLvisION zwei Modi: Snapshot und Live-Modus. In einer Eingabemaske (Abbildung 7.8) werden zunächst alle für eine Bildaufnahme relevanten Parameter wie z. B. Belichtungszeit, Wellenlänge, Intensität und zu verwendende Filter (bei Fluoreszenzmikroskopie) eingetragen bzw. ausgewählt. Nach Betätigen der Befehlsschaltfläche für einen Snapshot wird genau ein Bild aufgenommen und auf dem Monitor ausgege-

ben. Wird die Befehlsschaltfläche für den Live-Modus betätigt, führt das System entsprechend der eingetragenen Parameter kontinuierlich Bildaufnahmen mit unmittelbarer Anzeige auf dem Monitor aus. Ein erneutes Betätigen der Befehlsschaltfläche stoppt diese kontinuierliche Bildaufnahme.

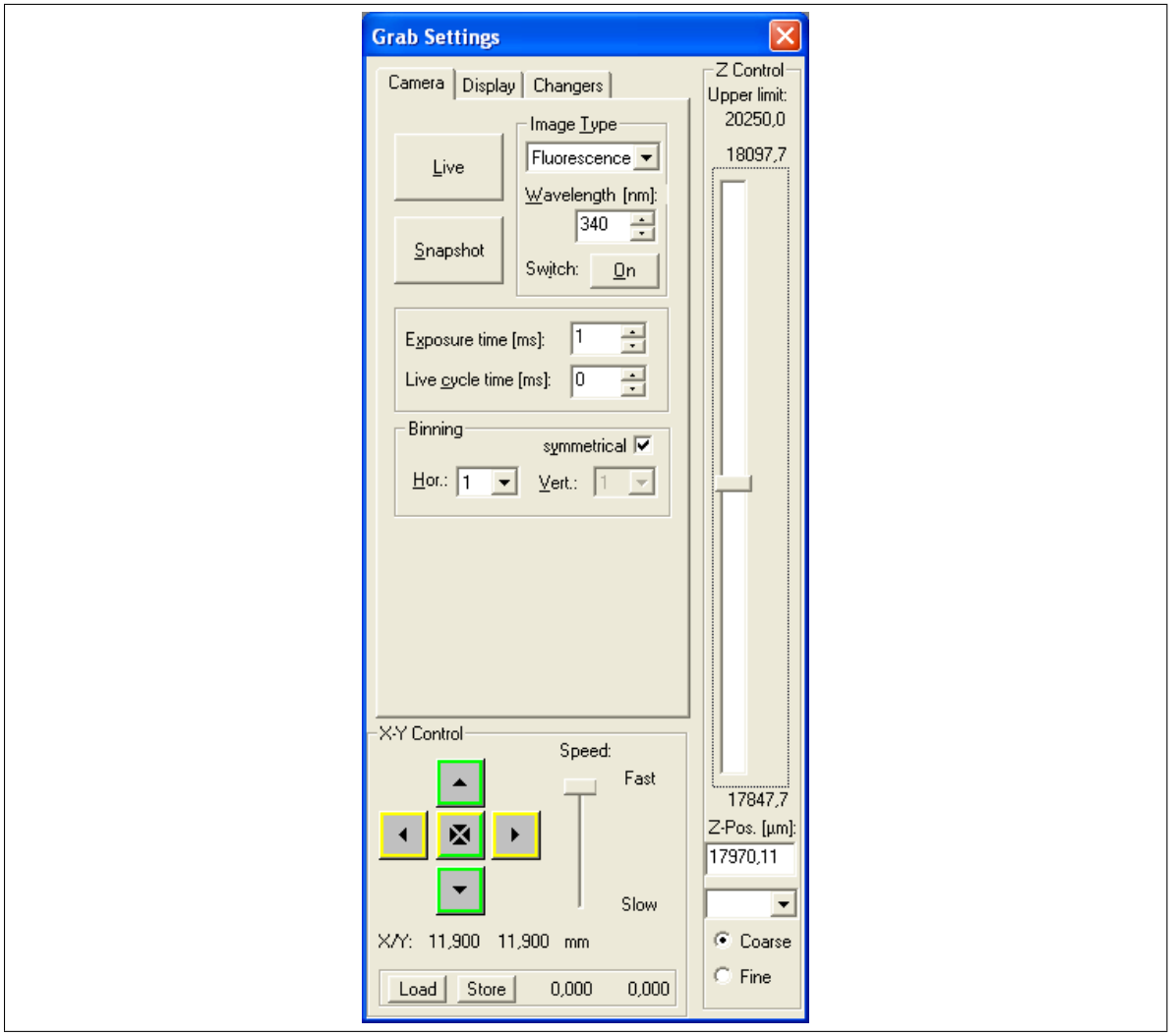


Abbildung 7.8: Eingabemaske zur Bestimmung der Parameter für Bildaufnahmen der PC-Software TILLvisION (Version IV). Im unteren Bereich befinden sich zudem Befehlsschaltflächen zum Navigieren und Fokussieren.

Auf ICU-Ebene unterscheiden sich die beiden Modi Snapshot und Live derzeit nicht: Nach einer allgemeinen Initialisierung der im System verwendeten digitalen, analogen I/O, der EIA-232-Schnittstellen sowie einiger Events wird ein Standardprotokoll aufgesetzt, das in einer Schleife jeweils die Vorgänge zur Aufnahme eines Bildes abarbeitet. Die Kommandosequenz der Initialisierung sowie des Standardprotokolls sind im Anhang in den Auflistungen B.3 und B.4 auf Seite 144 f. abgedruckt. Das Protokoll zur Bildaufnahme besteht aus einer ständig aktiven Endlosschleife, in welcher der Ausführzeitpunkt des ersten Befehls an ein Software-Event gekoppelt ist. Somit kann die Aufnahme eines Bildes durch Senden des Kommandos *SoftEvent* (s. Seite 100) angestoßen werden. Im Falle eines Snapshot wird das Kommando lediglich einmal gesendet, im Falle des Live-Modus werden zu Beginn zwei SoftEvent-Kommandos und unmittelbar nach Auslesen der Bilddaten jeweils ein weiteres gesendet. Das im Kumulativ-Modus verwendete Software Event ist somit immer aktiv, so dass am Ende einer Bildaufnah-

me unmittelbar mit der nächsten begonnen wird. Innerhalb der Schleife wird durch Auslesen der bidirektionalen Synchronisationsleitung zur CCD-Kamera zunächst überprüft, ob diese für eine neue Bildaufnahme bereit ist. Sobald diese Überprüfung positiv verläuft, werden Triggersignale auf entsprechenden digitalen Ausgängen zur Kamera sowie zur Lichtquelle erzeugt. Zusätzlich wird abhängig vom „Warten-Event“ entschieden, ob die aktuelle ICU-Systemzeit an den PC gesendet wird. Kommt es durch eine negative Überprüfung des Kamerastatus zu einer Verzögerung, wird diese hierüber an die PC-Software gemeldet. Zudem können die Bilddaten durch die übermittelte ICU-Systemzeit eindeutig einem Aufnahmezeitpunkt zugeordnet werden.

Neben der Möglichkeit zur Einstellung der Bildaufnahmeparameter (Abbildung 7.8, obere Hälfte sowie die Reiter „Display“ und „Changers“) bietet die Eingabemaske Befehlsschaltflächen zum Navigieren und Fokussieren des Präparats (Abbildung 7.8, untere und rechte Hälfte: X-Y und Z Control). Über den motorisierten xy-Tisch bzw. den motorisiert verfahrbaren Mikroskopsockel kann das Präparat somit softwaregesteuert relativ zum Objektiv bewegt werden. Der zweistufige Fokus kann über die Eingabemaske ebenfalls softwaregesteuerte eingestellt werden. Der Parameter Ausführzeitpunkt der entsprechenden Kommandos ist jeweils als unmittelbar und damit protokollunabhängig angegeben, so dass beide Aktionen eigenständig und parallel zu einem laufenden Protokoll eines Snapshot bzw. der kontinuierlichen Bildaufnahme des Live-Modus ausgeführt werden. Die Auflistung B.3 zur ICU-Initialisierung und die Auflistung B.4 zum Standardprotokoll von Bildaufnahmen veranschaulicht, dass durch die Grundfunktion der PC-Software bereits alle Protokollmechanismen sowie die Mechanismen zum Aufsetzen und Nutzen von Events verwendet werden. Mitsamt der Möglichkeit zum direkten Manipulieren der x-, y- und z-Position werden zudem alle derzeit implementierten Hardwareschnittstellen (digitale, analoge, sowie serielle EIA-232-Schnittstellen) verwendet. Durch diesen Einsatz der ICU wurden somit alle Hardwareschnittstellen, die mit dem Aufsetzen und der Ausführung von Protokollen verbundenen Mechanismen sowie die adaptive, protokollunabhängige Ausführung von Kommandos während der Abarbeitung von zuvor definierten Protokollen im Dauerbetrieb erfolgreich überprüft.

7.3.2 TILLvisION IV - Protokoll Modus

Zum Erstellen von komplexen zeitlich vorherbestimmten Abläufen bietet TILLvisION einen Protokolleditor. Der Benutzer kann hierin eine Sequenz von Aktionen zusammenstellen sowie die Parameter und den Ausführzeitpunkt jedes einzelnen Eintrags bestimmen. Zudem ist - mit Einschränkungen - die Verwendung von Schleifen und die Ausführung in Abhängigkeit von Bedingungen (z. B. nur im 5-ten Schleifendurchlauf ausführen) möglich. Ein Großteil der ICU-Funktionalität wird dem Benutzer somit unmittelbar zu Verfügung gestellt. Für einfaches Einbinden wurden komplexe, geräteübergreifende Aktionen wie z. B. die in Kapitel 6 beschriebene Bildaufnahme unter Mitwirkung der ICU, des Polychrome sowie der CCD-Kamera zu Einheiten zusammengefasst. Somit besteht die komplette Bildaufnahme aus lediglich einem Eintrag im TILLvisION Protokoll bei dem Belichtungsart (Fluoreszenz, Durchlicht, etc.), Belichtungsdauer, sowie der Ausführzeitpunkt angegeben werden müssen.

Unter Verwendung des Protokolleditors wurden Protokolle zum Testen der verschiedenen Systemkomponenten sowie zur Funktionsüberprüfung und für Dauertests des Gesamtsystems erstellt. Weiterhin wurde das protokollgesteuerte Mikroskopiesystem u. a. am BIZ im Rah-

men von biologischen Forschungsprojekten, bei denen das Wachstumsverhalten von Neuronen [110, 111] charakterisiert wurde, eingesetzt. Als Beispiel für die Art der Nutzung des Gesamtsystems wurde der Verlauf der folgenden Beispielanwendung definiert. Ein in der Biologie, der Medizin sowie der Chemie häufig anzutreffendes Ablaufmuster beschreibt die periodische Bildaufnahme eines zuvor ausgewählten Blickfelds. Nach Ablauf eines zeitlichen Vorlaufs wird eine Veränderung der Versuchsbedingungen wie z. B. durch Hinzufügen einer Wirksubstanz herbeigeführt, die durch weitere periodische Bildaufnahme beobachtet und analysiert wird. Dieser Ablauf wurde durch das in Abbildung 7.9 gezeigte Protokoll definiert und am Beispiel eines Versuchs zur Kalzium-Oszillation [27, 102, 17] ausgeführt.

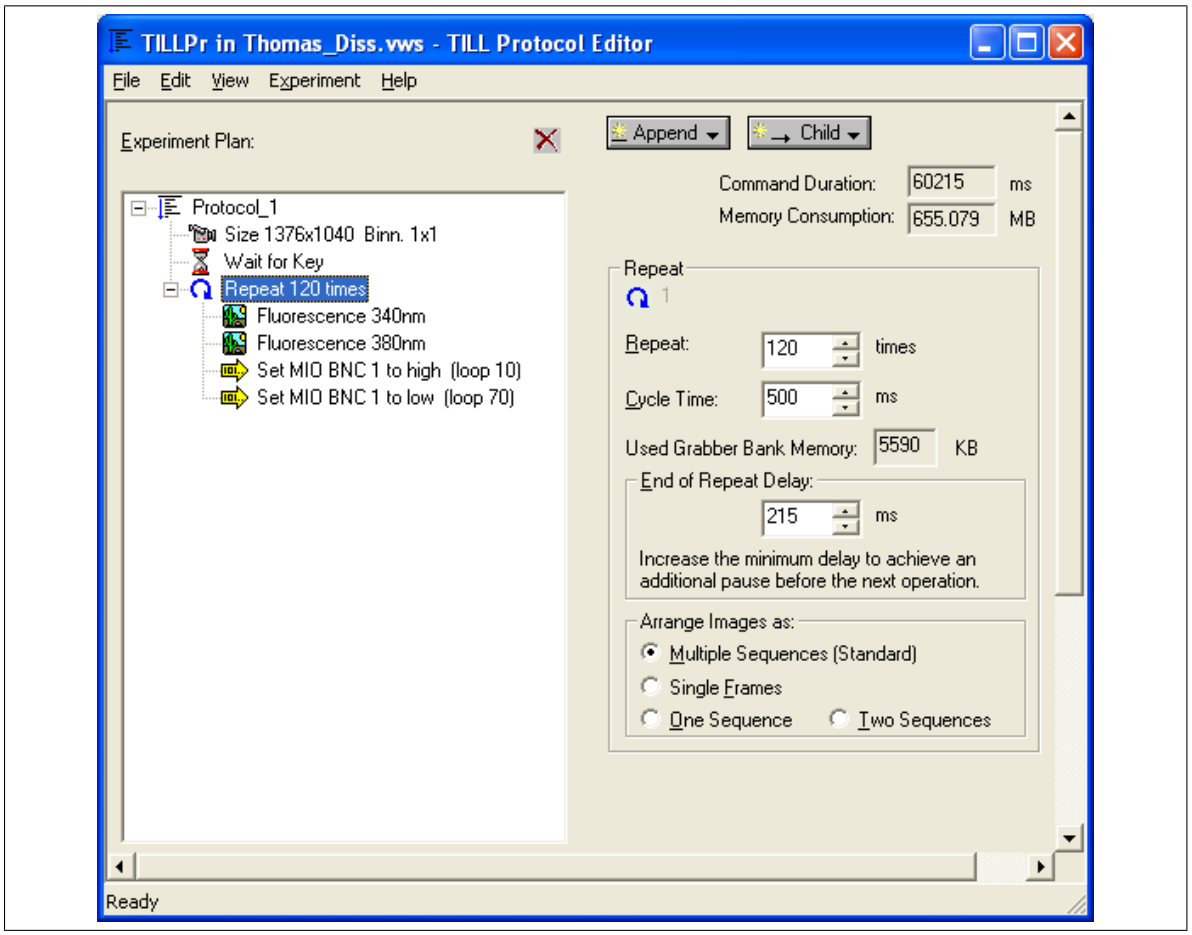


Abbildung 7.9: Eingabemaske zum Erstellen von benutzerdefinierten Protokollen. Einzelne Aktionen können in einer Baumstruktur angeordnet werden um den logischen Fluss zu spezifizieren. Die kommandospezifischen Parameter können für jedes Kommando individuell eingestellt werden.

Mit dem ersten Protokolleintrag wird zunächst die CCD-Kamera für die Bildaufnahme konfiguriert. Der zweite Protokolleintrag führt zu einem Benutzerdialog, der die weitere Ausführung des Protokolls blockiert. Somit startet der eigentliche Versuch mit der ersten Bildaufnahme erst, wenn das Dialogfenster vom Benutzer quittiert worden ist. In einer Schleife werden im Anschluss jeweils 120 Fluoreszenzbilder mit einer Anregungswellenlänge von 340 nm bzw. 380 nm aufgenommen. In der 10. Schleife wird über einen digitalen Ausgang der ICU das Hinzu-Pipettieren der Wirklösung mittels eines elektronisch gesteuerten Quetschventils freigegeben. In der 70. Schleife wird wieder auf das Hinzupipettieren von einfacher Nährlösung umgeschaltet.

Durch die bedingte Ausführung in Abhängigkeit der jeweils durchlaufenen Schleifennummer wird die letzte bisher noch nicht verwendete ICU-Funktionalität - die eingebettete Mathematik - mit eingesetzt. Über die verschiedenen, benutzerdefinierten Ablaufprotokolle wird zudem die flexible Konfigurierbarkeit der ICU überprüft bzw. verifiziert. Kontrollmessungen der Ausführzeiten des beschriebenen Protokolls bestätigen das im vorherigen Abschnitt beschriebene Zeitverhalten der ICU.

7.3.3 Abrastern einer Multititerplatte

Über den im vorherigen Abschnitt vorgestellten Protokolleditor von TILLvisION wurde ein Ablauf zum Abrastern einer Mikrotiterplatte definiert. Dieser kann in folgende logische Teile untergliedert werden:

- Aufnahme einer Zeile eines Wells: Schleife um eine Bildaufnahme mit anschließender (relativer) Positionierung der Probe auf die Position des benachbarten Blickfelds,
- Aufnahme eines kompletten Wells (alle Zeilen): Schleife um eine Bildaufnahme, eine (relative) Positionierung der Probe auf die Anfangsposition der nächsten Zeile und die Aufnahme einer Zeile eines Wells (s. o.),
- Aufnahme einer Reihe von nebeneinander liegender Wells der Multititerplatte: Schleife um die Aufnahme eines Wells mit anschließender (relativer) Positionierung der Probe auf die Ausgangsposition eines neuen Wells,
- Aufnahme aller Wells der Multititerplatte: Schleife um die Aufnahme einer Reihe von nebeneinander liegender Wells (s. o.) und eine (relative) Positionierung der Multititerplatte auf die Anfangsposition der nächsten „Well-Zeile“.

Die Überprüfung des Protokolls wurde im Rahmen eines biologisch relevanten Versuchs durchgeführt: Zur Untersuchung der Zellkernmorphologie bei HEK-293-Zellen wurden diese in einer 24-Wellplatte ausgesät und über einen Zeitraum von ca. 20 Stunden in einem Inkubator kultiviert. Vor den Messungen wurde die Zellkultur mit dem Fluoreszenzfarbstoff (Hoechst-33258), zur Markierung des Zellkerns, eingefärbt. Das verwendete Mikroskopiesystem bestand aus einem iMIC mit verfahrbarem xy-Tisch für Multititerplatten und einem 10x Luftobjektiv[1], einer CCD-Kamera[2] sowie der Beleuchtungsquelle Polychrome V. Der Fluoreszenzfarbstoff wurde mit einer Wellenlänge von 405 nm bei einer Belichtungszeit von 100 ms angeregt.

Die mit Hilfe eines entsprechenden Protokolls aufgenommenen Einzelbilder (18 mal 24 Aufnahmen) wurden unter Verwendung der Software ImageJ[3] zu einem Gesamtbild pro Well zusammengefügt. Abbildung 7.10 zeigt exemplarisch das zusammengefügte Bild *eines* kompletten Wells, einen Bildausschnitt in der Originalvergrößerung, sowie das in TILLvisION erstellte Protokoll zum Rastern eines Wells. Im linken Bildausschnitt sind die Abmessungen des Wells durch den hellen Ring deutlich erkennbar. Der innere Bereich zeigt eine homogene Verteilung von Zellkernen (kleine helle Punkte) bis auf einen kleinen Bereich in der linken Bildhälfte.

[1] Das verwendete Objektiv ist vom Typ Olympus UPlan Apo, 10x/0,40, ∞ /0,17.

[2] Die verwendete Kamera ist vom Typ „SensiCam® QE“ der Firma PCO.

[3] ImageJ ist ein in Java geschriebenes und damit plattformunabhängiges Bildbearbeitungs- und Bildverarbeitungsprogramm.

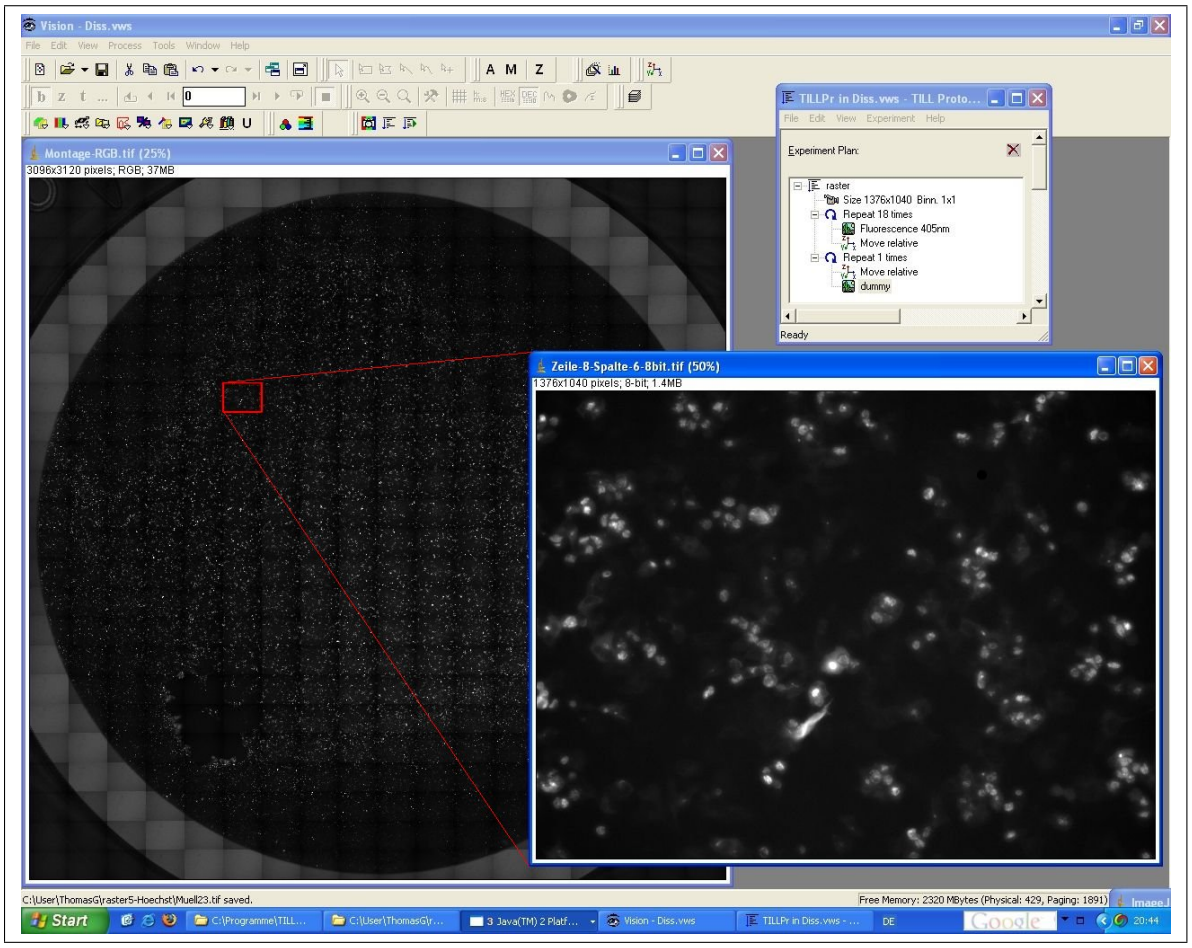


Abbildung 7.10: Gesamtbild eines Wells, dass durch Rasterung einer Mikrotiterplatte bei zehnfacher Vergrößerung aufgenommen wurden (links im Bild); exemplarischer Bildausschnitt des Wells (rechts unten im Bild); Protokollausschnitt zum Abrastern eines Wells (rechts oben im Bild).

Vermutlich wurde der Zellrasen bei der Zugabe des Farbstoffs an dieser Stelle durch die Pipettenspitze beschädigt. Die rechte Hälfte der Abbildung zeigt die Originalaufnahme eines Einzelbilds.

8 Diskussion und Ausblick

Ausgehend von der in Kapitel 1 definierten Zielsetzung dieser Dissertation wurde ein allgemeines Konzept zur echtzeitfähigen Steuerung modular aufgebauter Systeme erarbeitet, das an sich autonome Geräte als Komponenten in ein Gesamtsystem integriert. Das in Kapitel 3 vorgestellte Konzept berücksichtigt nicht nur die Eigenschaften (z. B. verschieden Schnittstellen oder Zeitanforderungen) potenziell eingesetzter Geräte (Bottom-up-Ansatz), sondern auch die Eigenschaften und Anforderungen der angestrebten Steuerung (Top-down-Ansatz) von zeitlich deterministischem, interaktivem bis hin zu adaptivem Systemverhalten. Nach dem erarbeiteten Konzept werden zeitunkritische Prozesse, wie z. B. die Konfiguration der verschiedenen Systemkomponenten über einen Anwender-PC gesteuert. Zeitkritische Prozesse hingegen werden von einer - als zentraler Knotenpunkt für Hardwaresynchronisation eingesetzten - Echtzeitplattform gesteuert, um zeitoptimales Verhalten des Gesamtsystems zu erhalten. Anhand aus dem Konzept abgeleiteter Anforderungen wurde die Echtzeitplattform (**I**ntegration **C**ontrol **U**nit (ICU)) strikt modular konzipiert und ist mit ihren mannigfaltigen Erweiterungsmöglichkeiten vielseitig einsetzbar (s. Kapitel 4). Um langfristig einen flexiblen Einsatz der ICU in bereits existierenden und zukünftig geplanten Systemen (s. Kapitel 2) zu ermöglichen, wurden die in Kapitel 5 vorgestellten Firmwarekonzepte und Lösungen ebenfalls strikt auf die Modularität des Steuerungskonzepts, sowie die verschiedenen Systemverhalten (deterministisch, interaktiv und adaptiv) ausgelegt. Die komplette Funktionalität der ICU wird über eine Kommandoschnittstelle auf die Ebene des Anwender-PCs durchgereicht, so dass das Systemverhalten durch das Verfassen von Ablaufprotokollen flexibel definiert werden kann. Nach der Übertragung eines solchen Protokolls erfolgt die Ausführung zeitrichtig im Rahmen der ICU Zeitauflösung. Die so konzipierte Echtzeitplattform ermöglicht durch die freie Konfigurierbarkeit sowohl der zeitlichen Abläufe als auch der Synchronisationsabhängigkeiten größtmögliche Flexibilität. Die aus dem Einsatz der ICU in derzeitigen Systemen gewonnenen Erkenntnisse sowie die Ergebnisse der Funktionsüberprüfung (s. Kapitel 7) zeigen, dass sich die in dieser Dissertation präsentierte Echtzeitplattform für zeitkritische Anwendungen hardwaresynchronisierter, modularer Systeme im Allgemeinen hervorragend eignet. Im Speziellen bildet die Echtzeitplattform damit die Grundlage für zeitgenaue Hardwaresteuerung eines Mikroskop-basierten Maschine Vision Systems bestehend u.a aus den Komponenten Mikroskop, Kamera, Lichtquelle und Laserscann-Vorrichtungen.

8.1 Vergleich und Abgrenzung zu vorher bestehenden Systemen

Handelsübliche Mikroskopiesysteme sind - bedingt durch ihren traditionellen Aufbau und die ursprüngliche Ausrichtung auf die unmittelbare Bedienung durch einen menschlichen Benutzer - nur bedingt für automatisierte Anwendungen geeignet. Die konventionelle Geometrie der Mikroskopstative, der Okulare, der Halterungen für Durchlichtquellen etc. behindert zum einen automatisiertes Austauschen, Versorgen und Manipulieren der zu untersuchenden Proben und erschwert zudem ein schnelles Umschalten zwischen verschiedenen Mikroskopietechniken bzw. eine optische Probenmanipulation mittels geführter Laserstrahlen (Lasermikrodissektion, optische Pinzette). Die herkömmlichen, vorwiegend in Anwendungen wie der pharmakologischen Wirkstoffsuche eingesetzten, optischen Auslesegeräte, die sog. Reader, sind hingegen zwar voll automatisiert und auf Schnelligkeit sowie einfache Handhabung der Proben optimiert, bieten

aber i. d. R. keine ortsaufgelösten Bilder der untersuchten Proben, sondern integrale Messgrößen, die über ausgedehnte Areale integrieren und lediglich eine Population charakterisieren. Das angestrebte Mikroskope-Machine-Vision-System verwendet daher ein neuartiges, speziell für den Einsatz in automatisierten Anwendungen entwickeltes, inverses Mikroskop Plattformkonzept (iMIC), das durch die Reduzierung auf die wesentlichen optischen Elemente und deren geschickte räumliche Anordnung ein gleichzeitiges Installieren von mehreren Mikroskopietechniken sowie schnelles Umschalten zwischen den verschiedenen Techniken während der Laufzeit möglich macht. Zudem ist der Raum oberhalb des Objektivs komplett frei für die Handhabung und Manipulation der Proben. Die am iMIC betriebenen Laser-Scanning-Techniken, die innerhalb kontrastreicher zwei- und dreidimensionaler Bildgebung und bei optischer Probenmanipulation (z. B. optische Pinzette, Dissektion) eingesetzt werden, verwenden neuartige digitale Reglersysteme, deren Entwicklung ursprünglich aus den Leistungsansprüchen des BIZ angestoßen wurden. Die in den Reglersystemen eingesetzte DSP-Plattform hat sich im Arbeitsumfeld dieser Dissertation zudem in weitern Projekten bewährt, so dass die Plattform ebenfalls als zentrale Recheneinheit in der Steuerungshardware der Echtzeitplattform ICU zu verwenden war. Damit wurde statt auf ein PC-basiertes Echtzeitsystem bzw. einer echtzeitfähigen PC-Erweiterung auf ein bestehendes schlankes Echtzeitbetriebssystem auf Basis der DSP-Plattform aufgebaut. Die Hardware konnte damit optimal auf das erarbeitete Konzept hin ausgelegt und den daraus entstandenen Anforderungen angepasst werden. Verglichen mit zum Beginn der Arbeiten zu dieser Dissertation existierenden Steuerung eines Imaging-Systems wurde die Hardware strikt modular und erweiterbar konzipiert und damit dem modalen Charakter des Steuerungskonzepts angepasst. Bei der Programmierung der Firmware wurde im Gegensatz zum bestehenden System neben der Unterstützung der modularen Eigenschaften der Hardware vor allem auf die Möglichkeit zur flexiblen Gestaltung der Ablaufkontrolle geachtet. So wurde als konsequente Umsetzung des Steuerungskonzepts anstelle von vordefinierten Kontrollszenarien die Möglichkeit zur komplett freien Konfiguration in Form von Protokollen geschaffen. Die Struktur der Protokolle und die Möglichkeiten zum Beschreiben von Abläufen orientieren sich dabei am Aufbau und den Eigenschaften höherer Programmiersprachen. Der Ausführzeitpunkt jedes Protokollkommandos kann dabei individuell zeitlich oder ereignisabhängig bestimmt werden. Abgesehen von den allgemeinen Möglichkeiten der Hardwaresynchronisation von modularen Systemen bildet die ICU als zentraler Bestandteil einer automatisierten Mikroskopieplattform die Basis für die Steuerung zeitrichtiger Abläufe, wie sie für ein Mikroskop-basiertes Machine-Vision-System für die Korrelation der aufgenommen Bilddaten unabdingbar sind.

8.2 Interpretation und Bewertung der Ergebnisse aus Kapitel 7

Bedingt durch den konzeptionellen Aufbau der ICU Hard- und Firmware beziehen sich die Anforderungen an die Echtzeit des Systems prinzipiell auf zwei kritische Gruppen: Die Abhandlung der Hardwareschnittstellen einerseits und die der Protokollabarbeitung andererseits. Für die Gewährleistung der Echtzeitfähigkeit der ICU ist es entscheidend, dass die Summe des Zeitbedarfs beider Gruppen innerhalb eines Zyklus die spezifizierte Basis der Echtzeit nicht überschreitet (vgl. Abschnitt 5.7):

- *Abhandlung der Hardwareschnittstellen*: Alle Hardwareschnittstellen werden durch die zyklisch laufende TMSI betrieben. Wie in Abschnitt 7.1.1 gezeigt wurde, ist der Zeitbe-

darf hierfür bis auf eine Abweichung von 4,15 % konstant. Durch den Aufbau der TMSI ist hardwareseitig automatisch gewährleistet, dass Daten innerhalb der Dauer eines Zyklus bzw. innerhalb des Echtzeitrasters vom DSP an die einzelnen elektronischen Bausteine (et vice versa) übertragen werden.

- *Abarbeitung eines Protokolls*: Die durch den Benutzer formulierten Protokolle werden zur Ausführung auf die ICU übertragen und dort durch einen entsprechenden Mechanismus ausgeführt (vgl. Abschnitte 5.3 ff.). Nach dem in Abschnitt 5.7 vorgestellten Zeitplanungsmodel steht für die Abarbeitung von Protokollen maximal die nach der Bedienung der Hardwareschnittstellen verbleibende Rechenzeit eines TMSI-Zyklus zur Verfügung. Die tatsächlich für die Abarbeitung eines Protokolls innerhalb eines Zyklus benötigte Zeit ist selbstverständlich vom durch den Benutzer vorgegeben Protokoll abhängig. Die Verantwortung für die Gewährleistung, dass ein Protokoll tatsächlich in Echtzeit abgearbeitet wird, kann daher nicht auf Seiten der ICU-Firmware abgefangen werden, sondern liegt vielmehr auf der Seite des Benutzers.

Ist aufgrund einer unsachgemäßen Protokolldefinition die in Summe benötigte Zeit größer als die Dauer eines TMSI-Zyklus, führt dies lediglich dazu, dass die ansonsten garantierte maximale Verzögerung (die Echtzeitbasis) eines Kommandos nicht eingehalten werden kann. Sobald alle laut Protokoll auszuführenden Kommandos abgearbeitet sind, werden weitere, zeitlich korrekt formulierte Protokollteile wieder bis auf das Echtzeitraster zeitgenau ausgeführt.

Die gleichzeitige Anforderung von garantierter Echtzeit und programmierbarer Flexibilität ist an sich ein Widerspruch. Jedes Echtzeitsystem bietet für die Dauer der Echtzeitbasis eine bestimmte, systemeigene Rechenkapazität, mittels derer alle geplanten Aufgaben zu erledigen sind. Bietet man, wie im Fall der hier vorgestellten Echtzeitplattform ICU, dem Benutzer die Möglichkeit die in Echtzeit zu erledigenden Aufgaben selbst zu bestimmen, besteht generell die Gefahr, dass der Bedarf der daraufhin spezifizierten Aufgaben größer als die zur Verfügung stehende Rechenkapazität ist. Um die Leistung der ICU in benutzerdefinierten Protokollen voll ausnutzen zu können ohne das Echtzeitverhalten zu beeinträchtigen, wurde die Ausführdauer aller derzeit möglichen Protokollkommandos ermittelt (s. Abschnitt 7.1). Durch die Angabe der im Einzelnen benötigten Rechenzeiten besteht somit bereits bei der Definition von Protokollen die Möglichkeit, korrektes Zeitverhalten zu garantieren.

8.3 Mehrwert

Die Komplexität spezialisierter Geräte sowohl in experimentellen als auch in angewandten Naturwissenschaften nimmt stetig zu. Oftmals werden verschiedene, bereits etablierte Einzelgeräte zu einem Gesamtsystem zusammengefügt, wobei jedes Gerät einen Teilbereich der Gesamtfunktionalität abdeckt. Damit in zeitkritischen Anwendungen der Mehrwert des Systems die Summe derer der Einzelteile übertrifft, ist eine zeitgenaue Koordination aller Systemkomponenten notwendig. An dieser Stelle ansetzend wurde mit dem in dieser Dissertation vorgestellten Steuerungskonzept und seinem zentralen Element für hardwaresynchronisierte Abläufe eine universell einsetzbare Basis geschaffen, mittels der modulare, aus autarken Einzelgeräten bestehende Systeme, echtzeitfähig betrieben werden können.

Im Konkreten wird die hier geschaffene Echtzeitplattform innerhalb von Systemen zur automa-

tisierten Mikroskopie eingesetzt. Hierbei kommt es einerseits auf die Koordination von im Vorfeld beschriebenen, präzisen und zeitlich hoch aufgelösten Vorgängen und andererseits auf die optimierte wechselseitige Synchronisation des Zusammenspiels von Systemkomponenten an. Besonders im Hinblick auf das angestrebte Machine-Vision-System, das auf Basis der automatisierten Mikroskopie geschaffen werden soll, bieten das Steuerungskonzept und die Echtzeitplattform Möglichkeiten im laufenden Betrieb zwischen deterministischem und adaptivem Verhalten zu wechseln. Somit wurden erstmals die hardwareseitigen Grundlagen für Mikroskop-basiertes Machine-Vision geschaffen: Vordefinierte Abläufe können mittels aus Bildverarbeitung gewonnener Erkenntnisse online abgeändert bzw. angepasst werden.

Verglichen mit händisch betriebenen Mikroskopiesystemen ermöglichen automatisierte Systeme technisch korrekte und 100 % reproduzierbare Versuche. Unbekannte Randbedingungen werden somit normiert und ergebnisbeeinflussende Fremdeinflüsse durch anwenderbedingte Ungenauigkeiten minimiert. Dies führt zu einer Effizienzsteigerung sowohl innerhalb der biologischen bzw. medizinischen Lebendzellforschung als auch innerhalb der Anwendungen im Bereich des Pharma-Screenings. Durch die Steigerung der Effizienz kann erwartet werden, dass man qualitativ hochwertige Ergebnisse schneller erhält, wodurch sich das, insbesondere in der Pharmaindustrie wichtige, Kosten-Nutzen-Verhältnis verbessert. Zudem werden durch automatisierte, Mikroskop-basierte Analyse- oder Machine-Vision-Systeme neue Methoden bzw. Assays durch die Kombination von Bildaufnahme und Bildanalyse bzw. Bilderkennung mit anschließender Anpassung der weiteren Abläufe überhaupt erst ermöglicht. Die vorgestellten technischen Konzepte und Lösungen stellen somit eine Basistechnologie (Enabling Technology) für neue Verfahren und Anwendungen innerhalb der Life Sciences dar.

8.4 Ausblick

Die ICU-Echtzeitbasis hängt in erster Linie von der TMSI-Zykluszeit ab, die derzeit 34 µs beträgt. Ausführzeitpunkte von in Protokolldefinitionen abgelegten Kommandos können derweil in Anlehnung an die Zeitbasis der Peripherie für Laser-Scanning-Anwendungen auf eine Genauigkeit von 10 µs angegeben werden. Die in Abschnitt 5.7 beschriebenen Aufgaben 1 und 2 können somit lediglich quasisynchron abgearbeitet werden, was zum derzeitigen Jitter von 34 µs führt (s. Abschnitt 7.2). Zukünftige Arbeiten könnten sich daher auf ein weiteres Herabsetzen der Echtzeitbasis bis auf eine Zykluszeit von 10 µs konzentrieren.

In einem ersten Ansatz kann ein Performancegewinn erreicht werden, indem die Taktrate der physikalischen Schnittstelle zur Hardware, der TMSI, entsprechend erhöht wird. Da die maximale Taktrate allerdings durch den angeschlossenen Baustein mit der niedrigst möglichen Taktrate limitiert wird (derzeit die EIA-232-Schnittstellenbausteine mit 4 MHz), bedarf diese Änderung entweder einer Neuauswahl der verwendeten Bausteine mit verhältnismäßig schnelleren Schnittstellen oder einer Umsetzung der Taktrate auf eine entsprechend langsamere. Letzte Methode zur Unterstützung von Bausteinen, deren Schnittstellen geringere Taktraten zulassen als die der TMSI, könnte durch eine CPLD-Programierung erfolgen. Die Erhöhung der TMSI-Taktrate setzt allerdings lediglich die Gesamtzykluszeit herab. Der Zeitbedarf von Aufgabe 1 bzw. der in Aufgabe 2 abgearbeiteten Protokollkommandos bleibt unverändert. Das Herabsetzen der Zykluszeit bei gleichbleibendem Zeitbedarf von Aufgabe 1 geht daher komplett zu Lasten der für Aufgabe 2 (und 3) zur Verfügung stehenden Zeit (s. Abschnitt 5.7).

Neben der weiteren Optimierung der in Aufgabe 2 abgearbeiteten Protokollkommandos besteht durch Einsatz des ursprünglich nicht zur Verfügung stehenden parallelen Interfaces des DSP die Möglichkeit den Zeitbedarf von Aufgabe 1 zu reduzieren. Hierfür müsste die DSP-Schnittstelle zur Hardware TMSI auf einen zweiten Baustein ausgelagert werden (z. B. auf einen Mikroprozessor oder einen programmierbaren Logikbaustein). Über das parallele Interface erfolgt der Datenaustausch zwischen dem DSP und diesem Erweiterungsbaustein, von wo aus die Daten über eine emulierte TMSI an die eigentlichen Schnittstellenbausteine (DAC, ADC, etc.) weitergeleitet werden. Allerdings ist diese Änderung mit erheblichem Aufwand sowohl bzgl. der Hardwareänderungen als auch bzgl. der Programmierung des Erweiterungsbausteins verbunden. Durch die Beschränkung der Nutzung der DSP-eigenen TMSI auf die Übertragung rein zyklischer Daten, reduziert sich der Zeitbedarf von Aufgabe 1 entsprechend.

Der Umfang der Protokollfunktionalität beinhaltet bereits Elemente höherer Programmiersprachen wie Variablen, mathematische Operationen, Schleifen und bedingte Ausführung. Eine mögliche Erweiterung dieses Funktionsumfangs wäre durch die Möglichkeit gegeben, mehrere Instanzen der Protokollstruktur anzulegen. Hierdurch könnte der Benutzer einzelne Teilabläufe zusammenfassen und in Form von Funktionen auf der ICU hinterlegen, die dann aus dem Hauptprotokoll heraus aufgerufen werden könnten. Eine zusätzliche Erweiterung wäre durch das Abspeichern von solchen Funktionen im ICU-internen, nichtflüchtigen Flash-Speicher gegeben. Ständig benötigte Protokolle bzw. Protokollteile müssten somit nicht stets neu erstellt und vom Anwender-PC auf die ICU übertragen werden. Über automatisches Laden und Starten von im Vorfeld abgespeicherten Protokollen unmittelbar nach dem Einschalten könnte die ICU zudem als eigenständiges Embedded System arbeiten. Dies könnte genutzt werden um autarke Anwendungen ohne Anwender-PC zu unterstützen.

Abgesehen von den möglichen Maßnahmen zur Steigerung der Performance konzentrieren sich die derzeitigen Arbeiten auf die Etablierung der ICU in weiteren Anwendungen auf dem Migrationspfad hin zu Mikroskop-basiertem Maschine Vision. Unmittelbar wird an einem System zum automatisierten Abrastern von Mikrotiterplatten mit kontinuierlicher Anpassung der Fokusposition gearbeitet. Die Ausbaustufe mit bildbasiertem Autofokus stellt bereits eine einfache Maschine Vision Anwendung dar. Weiterhin wird die Integration der Mikroskopsteuerung und der verschiedenen Laser-Scanning-Module vorangetrieben, um die bisher parallel bearbeiteten Projektteile zusammenzuführen. Die damit in einer Anwendung zur Verfügung stehenden Mittel des regulären Imaging, der Laser-Scanning-basierten Bildgebung und der optischen Probenmanipulation führen in Verbindung mit der Erweiterung der online Bildauswertung schrittweise zu stetig anspruchsvolleren Machine-Vision-Anwendungen.

A Glossar

ActiveX

ActiveX bezeichnet ein Softwarekomponenten-Modell von Microsoft für aktive Inhalte. Diese Inhalte sind Software-Komponenten für andere Anwendungen, Makroprogrammierungen und Entwicklungsprogramme, welche gleichermaßen in verschiedenen Programmiersprachen und Umgebungen verwendet werden können. Einige Programme nutzen z. B. den Internet Explorer zur Anzeige von Informationen.

Adhärenz

lateinisch: Anhaftung: Innerhalb der Biologie bezeichnet die Zell-Adhärenz das Anhaften von Zellen an z. B. einem Objektträger oder dem Boden einer Mikrotiterplatte. Die verschiedenen Mechanismen der Anhaftung werden durch die molekular Adhäsion zweier verschiedener Stoffe beschrieben (vgl. Adhäsion).

Adhäsion

lateinisch: Verklebung: Die Adhäsion (auch Anhangskraft) bezeichnet die Zusammenhangskräfte zwischen den Molekülen zweier verschiedener Stoffe bzw. das Haften zweier Stoffe oder Körper aneinander. Unter Zelladhäsion versteht man in der Zellbiologie die Kontakte zwischen Zellen. Diese können in einem Gewebe oder in einem Aggregationsverband vorliegen, der durch aktives Zusammenwandern von Einzelzellen oder durch passives Zusammenstoßen und Zusammenkleben von Zellen und Zellklumpen erfolgt.

API

Ein **A**pplication **P**rogram **I**nterface (API) (deutsch: Schnittstelle zur Anwendungsprogrammierung) ist eine Schnittstelle, die anderen Programmen von einem Softwaresystem zur Anbindung des Systems zur Verfügung gestellt wird. Im Gegensatz zu einer Binärschnittstelle (ABI) definiert eine API nur die Verwendung der Schnittstellen auf Quelltextebene. Neben dem Zugriff auf Datenbanken, die Hardware wie Festplatte oder Grafikkarte kann eine API auch das Erstellen von Komponenten der grafischen Benutzeroberfläche ermöglichen oder vereinfachen.

Assay

Als Assay bezeichnet man v.a. in der Labormedizin einen standardisierten Reaktionsablauf zum Nachweis einer Substanz mit einer spezifischen Methode. Aufgebaut ist ein Assay grundsätzlich aus einem mit einem spezifischen Antikörper beschichteten Träger (Reagenzglas, MikroTiterplatte o.ä.) und einem unspezifischen Antikörper, der mit einer gut nachweisbaren Substanz konjugiert ist. Der spezifische Antikörper reagiert dabei nur mit der gesuchten Substanz (Hormone, Proteine). Der unspezifische Antikörper reagiert mit dem Protein des spezifischen Antikörpers.

Assemblercode

Eine Assemblersprache ist eine spezielle Programmiersprache, welche die Maschinensprache einer spezifischen Prozessorarchitektur in einer für den Menschen lesbaren Form repräsentiert.

Jede Computerarchitektur hat folglich ihre eigene Assemblersprache. Ein Programm in Assemblersprache wird auch als Assemblercode bezeichnet. Es wird durch einen speziellen Compiler, ebenfalls Assembler genannt, in direkt ausführbare Maschinensprache (auch Maschinencode) umgewandelt. [10]

Boot-Loader

Ein Boot-Loader (verkürzte Form des ursprünglichen Wortes bootstrap loader) ist eine spezielle Software, die gewöhnlich durch die Firmware (resp. BIOS bei IBM-kompatiblen PCs) eines Rechners von einem bootfähigen Medium geladen und anschließend ausgeführt wird. Der Boot-Loader lädt dann weitere Teile des Betriebssystems bzw. die Hauptanwendung (in eingebetteten Systemen).

Cellomics

Cellomics, oder auch „Systems biology“ genannt, beschreibt ein akademisches Feld, das bestrebt ist über die Zusammenführung verschiedenster Informationsquellen komplexe biologische Systeme zu erklären. Bei der Untersuchung der Beziehungen und der Wechselwirkungen zwischen verschiedenen Teilen eines biologischen Systems (z. B. Gene, proteinbasierte Signalwege, Metabolismus, Plasmabildung, Zellen, Gewebe oder ganze Organismen) ist das Ziel letztendlich ein verständliches Model des gesamten Systems zu entwickeln. Damit baut das Gebiet der Cellomics logisch auf dem der Genomics und dem Proteomics auf. Zurzeit befinden sich die Forschungen auf dem Gebiet der Cellomics in den Anfängen und bedienen sich daher überwiegend Methoden der Computersimulation und der Heuristik. [175, 69, 26]

Computer Vision

Der Begriff Maschinelles Sehen (englisch Computer Vision) beschreibt im Allgemeinen die computergestützte Lösung von Aufgabenstellungen, die sich an den Fähigkeiten des menschlichen visuellen Systems orientieren. Vor allem werden maschinell sehende Systeme derzeit in industriellen Herstellungsprozessen in den Bereichen Produktautomatisierung und Qualitätssicherung eingesetzt. Weitere Einsatzgebiete finden sich z. B. in der Verkehrstechnik – von der einfachen Radarfalle bis hin zum „sehenden Fahrzeug“ – und in der Sicherheitstechnik (Zugangskontrolle, automatische Erkennung von Gefahrensituationen).

DMA

Der Begriff **D**irect **M**emory **A**ccess (DMA) (deutsch: Speicherdirektzugriff) bezeichnet in der Computertechnik eine Zugriffsart, die direkt auf den Speicher zugreift. Dabei erlaubt die DMA-Technik angeschlossenen Peripheriegeräten ohne Umweg über den zentralen Rechenkern eines Prozessors direkt mit dem Arbeitsspeicher zu kommunizieren. Der Vorteil der DMA-Technik ist die schnellere Datenübertragung bei gleichzeitiger Entlastung des Prozessors.

Driver

Ein Gerätetreiber, häufig kurz nur Treiber (englisch: Driver) genannt, ist ein Programm-Modul, welches entweder als Bestandteil des Betriebssystems (Kerneltreiber) oder als ein davon unabhängiges Modul (Modultreiber) anderen Programmen ermöglicht, über eine standardisierte Softwareschnittstelle auf eine angeschlossene Hardwarekomponente zuzugreifen. Auf diese Weise kommen die Programme mit jeder Hardware zurecht, für die es einen Treiber gibt.

FLIP (Fluorescence Loss In Photobleaching)

Als FLIP bezeichnet man die Abschwächung oder das Verschwinden der Fluoreszenz in einem definierten Bereich, der an einen mehrfach ausgebleichten Bereich angrenzt. Ebenso wie FRAP wird auch FLIP dazu verwendet, die Mobilität von Proteinen in Membranen oder lebenden Zellen zu messen. [162, 145]

Fluoreszenzmikroskopie

Die Fluoreszenzmikroskopie untersucht die Fluoreszenz von Objekten, also selbstleuchtende Objekte. Sie beruht auf der konventionellen Lichtmikroskopie, hat aber noch zusätzliche Komponenten. Das Objekt wird mit Licht einer bestimmten spektralen Verteilung beleuchtet. Enthält es fluoreszierende Stoffe, werden diese zum Eigenleuchten angeregt und geben Licht mit bestimmten genau definierten Wellenlängen ab. Das abgestrahlte Licht (Fluoreszenzlicht) ist immer langwelliger als das Beleuchtungslicht. Um das sehr schwache Fluoreszenzlicht vom sehr viel stärkeren Beleuchtungslicht zu trennen, werden entsprechende Filter in den Beobachtungsstrahlengang gesetzt. Bei Untersuchungen mit einem Fluoreszenzmikroskop werden zu untersuchende Präparate i. d. R. mit fluoreszierenden Stoffen angefärbt.
Die Fluoreszenzmikroskopie wird meist in der Biologie eingesetzt, um fluoreszierende Bestandteile von Lebewesen (man spricht von Autofluoreszenz) nachzuweisen und zu lokalisieren (z. B. Chlorophylle). Darüber hinaus lassen sich Bestandteile von Lebewesen mit ungiftigen Fluoreszenzfarbstoffen (Fluorochromen) anfärben und so markieren. Das erlaubt die In-Vivo-Verfolgung von Stoffwechselvorgängen.

FRAP (Fluorescence Recovery After Photobleaching)

FRAP bezeichnet die Regenerierung der Fluoreszenz in einem definierten Probenbereich nach einem Ausbleichvorgang. Der FRAP-Effekt entsteht, indem ungebleichte Fluorochrome aus der Umgebung in den gebleichten Bereich wandern. FRAP dient zur Messung der Mobilität von Molekülen, z. B. bei Diffusions-, Transport- oder anderen Bewegungen fluoreszenzmarkierter Moleküle in Membranen oder lebenden Zellen. [181, 105, 22, 126]

FRET (Fluorescence Resonance Energy Transfer)

FRET ist die strahlungsfreie Übertragung von Photonenenergie von einem angeregten Fluorophor (dem Donor) auf ein anderes Fluorophor (den Akzeptor), wenn der Abstand zwischen beiden nicht mehr als 1-10 nm beträgt. Diese Energieübertragung erfolgt strahlungsfrei, im wesentlichen durch eine Dipol-Dipol-Wechselwirkung zwischen Donor und Akzeptor. Der Akzeptor gibt die übertragene Energie in Form von Strahlungsenergie ab. Durch die Detektion und Beobachtung dieser Stahlung kann mittels FRET somit die relative Nähe der Moleküle über die optische Grenze der Lichtmikroskopie hinaus aufgelöst werden. Nachweisbar sind damit z. B. molekulare Wechselwirkungen zwischen zwei Proteinpartnern, Strukturänderungen innerhalb eines Moleküls (u. a. Enzymaktivität oder DNA/RNA-Konformation) oder Ionenkonzentrationen. [156, 124, 137, 70, 15]

Genomik

Die Erforschung des Genoms und die Wechselwirkung der darin enthaltenen Gene wird als Genomik bezeichnet (englisch: Genomics). [147, 90, 28, 19, 4]

HCS (High Content Screening)

HCS ist eine Methode zur automatisierten und damit schnellen und parallelisierten Zellbiologie basierend auf den Gebieten der Optik, Chemie, Biologie und der Bildverarbeitung. HCS wird überwiegend innerhalb der biologischen Forschung sowie der pharmakologischen Wirkstoffsuche eingesetzt. Durch den Einsatz von lebenden, zellulären Testobjekten erstreckt sich die Analyse sowohl räumlich aufgelöst als auch zeitlich aufgelöst und berücksichtigt bei der Ergebnisfindung daher wesentlich mehr Informationen als so genannte Einzelpunktmessungen. HCS ist die Kombination moderner Zellbiologie mit wohl etablierten molekularen Analysewerkzeugen, voll automatisierter hoch auflösender Mikroskopie und roboterbasierter Experimentdurchführung. [38, 25, 41]

HTS (High Throughput Screening)

HTS ist eine Methode zur automatisierten und damit schnellen und parallelisierten wissenschaftlichen Versuchsdurchführung. HTS wird überwiegend für das Screening (deutsch: Sichten im Sinne von Aussieben) von Substanzen auf molekularer Ebene und damit innerhalb der pharmakologischen Wirkstoffsuche sowie den dazugehörigen Forschungsfeldern der Biologie und der Chemie eingesetzt. Durch die Verwendung von vollautomatisierten Auslesegeräten (sog. Reader) ist ein einziger Wissenschaftler in der Lage bis zu 100.000 und mehr Proben pro Tag zu untersuchen. Das Ergebnis eines solchen Screens sind positiv getestete Substanzen, die eine zuvor aufgestellte Hypothese unterstützen. Von diesem Ergebnis ausgehend werden die verbleibenden Substanzen i. d. R. in weiteren, komplexeren Test z. B. mittels HCS-Methoden an lebenden Zellen eingehender untersucht.

in silico

„in silico“ (lateinisch: in Silizium) bezeichnet Vorgänge, die im Computer ablaufen. Der Begriff ist eine Anspielung auf die Tatsache, dass die meisten heutigen Computer-Chips auf der Basis des chemischen Elements Silizium hergestellt sind.
Der Begriff ist im Umfeld der Bioinformatik entstanden, die eine Computerunterstützung zur Aufklärung von biochemischen Prozessen in lebenden Organismen anbietet, insbesonderen den Körperzellen der Organe des Menschen. Durch computergestützte Simulation der zugehörigen biochemischen Prozesse können nun Experimente im Computer angestoßen werden, und die errechneten Resultate wie andere experimentelle Beobachtungen gehandhabt werden - man spricht dann von einem Experiment „in silico“.

in-vitro

„in-vitro“ (lateinisch: im Glas) bezeichnet Vorgänge, die außerhalb des lebenden Organismus stattfinden, im Gegensatz zu solchen, die im lebenden Organismus „in-vivo“ ablaufen. Medizinische Forschung lässt sich in den ersten Schritten deutlich billiger, einfacher und kontrollierter im Reagenzglas (in-vitro) durchführen. Die dabei gewonnenen Erkenntnisse sind jedoch nicht unbedingt auf die Vorgänge in der Natur übertragbar. Daher wird man diese i. d. R. mit einer weiteren Versuchsreihe („in-vivo“) überprüfen müssen.

in-vivo

„in-vivo“ (lateinisch: im Lebenden) bezeichnet Prozesse, die im lebenden Organismus ablaufen. Im Gegensatz dazu werden Abläufe, die im Reagenzglas oder ganz allgemein außerhalb lebender Organismen stattfinden, mit dem Begriff „in-vitro“ belegt.

Inkubator

Ein Inkubator ist ein Gerät, mit dessen Hilfe kontrollierte Außenbedingungen für diverse Brut- und Wachstumsprozesse geschaffen und erhalten werden können. Im Speziellen erzeugt ein Inkubator ein Mikroklima mit eng geregelter Luftfeuchtigkeit und -temperatur. Inkubatoren werden in der Zell- und Mikrobiologie für in Petrischalen oder Mikrotiterplatten gehaltene Zell- und Bakterienkulturen verwendet.

Inverses Mikroskop

Bei den inversen Auflichtmikroskopen stellt der Objekttisch üblicherweise den höchsten Punkt des Gerätes dar. Das Objekt liegt mit der zu mikroskopierenden Fläche nach unten auf dem Objektteller und wird auch von unten bestrahlt. Ansonsten sind Beleuchtungs- und Abbildungsstrahlengang prinzipiell gleich, lediglich um 180° gedreht. In der Handhabung der beiden Geräte-Typen ergeben sich damit in der Praxis primär folgende Unterschiede:

- Die Größe der Objekte ist bei inversen Auflichtmikroskopen nur durch die Tragfähigkeit des Objekttisches begrenzt. So können größere Objekte untersucht werden, ohne sie zerkleinern (und damit eventuell zerstören) zu müssen.
- Die Objekte liegen bei inversen Auflichtmikroskopen immer plan und im rechten Winkel zum Objektiv.
- Die Objektive sind bei inversen Auflichtmikroskopen deutlich exponiert, das heißt, herunterfallende Objekte oder aus dem Objekt tropfende Flüssigkeiten können die Frontlinse beschädigen. Auch stauben die Objektive inverser Auflichtmikroskope deutlich leichter ein.

Konfokalmikroskopie

Ein Konfokalmikroskop ist eine Variante des Lichtmikroskopes, das optische Schnitte mit mikroskopischer Auflösung in räumlich ausgedehnten Objekten erzeugen kann. Mit einem Computer können diese Schnittbilder schichtweise zu einer räumlichen Darstellung zusammengesetzt werden. In einem „normalen“ Lichtmikroskop ist das Bild eine Überlagerung aus einer scharfen Abbildung der Punkte im Fokus und einer unscharfen Abbildung der Punkte außerhalb. In einem Konfokalmikroskop dagegen passiert das vom Präparat ausgehende Licht (reflektiertes, transmittiertes oder auch Fluoreszenzlicht) eine Lochblende oder einen schmalen Schlitz. Dadurch wird das Licht von außerhalb der Fokusebene ausgeblendet und erreicht nicht den Detektor (meist ein Photomultiplier oder ein CCD-Chip). Das Ergebnis ist eine deutliche Reduzierung des Streulichts. Die effektive Auflösung ist dadurch auch ohne Zunahme der Vergrößerung viel höher als in einem herkömmlichen Lichtmikroskop. Aus diesem Grund ermöglicht dieses konfokale Prinzip z. B. die Untersuchung kleiner Details von Zellen in Gewebeschnitten.

Lab-on-a-Chip

Der Begriff Westentaschenlabor oder Chiplabor (englisch lab-on-a-chip device), gelegentlich auch Labor-auf-dem-Chip, bezeichnet ein mikrofluidisches System, das die gesamte Funktionalität eines großen Labors auf einem nur plastikkartengroßen Kunststoffsubstrat unterbringt.

Auf dem Westentaschenlabor finden umfangreiche biologische, chemische und physikalische Prozesse statt. Häufig werden für Westentaschenlabore Silizium oder Borosilikatglas verwendet. Von Bedeutung ist die Platzsparsamkeit der Westentaschenlabore, da auf geringstem Raum komplexe Prozesse ablaufen können.

Laser-Scanning

Der Begriff Laser-Scanning (deutsch: Laserabtastung) bezeichnet die Abtastung von Oberflächen oder Körpern mittels Lasertechnologie und wird u. a. innerhalb der Mikroskopie (konfokale Laser-Scanning Mikroskopie) eingesetzt. Das Prinzip des konfokalen Laserscanninges beruht darauf, dass ein fokussierter Laserstrahl über eine Probe gescannt wird und das zurückfallende Licht hinter einer kleinen Punktblende detektiert wird. Durch die Anordnung der Blende wird nur Licht aus der Brennebene detektiert und man erhält ein Schnittbild nur aus dieser Ebene. Wie dick diese Ebene ist hängt von der Schärfentiefe des verwendeten Mikroskops ab. Ändert man zwischen einzelnen Aufnahmen die Fokussierung, so kann man einen ganzen Bildstapel aufnehmen und erhält so einen 3D-Datensatz (s. auch Konfokalmikroskop).

Mikrosensorarray

Ein Mikrosensorarray integriert mehrere Sensorstrukturen auf einem Sensorchip (multiparametrischer Sensorchip). Durch das parallele Messen mehrerer verschiedener Parameter an ein und der derselben Probe, sowie durch eine ggf. anschließende Korrelation der Parameter, können wesentlich detailliertere Aussagen über die durchgeführten Versuche getroffen werden, als es basierend auf lediglich einem gemessenen Parameter möglich ist.

Mikrotiterplatten

Eine Mikrotiterplatte (auch Multi-Well Platte) ist eine flache Platte mit vielen Vertiefungen (den sog. Wells), die ähnlich einem Reagenzglas als Reagenzräume (Bioreaktor) im Kleinformat verwendet werden. Mikrotiterplatten haben sich im Rahmen von analytischer Forschung vor allem im Bereich der klinischen und pharmakologischen Untersuchungen (Screening) mit hohem Probendurchsatz als Standardwerkzeug etabliert. Auf typischen Platten sind 6, 24, 96, 384 oder 1536 Wells in einer rechteckigen, 2:3 Matrix bei Kantenlängen von 85,47 mm mal 127,76 mm angeordnet.

Multi-Photonen Laser-Scanning-Mikroskopie

Die Arbeitsweise eines Multi-Photonen Laser-Scanning-Mikroskops [87] ist mit der eines konfokalen Mikroskops vergleichbar. Ein Laser rastert die zu untersuchende Probe Punkt für Punkt ab und regt geeignete Zielmoleküle zur Fluoreszenz an. Im Gegensatz zur üblichen Fluoreszenzmikroskopie wird jedoch nicht mit einer Wellenlänge angeregt, die unterhalb der emittierten Wellenlänge liegt, sondern mit deutlich größerer Wellenlänge. Im Fall der Zwei-Photonen-Mikroskopie beträgt bspw. die Anregungswellenlänge in etwa das Doppelte der zu beobachtenden Emissionswellenlänge, bei Drei-Photonen-Anregung ein Dreifaches etc.

Treffen mehrere Photonen gleichzeitig an einem Ort auf, so kann sich ihre Anregungsenergie addieren. Ein entsprechendes Zielmolekül (Fluorochrom) kann durch diese Energiesumme zur Abgabe eines Fluoreszenz-Photons angeregt werden. Die Wahrscheinlichkeit für gleichzeitiges Auftreffen der Photonen ist jedoch nur in einem eng begrenzten Volumen um die fokale Ebene herum ausreichend für eine messbare Fluoreszenz. Ähnlich der konfokalen Mikroskopie kann auf diese Weise eine einzelne Ebene der Probe scharf abgebildet werden. Durch Verschieben dieser Fokusebene ist eine dreidimensionale Bildgebung (Tomografie) möglich.
Als Anregungswellenlänge wird üblicherweise Licht aus dem nahen Infrarot-Spektrum (NIR) verwendet. Solches Licht zeigt eine größere Eindringtiefe in biologische Proben und verursacht weniger Photoschäden (vgl. Phototoxizität). [35, 151]

Nipkow Konfokalmikroskopie

Bei der Nipkow Konfokalmikroskopie wird zur Beleuchtung der Probe im Gegensatz zur (Multi-Photonen-) Laser-Scanning-Mikroskopie i. d. R. kein Laser, sondern Weitfeldbeleuchtung eingesetzt. Eine „Abrasterung“ der Probe wird durch Einführen einer rotierenden Lochmaske, der sog. Nipkow-Scheibe, in den Strahlengang erreicht. Das Lochmuster der Nipkow-Scheibe ist im Strahlengang des Anregungslicht für die punktuelle Beleuchtung der Probe verantwortlich und fungiert gleichzeitig als Lochblende im Strahlengang des fluoreszierenden Lichts (vgl. Konfokalmikroskopie). [42, 113]

optische Pinzette

Eine optische Pinzette ist ein photonisches Gerät zur Manipulation, d. h. zum Festhalten und Bewegen, kleinster Objekte. Eine typische Ausführung spiegelt einen Laserstrahl in ein optisches Mikroskop ein, der dadurch in der Objekt-Ebene fokussiert wird. Die zu manipulierenden Teile müssen bei der verwendeten Wellenlänge durchsichtig sein. Wenn der Laser einmal so eingestellt ist, dass das Objekt im Fokus liegt, führt jede Lageabweichung dazu, dass es durch Impulsübertragung bei der Brechung wieder in den Fokus gezogen wird.
Durch Benutzung eines zweiten Lasers, mit einer Wellenlänge die vom Objekt absorbiert wird, meist Ultraviolett, hat man zusätzlich ein schneidendes Instrument (Laser-Skalpell) zur Verfügung. [6, 5, 142]

Photomultiplier

Ein Photomultiplier ist eine spezielle Elektronenröhre, um schwache Lichtsignale (bis hin zu einzelnen Photonen) zu verstärken und in ein elektrisches Signal umzuwandeln. Ein Photomultiplier besteht aus einer Photokathode und einem nachgeschalteten Sekundärelektronen-Vervielfacher (sog. Dynoden). Auf die Photokathode treffende Photonen lösen Elektronen aus deren Oberfläche. Die freigesetzten Photoelektronen werden in einem elektrischen Feld beschleunigt und treffen auf Dynoden aus deren Oberfläche jedes auftreffende Elektron mehrere Sekundärelektronen herauslöst. Somit nimmt die Anzahl der Elektronen von Dynode zu Dynode kaskadenartig zu, so dass der Verstärkungsfaktor exponentiell mit der Anzahl der Dynoden wächst.

Phototoxizität

Phototoxizität beschreibt die Stärke, mit der ein Stoff unter Einwirkungen von Licht vergiftende (toxische) Wirkungen auslöst. Innerhalb der Fluoreszenzmikroskopie von lebenden Zellen

führt die Phototoxizität aufgrund der Beleuchtung einer fluoreszierenden Probe zum selektiven Zelltod. Dabei hängt die Phototoxizität erheblich von den verwendeten Fluoreszenzfarbstoffen ab. Für die Durchführung von Versuchen an lebenden Zellen sollten allerdings generell sowohl die Intensität als auch die Beleuchtungsdauer minimiert werden, um ein möglichst langes Überleben einerseits und ein nicht Beeinflussen der Proben und damit der Messergebnisse andererseits zu gewährleisten.

Piezo-Aktor

Piezo-Aktoren basieren auf dem piezoelektrischen Effekt der das Zusammenspiel von mechanischem Druck und elektrischer Spannung in Festkörpern beschreibt. Er basiert auf dem Phänomen, dass bei der Verformung bestimmter Materialien auf der Oberfläche elektrische Ladungen auftreten (direkter Piezoeffekt). Umgekehrt verformen sich diese (zumeist Kristalle) bei angelegter elektrischer Spannung. Der sog. inverse Piezoeffekt wird beim Piezo-Aktor ausgenutzt, um durch Anlegen einer Spannung eine mechanische Verformung zu erreichen: Piezopositionierer.

Präprozessor-Switches

Ein Präprozessor ist ein Computerprogramm, das einen Eingabetext konvertiert und das Ergebnis ausgibt. Der Präprozessor der Programmiersprache C führt Änderungen am Programmtext durch, bevor der eigentliche C-Compiler das Programm übersetzt. U. a. werden Makrodefinitionen über die Anweisung „#define“ im Quellcode ersetzt und die zwischen den Anweisungen „#if“, „#ifdef“ und „#endif“ (Präprozessor-Switches) stehen Zeilen entsprechend der Bedingung in den Quellcode eingebunden.
Wird ein Programm übersetzt kann der Entwickler mittels der Präprozessor-Switches (Compile-Zeit-Schalter) über eine oder wenige Optionen Teile des Programmes konfigurieren und für die aktuelle Anwendung anpassen (z. B. mit oder ohne Debug-Programmcode, Änderungen für unterschiedliche Prozessoren u. s. w.).

Proteomik

Das Eiweiß-Inventar einer Zelle nennt man ein Proteom. Die Proteomik versucht, sämtliche Eiweiße im Organismus zu katalogisieren. Die Baupläne der Proteine finden sich in den Erbanlagen. Somit beschäftigt sich die Proteomik bevorzugt mit Ergebnissen sequenzierter Genome. Speichert die Erbsubstanz DNA lediglich Informationen so erfüllen die aus Aminosäuren bestehenden Eiweißmoleküle vielfache Aufgaben. Sie sind Grundsubstanz des Lebens und wehren als Antikörper Krankheiten ab, ermöglichen als Enzyme die Verdauung und sorgen als Muskeln für Bewegung. Im Gegensatz zur stabil bleibenden genetischen Ausstattung verändert sich der Proteinhaushalt eines Körpers ständig. So tragen zwar Raupe, Puppe und Schmetterling dieselben Gene in ihren Zellen, doch unterscheiden sich jeweils die Zusammensetzung und das Zusammenspiel ihrer Proteine wesentlich. Proteine sind also Ursache der Diversität des Lebens. [164, 147, 94, 146]

Quellcode

Unter Quellcode (englisch source code), Quelltext oder Programmcode versteht man in der Informatik den für Menschen lesbaren in einer Programmiersprache geschriebenen Text ei-

nes Computerprogrammes. Abstrakt betrachtet kann man den Quellcode eines Computerprogramms auch als Software-Dokument bezeichnen, welches das Programm so formal exakt und vollständig beschreibt, dass dieses aus ihm vollständig automatisch vom Computer generiert werden kann. Bevor das Programm, das der Programmierer schreibt, von einem Computer ausgeführt werden kann, muss es in Maschinensprache, also in eine vom Computer verständliche Folge von Bits, umgesetzt werden.

Terminalprogramm

Ein Terminalprogramm ist ein spezielles Programm zur Herstellung einer Verbindung mit einem anderen Computersystem oder Pheripheriegerät. Das Programm simuliert gegenüber diesem anderen Gerät einen Terminal. Voraussetzung hierfür ist die Verfügbarkeit einer entsprechenden Hardwareverbindung zwischen den beiden Geräten. Nach der Herstellung der Verbindung kann dann auf dem anderen System wie mit einem lokalen Terminal gearbeitet werden; So lassen sich Daten zwischen den beiden Geräten austauschen. Ein Beispiel für ein Terminalprogramm ist der Microsoft® HyperTerminal, der in jeder Standardinstallation von Microsoft® Windows enthalten ist.

TILLvisION

Die PC-Software TILLvisION wurde ursprünglich zum Betreiben eines Imaging-Systems, bestehend aus einer programmierbaren Lichtquelle (TILL Polychrome) und einer digitalen Kamera (i. d. R. CCD-Kamera), für die Aufnahme von digitalen Bildern und deren Analyse innerhalb der biologischen Lebendzellforschung entwickelt. Später wurde zusätzlichen Funktionalität zum händischen und/oder automatisierten Betreiben eines digital gesteuerten Prozessmikroskops (TILL iMIC) hinzugefügt.

Total Internal Reflection Fluorescence (TIRF)

Total **I**nternal **R**eflection **F**luorescence (TIRF) ist eine Methode der Mikroskopie, um Strukturen zu untersuchen, die sich sehr nahe (ca. 200 nm) an Oberflächen befinden. Man verwendet einen Lichtstrahl, der im Objektträger total reflektiert wird - lediglich eine evaneszente Welle dringt exponentiell abfallend ins Medium ein und regt in diesem sehr schmalen Bereich die Fluoreszenz an. Mit dieser Methode lassen sich z. B. Zellen auf Oberflächen untersuchen, bei denen normalerweise das oberflächennahe Signal vom Hintergrundstreulicht überdeckt werden würde. [8, 9, 150, 13, 71]

Photostimulation

Unter Photostimulation (englisch auch: uncaging) verteht man die künstliche Aktivierung von biologischen Botenstoffen, Zellen oder ganzen Organismen mit Hilfe von Licht. Sie kann benutzt werden um nichtinvasiv die kausalen Zusammenhänge zwischen verschiedenen biologischen Prozessen zu untersuchen. Durch die gezielte Bestrahlung von Teilen einer Probe/eines Organismus können Prozesse lokal aktiviert werden, die entweder in sich abgeschlossen lokal am Bestrahlungsort wirken, oder durch eine Kettenreaktion zu weiteren, globaleren Effekten führen.

XModem

Das XModem ist ein Protokoll, das eine gesicherte Datenübertragung regelt. XModem arbeitet blockorientiert, die zu übertragenden Daten werden in gleich große Einheiten (Blöcke) aufgeteilt. Die Blöcke haben immer eine Größe von 132 Byte und werden ggf. mit beliebigen Zeichen aufgefüllt. Erfolgreich übertragene Blöcke werden positiv bestätigt. Negativ bestätigte Blöcke werden bis zu zehn mal neu gesendet.

B Kommando-Listings zu den Anwendungsbeispielen in Kapitel 7

Die in den folgenden Listings verwendeten Protokollkommandos entsprechen der in Abschnitt 5.6.2 vorgestellten Struktur. Eine kurze Beschreibung der Kommandos ist in Abschnitt 5.9 aufgeführt. Detaillierte Informationen über die einzelnen Kommandos sind in der Dokumentation „ICU v1.0 Firmware Interface Description“ (s. Anhang C) enthalten.

B.1 Zyklisches Umsetzen eines digitalen Ausgangs I

```
0x50  RecordProtocol
0x52  Loop TimeStamp 0 NumberIterations 0
0x66  WriteDigitalOut TimeStamp 40 Bank 4 LineMask 2 ValueMask 1
0x66  WriteDigitalOut TimeStamp 40 Bank 4 LineMask 2 ValueMask 0
0x53  EndLoop TimeStamp 0
0x51  EndRecordProtocol
```

Listing B.1: Protokoll zum zyklischen Umsetzen eines digitalen Ausgangs

Listing B.1 beschreibt das Protokoll zum zyklischen Umsetzen eines digitalen Ausgangs. Die Protokolldefinition wird durch die Kommandos *RecordProtocol* und *EndRecordProtocol* eingerahmt. Durch die Kommandos *Loop* und *EndLoop* wird eine Schleife deklariert, die unendlich oft (spezifiziert durch „*NumberIterations 0*“) durchlaufen wird. Innerhalb der Schleife wird ein digitaler Ausgang mittels des Kommandos *WriteDigitalOut* nach jeweils 40 µs („*TimeStamp 40*“) auf den Wert logisch 1 bzw. logisch 0 gesetzt („*ValueMask 1*“ bzw. „*ValueMask 0*“). Die Ausführung des Protokolls startet mit dem Eingang des Kommandos *RecordProtocol* auf der ICU. Das entsprechende Zeitverlaufsdiagramm des Pegels des digitalen Ausgangs wird in Abbildung 7.6 dargestellt.

B.2 Zyklisches Umsetzen eines digitalen Ausgangs II

```
0x4f  CancelProtocol
0x4e  FreeEvent EventHandle 1
0x4d  ConfigEventDigIn EventHandle 1 EventType 1 EventMode 0
        Bank 2 LineMask 1 CompareMask 1
0x50  RecordProtocol
0x52  Loop TimeStamp 0 NumberIterations 0
0x66  WriteDigitalOut TimeStamp -144 Bank 4 LineMask 2 ValueMask 1
0x66  WriteDigitalOut TimeStamp 110 Bank 4 LineMask 2 ValueMask 0
0x53  EndLoop TimeStamp 0
0x51  EndRecordProtocol
```

Listing B.2: Protokoll zum zyklischen Umsetzen eines digitalen Ausgangs in Abhängigkeit des Signals eines digitalen Eingangs

Listing B.2 erweitert Listing B.1 um die Verwendung eines Events als Ausführbedingung des ersten Kommandos („*TimeStamp -144*“) innerhalb der Schleife. Durch das Kommando *ConfigEventDigIn* wird ein Event des Typs *digitaler Eingang* („*EventType 1*“) als Event Nummer 1 („*EventHandle 1*“) definiert. Das Event ist aktiv, wenn der Wert logisch 1 („*CompareMask 1*“)

am spezifizierten Eingang („*Bank 2 LineMask 1*“) detektiert wird. Ein entsprechendes Zeitverlaufsdiagramm wird in Abbildung 7.7 vorgestellt.

B.3 ICU Initialisierung der PC-Software TILLvisION

```
0x4f  CancelProtocol
0x66  WriteDigitalOut TimeStamp -14 Bank 3 LineMask 4 ValueMask 4
0x66  WriteDigitalOut TimeStamp -14 Bank 4 LineMask 2 ValueMask 0
0x66  WriteDigitalOut TimeStamp -14 Bank 1 LineMask 1 ValueMask 0
0x63  WriteAnalogOut  TimeStamp -14 ChannelNumber 4 VoltageLevel 32767
0x66  WriteDigitalOut TimeStamp -14 Bank 3 LineMask 8 ValueMask 8
0x63  WriteAnalogOut TimeStamp -14 ChannelNumber 1 VoltageLevel 32768
0x63  WriteAnalogOut TimeStamp -14 ChannelNumber 2 VoltageLevel 32768
0x63  WriteAnalogOut TimeStamp -14 ChannelNumber 3 VoltageLevel 29491
0x60  ConfigUART TimeStamp -14 UARTnumber 1
        BaudRate 10 Wordlength 0 StopBit 1 Parity 0
        ReadMode 1 ForceReadParameter 0 ReadTimeout 0
0x60  ConfigUART TimeStamp -14 UARTnumber 2
        BaudRate 10 Wordlength 0 StopBit 1 Parity 0
        ReadMode 1 ForceReadParameter 0 ReadTimeout 0
0x60  ConfigUART TimeStamp -14 UARTnumber 3
        BaudRate 10 Wordlength 0 StopBit 1 Parity 0
        ReadMode 1 ForceReadParameter 0 ReadTimeout 0
0x63  WriteAnalogOut TimeStamp -14 ChannelNumber 3 VoltageLevel 29491
0x4e  FreeEvent EventHandle 2
0x4d  ConfigEventSoft EventHandle 2 EventType 2 EventMode 0
0x02  ReadSystemConfig ConfigurationParameterId 2
0x02  ReadSystemConfig ConfigurationParameterId 5
0x4e  FreeEvent EventHandle 1
0x4d  ConfigEventDigIn EventHandle 1 EventType 1 EventMode 0
        Bank 4 LineMask 1 CompareMask 1
0x4e  FreeEvent EventHandle 4
0x4d  ConfigEventDigIn EventHandle 4 EventType 1 EventMode 0
        Bank 2 LineMask 1 CompareMask 1
0x4e  FreeEvent EventHandle 3
0x4d  ConfigEventWait EventHandle 3 EventType 3 EventMode 0
0x66  WriteDigitalOut TimeStamp -14 Bank 1 LineMask 1 ValueMask 0
0x66  WriteDigitalOut TimeStamp -14 Bank 1 LineMask 2 ValueMask 0
0x66  WriteDigitalOut TimeStamp -14 Bank 1 LineMask 128 ValueMask 0
```

Listing B.3: Kommandosequenz zur Initialisierung der ICU aus der PC-Software TILLvisION

Listing B.3 beschreibt die zur Initialisierung der ICU durch die PC-Software TILLvisION ausgeführten Kommandos. Die Bedeutung der einzelnen Kommandos kann mit Hilfe der vorangegangenen Beispiele abgeleitet werden. Ein als -14 angegebener TimeStamp spezifiziert eine unmittelbare Ausführung des Kommandos ohne protokollgesteuerte, zeitliche Verzögerung.

B.4 Standardprotokoll der PC-Software TILLvisION

```
0x50  RecordProtocol
0x52  Loop TimeStamp 0 NumberIterations 0
0x66  WriteDigitalOut TimeStamp -143 Bank 4 LineMask 2 ValueMask 0
0x66  WriteDigitalOut TimeStamp -144 Bank 4 LineMask 2 ValueMask 2
0x54  If TimeStamp 0 Condition 3
0x82  GetLoopIndex TimeStamp 0 LoopLevel 1
0x59  SendExperimentTimerValue TimeStamp 0
0x56  EndIf TimeStamp 0
0x66  WriteDigitalOut TimeStamp 0 Bank 3 LineMask 4 ValueMask 0
0x66  WriteDigitalOut TimeStamp 20 Bank 4 LineMask 2 ValueMask 0
0x66  WriteDigitalOut TimeStamp 0 Bank 3 LineMask 4 ValueMask 4
0x66  WriteDigitalOut TimeStamp 5080 Bank 3 LineMask 4 ValueMask 0
0x66  WriteDigitalOut TimeStamp 40 Bank 3 LineMask 4 ValueMask 4
0x53  EndLoop TimeStamp 3260
0x81  ExitLoop TimeStamp 0 ExitLoopFlag 0
0x51  EndRecordProtocol
```

Listing B.4: Standardprotokoll der ICU für Bildaufnahmen

Listing B.4 beschreibt das von TILLvisION aufgesetzte Standardprotokoll, das für die Ausführung des Snapshot und des Live-Modus verwendet wird. Das Standardprotokoll verwendet die durch die Initialisierung (s. Listing B.3) konfigurierten Events 1 und 2 („*TimeStamp -144*“ und „*TimeStamp -143*“). Alle als positive ganze Zahlen spezifizierten TimeStamps repräsentieren eine entsprechende Ausführverzögerung auf der Basis von 10 µs. Abschnitt 7.3.1 bietet eine genau Beschreibung der durch das Protokoll beschriebenen Vorgänge.

C ICU v1.0 Firmware Interface Description

Da Teile der PC-Software in Zusammenarbeit mit der Firma Bruxton Corporation, Seattle, USA erstellt werden, ist die Dokumentation der Schnittstelle zur ICU-Hardware in Englisch erstellt worden.

C.1 Introduction:

This document describes the functionality of the **I**ntegration **C**ontrol **U**nit (ICU) and is meant to be a guide and a reference for PC-software development. Additional sections describe general structures, hardware dependent behaviour and update procedures. The documentation evolves along the firmware development and does therefore not claim to cover all existing commands and/or detailed information. Do not hesitate to ask for further details if anything should be unclear. Please visit http://www.lrz-muenchen.de/~Geisler/ for contact information. The current versions of the ICU-firmware, boot-loader, CPLD JEDEC-files and this documentation are available at this webpage as well. Additionally, the website features an error-report portal that should be used to report errors, suggest changes or additional features and for requests of further information.

Several times within this document, the author refers to the appendix. The appendix is part of an automatically generated html website and covers the documentation of firmware internal constants such as status constants, module numbers, system configuration constants etc. Along with this document should be a file named ”appendix.html” and a directory named ”appendix”. Start ”appendix.html” in a web-browser to view the additional information provided by the appendix. The appendix can be obtained along with this documentation from the website that has been mentioned above.

C.2 System description

C.2.1 Boot Sequence

The ICU’s operation is controlled by a DSP. Like any processor, the DSP boots and initialises all internal devices upon turning on the ICU’s power. This process takes a short period of time (several seconds), in which the host PC cannot communicate with the ICU. During the boot process a specific set off LEDs signal each boot state. (Note: The LEDs are on the DSP board also known as the Smart Move digital board) The knowledge about the different boot states is of no importance to the user and is therefore not explained in further detail. However, one can determine the end of the boot process by the final LED set: After a ”Night Rider” like blink sequence, one green LED on the very end of the line of LEDs is turned on. All other LEDs are turned off.

FYI: In normal operation, during protocol execution the green LED on the other end of the line of LEDs is turned on. If a severe error occurs, all green LEDs are turned off, and the red LED is turned on. The ICU must be hardware-reset at this point.

C.2.2 Host PC - ICU Interface

The ICU's interface to the host PC is a standard EIA-232 interface. A standard EIA-232 cable (no null-modem cable) must be used. Upon start-up this EIA-232 interfaces is configured with the following settings:

- baud rate: 57600 bits/s
- word length: 8 bit
- bits: 1
- parity: none
- hardware control: non

Using the "CmdConfigRS232" command, the EIA-232 interface to the ICU can be reconfigured. The settings that were set last are valid until the ICU is powered off.

Communication between the host PC and the ICU (both directions) follow the conventions described in a later paragraph on the "Command Structure". A data stream sent to the ICU is immediately decoded and scanned for valid command sequences. Once a valid command sequence has been detected, the corresponding command is immediately handled. In general, every command responds with an 'acknowledge'. If the command implies to return any data, the data is included in the response.

Any errors that occur during the decoding process, command and/or protocol execution as well as system errors (e.g. buffer overflow) are immediately reported to the host PC by a spontaneous response. Please refer to the section on "ICU Error Responses" for more information on possible errors.

C.2.3 Command Structure

The ICU uses the general command structure, which has been developed for the interface of TILL's Polychrome V. This structure has been described in detail in Bruxton's document "TILL Polychrome V Internals" and is presented at the end of this section.

The firmware "command decoder" source code can be obtained from the firmware developer as a reference for the command decoding procedure as well as the parameter ordering/alignment. Commands that are send from a host PC to the ICU as well as responses that are send from the ICU to a host PC follow the conventions of the general command structure.

The following is an excerpt from Bruxton's documentation:

——— start of quote from Bruxton's documentation ———

Communications Protocol

This section contains a description of the communications protocol between the host computer and the Polychrome V. [...]

Transport Protocol

RS-232

The Polychrome V supports RS-232 serial communications using 8 bit characters, no parity, one stop bit. This combination of settings is sometimes abbreviated as 8N1.

The Polychrome V supports a standard set of baud rates. The unit defaults to 19200 baud. During operation, the most common other rate used is 57600 baud. A host computer searching for a Polychrome V unit on a serial port should attempt to communicate at both 19200 and 57600 baud.

The RS-232 communications standard does not provide any communications error checking beyond signal timing and optional parity. The Polychrome V does not make use of parity, instead, it performs communications error checking at the transport level. Each command from the host computer to the Polychrome V, and each response from the Polychrome V to the host computer, has the following sequence of 8-bit characters:

Offset	Content
0	Command code
1	Command/response length (N) (true)
2	Command/response length (N) (complement)
3..N	Parameters
N+1	CCITT CRC-16 (low order byte)
N+2	CCITT CRC-16 (high order byte)

The command code is the same for both a command and the associated response. That is, the response to a command with command code X is a response with the same command code X.

The command or response length N is the total length of the command or response excluding the trailing CRC value. The command or response length is represented twice: once at offset 1 as the true value, and once at offset 2 as the complement of the true value. For example, if a command is 20 bytes long, the byte at offset 1 will contain the value 20, and the byte at offset 2 will contain the value 235. The total length of the command, including the 16-bit CRC value, will be 22 bytes. Many commands and responses have no parameters. In that case the command or response length N is the value 3

The 16-bit CRC value is calculated using code equivalent to the following C++ fragment. Note that the fragment assumes that an 'int' is wider that 16 bits to simplify the code. For example, this would be the case on a 32-bit computer:

```
// CRC calculation based on code supplied by Thomas Geisler. This code was
// originally based on code from Peter Boswell, provided as part of the
// documentation of xmodem.
//
// The algorithm calculates the standard CCITT CRC-16, based on the binary
// polynomial x^16 + x^12 + x^5 + x^1, where '^' is understood to mean 'to
// the power of'. Many CRC-16 implementations, such as that in Numerical
// Recipies by Press, et al, pre-calculate a table to avoid shifting data
// in the inner loop. Modern processors implement fast shifts, so this
// algorithm is efficient, without the use of lookup tables.
//
// The standard implementation based on the xmodem code is written for a
// 16-bit processor, so it must check bit 15 of 'CRC_value' before shifting
```

```
// the value left, since bit 15 will be lost in the shift. That is, the
// 'if' statement must check 'CRC_value & 0x8000', then shift and
// exclusive-or with 0x1021 if bit 15 was set prior to the shift. This
// algorithm is implemented for a 32-bit or greater processor, so
// 'CRC_value' is first shifted left, then bit 16 of the shifted value is
// checked.
//
// The 'data' array may be treated as either signed or unsigned 'char'
// values. When a 'data' value is read, it is shifted left by 8 bits, so
// any sign extension is moved into the high-order 16 bits. The resulting
// value is exclusive-ORed with 'CRC_value', so only the high-order 16-bits
// of 'CRC_value' are affected by the sign extension, and these bits are
// discarded. If a 32-bit CRC value were being calculated, 'data' would
// have to be treated as an unsigned value.

char CRC(const char* data, int length)
{
        int CRC_value = 0;
        for (int index = 0; index < length; ++index)
        {
                CRC_value ^= (int) data[index] << 8;
                for (int bit = 8; bit > 0; --bit)
                {
                        CRC_value <<= 1;
                        if ((CRC_value & 0x10000) != 0)
                                CRC_value ^= 0x1021;
                }
                // End of processing character
        }
        // End of processing string
        return CRC_value & 0xffff;
}
```

Listing C.1: CRC function (from Bruxton SDK)

——— end of quote from Bruxton's documentation ———

C.2.4 ICU Timing

The ICU's main feature is its ability to control external devices with a predefined timing within a protocol. Examples of external devices are: the iMIC with the objective changer, the filter slider, the xy-table and the z-drive, the Polychrome with triggered wavelength control, a CCD-camera with a synchronization line and any third party or customer devices that connect to the provided interfaces. Commands that can be added to a protocol have at least one parameter: a four byte (32bit) integer time stamp. The time stamp specifies, when the command is to be executed with a resolution of 10 µs. The protocol timer is reset upon the start of a recorded protocol. The timing of the first command within a protocol is relative 0. All further timing values are relative to the execution time of the previous command.

The protocol timer is also reset at specified trigger or gate events, resulting in a timing of the following commands relative to the event.

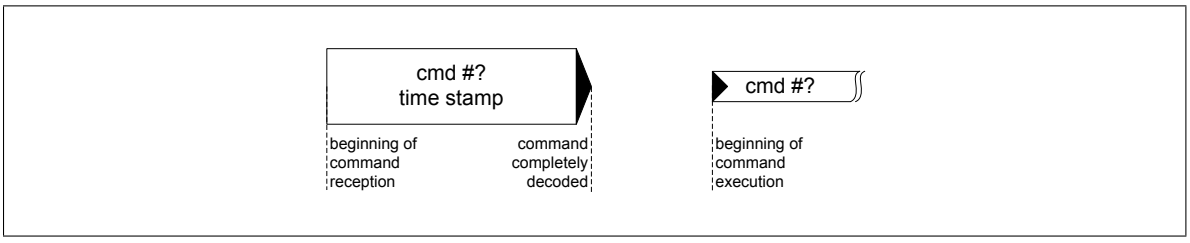
Additionally, the protocol timer is reset at the beginning of a loop block to guarantee exact timing for each loop cycle.

Several predefined constants for the time stamp parameter indicate special timing behaviour:

- reserved: 0xFFFFFFF0
- reserved: 0xFFFFFFF1
- immediately* 0xFFFFFFF2: issuing commands with the time stamp set to the constant "immediately" will result in immediate execution of the command even if a protocol is currently executed. A currently running protocol is thus bypassed!
  Note: Mind that bypassing a currently executed protocol might lead to completely different results that may damage the hardware!
- wait for gate** 0xFFFFFFFE
- wait for trigger** 0xFFFFFFFF

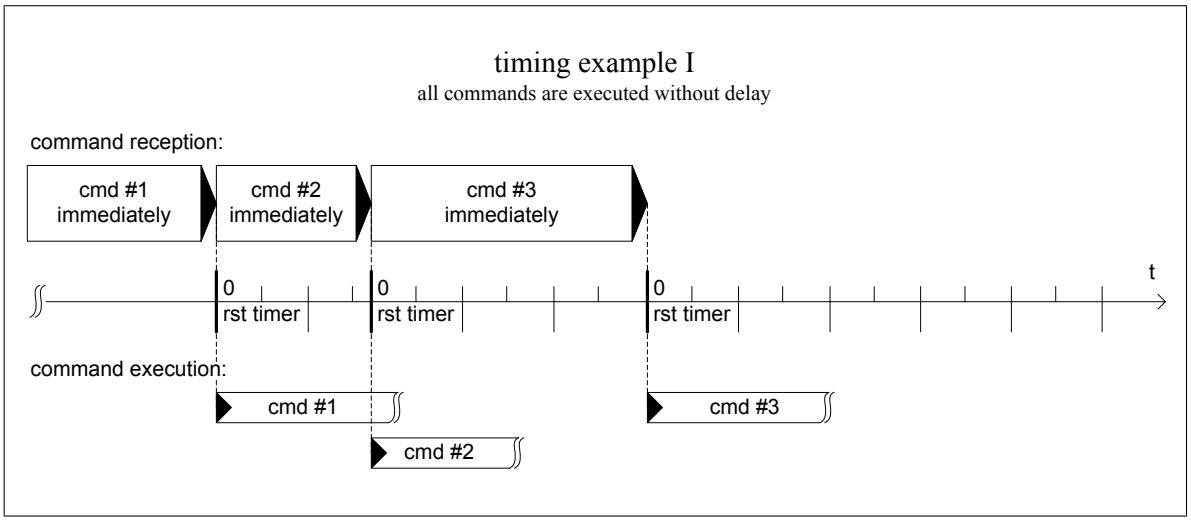
* the difference between setting the time stamp to zero or to the value of the immediate constant will be explained below.
** not supported so far

Commands that are issued outside the protocol recoding mode (see "CmdRecordProtocol") are executed right away according to the time specified by the individual time stamp parameters. Therefore, after completely receiving a command, the execution of the command is delayed by the time specified by the time stamp parameter. If a second command is completely received during this delay time, then the beginning of its execution will be delayed relative to the execution time of the first command. This behaviour applies to all commands, which are completely received and decoded while the execution of previous commands is delayed, resulting in a protocol like behaviour. The following timing examples explain the timing behaviour outside a protocol:

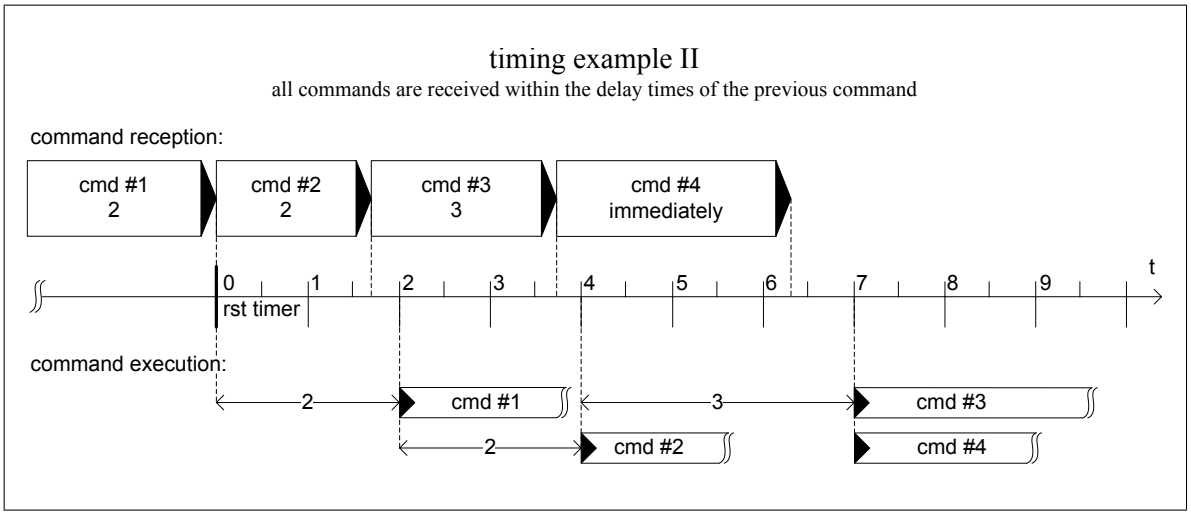


When receiving a command, only the moment when the command is completely received and decoded is of importance for the timing behaviour. While recording a protocol, the command is added to the protocol list at this point. Outside protocol recording mode, the timing behaviour is determined at this point.

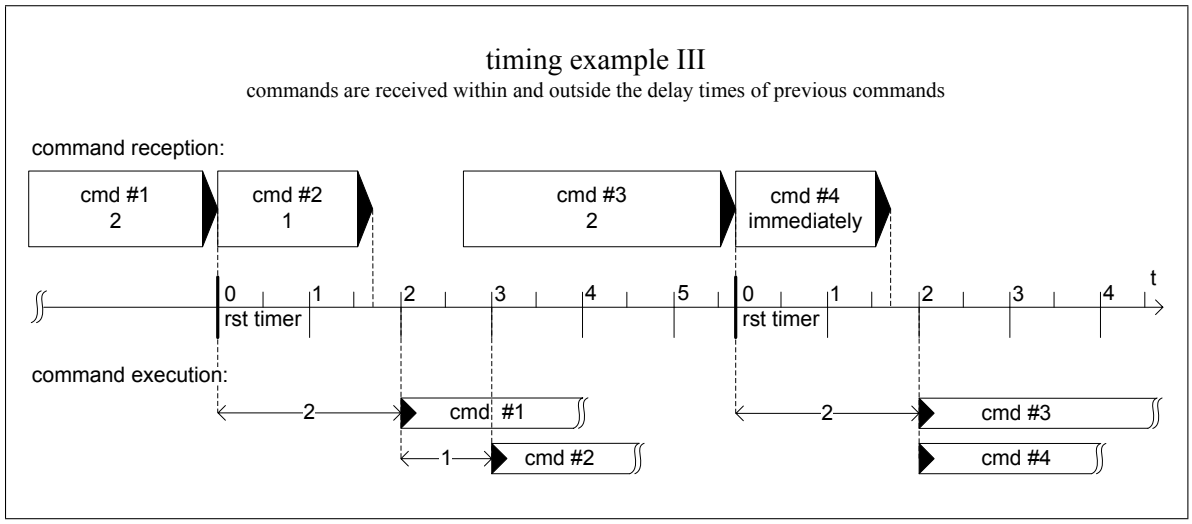
When a command is executed (within a protocol or outside with a given delay) only the beginning of execution is of importance for the timing behaviour. In theory the execution of commands is only initiated at specified moments and the command's functionality is performed in the background from then on. The duration of each command is of no importance for the ICU's internal timing behaviour. However, due to the complexity of controlled systems (like the iMIC) restrictions may apply. Since those restrictions are unknown on the firmware level, they need to be considered by PC-software developers.



Timing Example I: The time stamps of all commands indicate immediate execution. Therefore, execution is not delayed and the moment, at which a following command is completely received and decoded, is not during a delay time. Thus, each command is executed as soon as it is received and decoded with no timing relation to previous commands.



Timing Example II: All commands (except the first one) are immediately decoded when the system delays a previously received command. Thus, the given time stamps are set in relation to the execution time of the previous command.



Timing Example III: Command #2 is received while the execution of command #1 is delayed. Execution of command #2 is therefore delayed relative to the beginning of execution of command #1. The reception of command #3 is finished after the execution of command #1 and #2 was initiated. Mind that the beginning of reception is of no importance! Upon completely receiving and decoding command #3, the timer is therefore reset. Command #4 is completely received and decoded, while the execution of command #3 is delayed. Thus, the execution of command #4 is relative to the execution of command #3.

C.2.5 ICU Error Responses (0xE1)

Errors that occur upon the ICU's boot sequence, during communication between the host-PC and the ICU, or within protocol execution must be handled to guarantee proper functionality. However, not all errors can be handled on the firmware level and therefore must be passed to the host-PC for further error handling action. Error messages follow the conventions of the general command structure (refer to the section on "Command Structure"). The command byte is set to 0xE1. Error messages have the following sequence of parameters:

Length	Type	Parameter
1	int	error number
1	int	module number
3	int	line number
1	int	error level
1	int	emergency flag
1	int	exit flag

Detailed lists of "error numbers" and "module numbers" can be found in the appendix. For the error numbers, please refer to the appendix's section "Module Status Enumerations", since the error numbers correspond to the status values. For possible "module number", please refer to appendix section "Module Number Enumerations".

C.2.6 ICU Production

This section features several, valuable information on ICU production, updating procedures and hardware related firmware issues.

C.2.6.1 Digital Board Preparation

The Smart Move digital boards need to be prepared for the use in an ICU when they get shipped by Smart Move. The following procedure has to be performed in the correct order.

1. check the digital board and/or the ICU main board (v1.2) for the following hardware properties:

   The Zener-diodes on the ICU main board must be removed (D9, D10,D11, D12, D13, D14, D15) and the resistor net R6 (8 x 100ohm) must be replaced by a resistor net with 8 x 330ohm (referred to the schematic "mainb_with_flex_trg")

   OR

   SCI Pins on SV4 of the SmartMove digital board are pushed through the board to the top side of the PCB (5 Pins: SV4.8, SV4.10, SV4.12, SV4.16, SV4.18) (referred to the schematic "AR9632_5")

   The preferred method is method b

2. update the boot loader.Use the file "bootld.dli".

3. program the ICU specific JED files to the ICU CPLDs

4. update the ICU specific firmware Use the file "image.dli".

Known problems: the digital board might not boot properly after the described procedure. This might be due to a minimal capacitance change at the chip select line of the flash memory chip that is used to store the boot loader. Soldering a 10nF capacitor between the chip select pin and GND at flash number one will resolve the problem.

C.2.6.2 Firmware and Boot-Loader Update

The standard EIA-232 interface, which connects the ICU to the host PC, is used for uploading a new firmware or a new boot-loader image to the ICU. This description of the update procedure applies to firmware images as well as to boot loader images. Files that are to be updated must be of a certain format and must be provided with a specific file header. The typical file extension is *.dli (DownLoadable Image). Any standard terminal software, which supports the Xmodem protocol, is suitable to perform an update. On PCs that run Microsoft Windows©, the Windows Hyperterminal© can be used. To upload a new image (file ending *.dli) follow these steps:

1. physically connect the ICU to the host PC (regular EIA-232 cable; NO null modem!)

2. verify that the ICU's power is turned off

3. start a PC terminal software (e.g. Windows Hyperterminal©) and open a direct connection via the appropriate COM port (see 1.). Use the following settings:

   - baudrate: 115200
   - data bits: 8
   - parity: none

- stop bits: 1
- flow control: none

4. if the image file is on a floppy disk, copy the file to the PCs hard drive
5. Use Transfer - Send File (or similar) to send the downloadable image file (*.dli).
6. Select Xmodem or 1k Xmodem (faster) as the protocol. Select the downloadable image file on your hard disk and start sending.
7. Turn the ICU's power on.
8. The image file transfer starts after several seconds. Messages may occur during the update process. After a successful firmware update, the ICU responds with the message: "Transfer OK"; If errors occur, please check your terminal program configuration, re-power the ICU and try to send the image file again by repeating steps 5 to 8.
9. after a successful update, close the terminal program

C.2.6.3 CPLD Update

Use the Lattice ispLEVER-Starter software package to program the ICU's CPLDs. The software can be downloaded from http://www.latticesemi.com/ and usually comes with a 6 months licence for free.

The license can be renewed after expiring by simply downloading and installing the software again.

To program the CPLDs start "ispVM System". With the programming cable being connected to the target board (ICU main board SV4) and the ICU being turned on, click the "Scan" button. The software should recognized two devices in the following order:

- iM4A5-192/96
- iM4A3-128/64

By double-clicking the first device a dialog box opens. Select the data file for the main board's CPLD (io0815.jed) and choose "Erase, Program, Verify, Secure" as the operations that are to be completed.

By double-clicking the second device a dialog box opens. Select the data file for the Smart Move digital board's CPLD (ar9xxx.jed) and choose "Erase, Program, Verify, Secure" as the operations that are to be completed.

Finally click on "GO", which will start the programming process.

C.2.7 Digital Board LEDs

Onboard the SmartMove Digital Board (DSP-Board) are five LEDs that have already been mentioned in the section on the boot sequence. After the boot sequence and during normal operation of the ICU these LEDs are used to indicate the following internal states:

- LED 1 (green): Turned on, when the system is initialized and ready for use.
- LED 2 (green): Reserved
- LED 3 (red): Turned on, when a severe error occurred. An error message is send to the host PC and must be handled appropriately. Depending on the error, the ICU might not function properly and must be hardware-reset (see section ”ICU Error Responses”)
- LED 4 (green): Turned on, when a firmware internal task is pending.
- LED 5 (green): Turned on, when a protocol is executed.

C.2.8 ICU Hardware Configuration

Each **P**rinted **C**ircuit **B**oard (PCB) within the ICU with own functionality is equipped with a 2k-bit (256 byte) flash memory chip. The PCBs are distinguished by their major functionality and are given a fixed identification numbers, such as 0xA2 for the master main board. A detailed list of all identification numbers can be found in the appendix in the list entitled ”PCB Flash Chip IDs”.

The flash chips can be written and/or read by using the commands ”CmdWriteSystemConfig” and ”CmdReadSystemConfig” with the appropriate parameters as described in the corresponding sections. Besides the number of bytes that are stored to a flash chip, a list of parameters describes the chips that the PCB is equipped with and thus the interfaces that can be used during operation of the ICU. Each list entry consists of two bytes: The first byte describes the device class (e.g. EIA-232, ADC . . .) and the second byte describes the device type (e.g. MAX3110E, DAC7634E . . .). Detailed lists of ”Device Classes” and ”Device Types” can be found in the appendix.

C.3 System Commands:

C.3.1 CmdNoOp (0x00)

Issuing a ”CmdNoOp” command literally does nothing. However, it can be used as a dummy command within protocols to introduce an additional delay for compensation of conditioned execution of an if-else-endif block. Further on, it can be used to introduce delays that are greater than the maximum possible time that can be specified by the 32 bit time stamp, by adding the ”CmdNoOp” command with appropriate delays to the protocol. For extremely long delays the command might be placed within a loop.

The command immediately responds with no additional parameters (acknowledge). Execution starts at the time, specified by the parameter ”time stamp”. The command has the following sequence of parameters:

Length	Type	Parameter
4	int	time-stamp

C.3.2 CmdWriteSystemConfig (0x01)

Using the ”CmdWriteSystemConfig” command, individual system configuration parameters can be set or permanently written to the ICU. The command immediately responds with no

parameters (acknowledge). Execution starts immediately. If not specifically defined elsewhere the command has the following sequence of parameters:

Length	Type	Parameter
1	int	configuration parameter ID (see list below)
x		configuration parameter

Possible configuration parameter IDs:

- 0 = Device Serial Number (type: 3-byte integer)
- 1 = Device Hardware Configuration (type: array of integers)
- 2 = reserved*
- 3 = reserved*
- 4 = System Use Indicator (type: 3-byte integer): The value of the system use indicator is not changed by the ICU at any time.
- 5 = reserved*
- 6 = reserved*
- 7 = Command Decoder Inter-Byte-Timeout-Switch** (type: 1-byte integer): 1 = use timeout; 0 = do NOT use timeout, upon system re-boot this parameter is set to 1. If this switch is enabled, the command decoder starts a time out whenever a new byte was received. If the timeout expires, the command decoder's state machine is reset. This feature may be used to synchronize the communication between a host PC and the ICU upon the first connection after a system restart.
- 8 = CPLD Version (type: array of characters/integers): up to 20 characters (ASCII-text) can be permanently saved to the ICU's flash that specify the CPLD version as it is tagged in the revision control system
- 9 = Send Start-up Warning (type: 1-byte integer): Set this flag to have a warning send to the host-PC when ever a command is correctly decoded and the system use indicator (see parameter ID 4) is different from zero (0). Use this mechanism to have the ICU notify the PC-software that is was reset.

* reserved IDs are for reasons of consistency with "CmdReadSystemConfig"
** volatile: value is only valid until the next re-boot

Note: Calling this function to permanently write parameters into the ICU flash memory (such as the serial number) may take several seconds. The EIA-232 interface from the PC to the ICU is disabled during the write process!

C.3.2.1 WriteHardwareConfig (0x01-01)

When using the "WriteHardwareConfig" command with the configuration parameter ID set to one (1), the command expects additional parameters as explained below. The command immediately responds with no parameters (acknowledge). An error is returned instead of an acknowledgement if the specified device does not exist. Execution starts immediately. The command has the following sequence of parameters:

Length	Type	Parameter
1	int	configuration parameter ID: one (1)
1	int	I^2C device address*
1	int	number of configuration bytes
x	int	configuration parameters

* refer to appendix ”Hardware Configuration - t_CuMain_PCBAddresses” for possible I^2C device addresses

The configuration parameters are organized in groups of two bytes, where the first byte corresponds to the ”device class” and the second parameter corresponds to the ”device type”. Constants for both parameters can also be found in the appendix to this documentation: t_CuMain_HardwareDeviceClass and t_CuMain_HardwareDeviceType.

C.3.3 CmdReadSystemConfig (0x02)

Using the ”CmdReadSystemConfig” command, individual system configuration parameters can be read from the ICU. The command immediately responds with the configuration parameter ID and the requested configuration parameter or with no parameters (acknowledge) if the specified configuration parameter ID does not exist. Execution starts immediately. If not specifically defined elsewhere the command has the following sequence of parameters:

Length	Type	Parameter
1	int	configuration parameter ID (see list below)

Possible configuration parameter IDs:

- 0 = Device Serial Number (return-type: 3-byte integer)
- 1 = Device Hardware Configuration (return-type: array of integers)
- 2 = Firmware Version (return-type: ASCII-text): The version returned corresponds to the tag within the firmware revision control system. Firmware versions set to ”none” specify unofficial testing version that are neither checked in nor tagged within the revision control system
- 3 = Command Set ID (return-type: 3-byte integer)
- 4 = System Use Indicator (return-type: 3-byte integer): Can be used to check whether the device has been used after a boot process: returns zero, when read the first time after a system boot process. The value of the system use indicator is not changed by the ICU at any time, but it can be changed by using the command ”CmdWriteSystemConfig”.
- 5 = Firmware Compile Date (return-type: ASCII-text): The returned text is of the following format: ”yyyy-mm-dd - hh.mm”. If both, the firmware version and the operating system version, are defined (!= ”none”), the compile date request will return the ASCII-text ”none”.
- 6 = Operating System (SEROS) Version (return-type: ASCII-text): the returned SEROS version corresponds to the tag within the firmware revision control system. SEROS versions set to ”none” specify unofficial testing version that are neither checked in nor tagged within the revision control system

- 7 = Command Decoder Inter-Byte-Timeout-Switch (return-type: 1-byte integer): 1 = use timeout; 0 = do NOT use timeout, upon system re-boot this parameter is set to 1
- 8 = CPLD Version (return-type: ASCII-text): up to 20 characters specify the CPLD version as it is tagged in the revision control system.
- 9 = Send Start-up Warning (type: 1-byte integer): if this flag is set a warning is send to the host-PC when ever a command is correctly decoded and the system use indicator (see parameter ID 4) is different from zero (0); use this mechanism to have the ICU notify the PC-software that is was reset.

Note: Calling this function and reading parameters from the ICU flash memory (such as the serial number) may take several seconds. The EIA-232 interface from the PC to the ICU is disabled during the read process!

The command's response has the following sequence of parameters:

Length	Type	Parameter
1	int	configuration parameter ID (see list below)
x		requested configuration parameter

C.3.3.1 ReadHardwareConfig (0x02-01)

When using the "ReadHardwareConfig" command with the configuration parameter ID set to one (1), the command expects an additional parameter as explained below. The command immediately responds with the configuration parameter ID, the I^2C device address, the number of configuration bytes and all configuration parameters that could be read from the specified device. The number of configuration bytes equal zero (=0) and no configuration parameters are returned, if the specified device does not exist. Execution starts immediately. The command has the following sequence of parameters:

Length	Type	Parameter
1	int	configuration parameter ID set to one (1)
1	int	I^2C device address*

* refer to appendix "Hardware Configuration - t_CuMain_PCBAddresses" for possible I^2C device addresses

The command's response has the following sequence of parameters:

Length	Type	Parameter
1	int	configuration parameter ID: one (1)
1	int	I^2C device address*
1	int	number of configuration bytes
x	int	requested configuration parameter

* refer to appendix ”Hardware Configuration - t_CuMain_PCBAddresses” for possible I^2C device addresses

The configuration parameters are organized in groups of two bytes, where the first byte corresponds to the ”device class” and the second parameter corresponds to the ”device type”. Constants for both parameters can be found in the appendix to this documentation: t_CuMain_HardwareDeviceClass and t_CuMain_HardwareDeviceType.

C.3.4 CmdHardwareReset (0x07)

Using the ”CmdHardwareReset” command, the ICU can be hardware-reset. The command immediately responds with no additional parameters. The hardware-reset is performed as soon as the response is completely transmitted. The command has no parameters.

C.4 Events:

The ICU currently provides 128 internal events, which can be configured individually. The events may be used as conditions for if- and while-statements or instead of the time-stamps given to specific instructions within a protocol.

Events that are used as time stamps delay the execution until the specified condition is satisfied und may therefore be used to trigger or synchronize protocol sections. To choose an event as the condition to execute a command simply use any of the respective constants defined in the appendix in the list entitled ”Event Time Stamp Constants”. The event must be configured adequately before it is used instead of a time stamp.

Events can be either hardware- or software-based, e.g. based on the state of a digital input or a received byte at an EIA-232 interface.

C.4.1 CmdConfigEvent (0x4D)

Using the ”CmdConfigEvent” command, one of the ICU internal events can be configured. The command immediately responds with no additional parameters. The command has the following sequence of parameters:

Length	Type	Parameter
1	int	event handle*
1	int	event type
1	int	event mode**
x	int	event parameters

* chose a non used handle (currently 1-128)
** not specified yet, set to 0 for future compatibility

Depending on the chosen event type different sets of additional parameters must be provided.

- 1 = DIn (total of three additional parameters: 3 bytes): A DIn-event occurs, when the specified digital input (bank number and line mask) has the specified value (compare mask)

    - 1-byte integer: digital in bank number
    - 1-byte integer: digital in line mask
    - 1-byte integer: digital in compare mask
- 2 = Soft (no additional parameters): A soft-event occurs, when the command ”CmdSoftEvent” (0xA5) was send to the ICU and decoded correctly.
- 3 = Wait (no additional parameters): A wait-event occurs, in case a previously configured event that was used as a time stamp caused a delay of the protocol execution. Using the wait-event as a condition within an if-statement will automatically reset the event status.
- 4 = Jumper
    - 1-byte integer: number of jumper (1..5): A jumper-event occurs, when the specified jumber is set.
- 5 = Expression (an arbitrary number of additional parameters: x bytes):
    - x-byte integer (char): mathematical expression according to the specifications given in section ”Mathematics”.
- 6 = EIA-232 (total of one additional parameter: 1 bytes): An EIA-232-event occurs, when a response from a device was detected according to the specified read mode. Currently only the mode ”automatic end-byte” is supported.
    - 1-byte integer: UART number (1, 2, 3, ..)
- 7 = MUC (no additional parameters): A MUC-event occurs, when the trigger bit within the MUC status register is set. Mind that the MUC has to be configured adequately in order to enable trigger functionality.

Note: The attempt to configure an event that is already configured will result in an error. Please use the ”CmdFreeEvent” command prior to a new configuration if you are sure that the previously configured event is not used any more.

C.4.2 CmdFreeEvent (0x4E)

Using the ”CmdFreeEvent” command, a previously configured event can be freed and is thus available for new configuration again. The command immediately responds with no additional parameters. The command has the following sequence of parameters:

Length	Type	Parameter
1	int	event handle

Note: The ICU firmware does not check whether the event is used by a previously configured or running protocol at the time the ”CmdFreeEvent” command is executed!

C.4.3 CmdSoftEvent (0x5A)

Using the ”CmdSoftEvent” command, an event that has been previously configured as a software event is activated according to the specified mode. The command immediately responds with no additional parameters. The command has the following sequence of parameters:

Length	Type	Parameter
1	int	event handle

C.5 Protocol:

C.5.1 CmdRecordProtocol (0x50)

Issuing a ”CmdRecordProtocol” command switches the ICU into the protocol recording mode. The command immediately responds with no additional parameters. Execution starts immediately. The command has no parameters.

All further commands, sent to the ICU, are added to the protocol rather than being executed right away or according to their time stamp. Each command that gets added to a protocol this way responds immediately with no parameters (acknowledge), even if the response is specified differently within in a later section. Commands whose responses include any kind of data (e.g. ”CmdReadUART”;) first respond with an ”acknowledge” and will then respond a second time at the given execution time during protocol execution. Commands, which naturally respond with no parameters, will not send an additional response during protocol execution.

Recording a new protocol while a previously set up protocol is being executed is possible. Protocol execution will stop at the first command of the actually recorded protocol, if the previously set up protocol is completely executed and the actual protocol recordeding has not been finihed yet (no ”‘CmdEndRecordProtocol”’ has been send).

C.5.2 CmdEndRecordProtocol (0x51)

Issuing a ”CmdEndRecordProtocol” command switches the ICU out of the protocol recording mode. The command immediately responds with no parameters (acknowledge). Execution starts immediately resulting in starting protocol execution. The command has no parameters.

C.5.3 CmdCancelProtocol (0x4F)

Issuing a ”CmdCancelProtocol” command discards all remaining commands and completely resets the protocol. The command immediately responds with no additional parameters (acknowledge). Execution starts immediately. The command has no parameters.

C.5.4 CmdLoop (0x52)

Issuing a ”CmdLoop” command adds the beginning of a loop with the given number of iterations to the protocol. The command immediately responds with no additional parameters (acknowledge). Execution starts at the time, specified by the parameter ”time stamp”. The command has the following sequence of parameters:

Length	Type	Parameter
4	int	time-stamp
2	int	number of iterations

Issuing a ”CmdLoop” command outside protocol recording also results in the beginning of a loop. However, further commands are executed right away according to the time specified by the individual time stamp parameters.

For an infinite loop set the number of iterations equal to zero. An infinite loop cannot be stopped except by issuing the ”CmdCancelProtocol” command.

Nesting loops is possible. The current nesting depth equals eight.

C.5.5 CmdEndLoop (0x53)

Issuing a ”CmdEndLoop” command adds the end of a loop to the protocol. The command immediately responds with no additional parameters. Execution starts at the time, specified by the parameter ”time stamp”. The command has the following sequence of parameters:

Length	Type	Parameter
4	int	time-stamp

When a ”CmdEndLoop” command is issued, the complete loop will be executed the given number of iterations - 1 times.

C.5.6 CmdIf (0x54)

Issuing a ”CmdIf” command adds the beginning of an if-statement to the protocol. The command immediately responds with no additional parameters (acknowledge). Execution starts at the time, specified by the parameter ”time stamp”. The command has the following sequence of parameters:

Length	Type	Parameter
4	int	time-stamp
1	int	condition/event-handle

Nesting if-statements is possible. The current nesting depth equals eight.

The one byte long condition/event handle specifies the condition or the event that determines whether the if-statements are to be executed or not. The specified condition or event must be configured prior to protocol execution! For further details on configuring the condition please read the section on ”Events”.

C.5.7 CmdElse (0x55)

Issuing a ”CmdElse” command adds an else statement to the protocol. Each else-command refers to the last issued if-command. The command immediately responds with no additional parameters (acknowledge). Execution starts at the time, specified by the parameter ”time stamp”. The command has the following sequence of parameters:

Length	Type	Parameter
4	int	time-stamp

C.5.8 CmdEndIf (0x56)

Issuing a ”CmdEndIf” command adds the end of an if-statement to the protocol. Each end-if-command refers to the last issued if-command or else-command respectively. The command immediately responds with no additional parameters (acknowledge). Execution starts at the time, specified by the parameter ”time stamp”. The command has the following sequence of parameters:

Length	Type	Parameter
4	int	time-stamp

C.5.9 CmdSendResponse (0x57)

Using the ”CmdSendResponse” command, custom defined responses can be added to the protocol. The command immediately responds with no additional parameters (acknowledge). Execution starts at the time, specified by the parameter ”time stamp” and results in the ICU sending the specified response. The command has the following sequence of parameters:

Length	Type	Parameter
4	int	time-stamp
x		arbitrary response parameters

When getting sent back to the host PC, the response parameters are embedded in the regular command structure with the command byte equal to 0x57.

C.6 Mathematics:

The ICU supports the internal processing of mathematic expressions. For now these expressions are described by a literal text string of characters. All numbers, constants and variables are integer only. The form of the expressions is of the same type as one would write a mathematical expression or equation. White characters such as a space are allowed: ”1+2 * 3” will return 7.

Mathematical expressions can be used to define conditions (e.g. an if-condition). Refer to the section on the configuration of events for information on how to define conditions using mathematical expressions.

Variables can be used within such expressions. However, they must be declared prior to their use (refer to command ”CmdDeclareVariable”). Unused variables will stay in the ICU memory until they are released by issuing the command ”CmdFreeVariable”. Variables are addressed within expressions by the character ’v’ (lower case) with an adjacent number that specifies the variable (e.g. ”v5” specifies the variable that was previously declared setting the variable index to the integer value 5). Examples: ”1+2*v5” will return the result of the calculation 1+2*v5, where ”v5” is the value of the variable. ”V1+V2*V3” will return the result of the according multiplication and addition.

Special variables exist within the ICU. For now, only the counters that reflect the number of runs a loop of a certain level has been executed within a stack of nested loops, are available for

the use within expression. These variables are addressed within expressions by the character 'l' (lower case) with an adjacent number that specifies the level of the loop, whose counter is to be specified (e.g. "l5" specifies the counter of the loop at the fifth loop level). Loop counter variables are 1-based. Special variables do not need to be declared.

Besides basic calculations, logical operations can be performed by the math module. Currently three operations (<; >; ==) are available. Mind that the comparison "equal" must have the two '=' immediately one after the other without white characters such as a space. The result of such a logical expression is either 0 = false or 1 = true. "1 > 2" will return 0; "2 < 1+2*3" will return 1; "3 ==v1 *2" will return 0 in any case, since "v1" is an integer and thus the product of the variable and 2 cannot equal 3 in any case.

Note: the maximum number of variables that can be defined by the user is currently 20.

C.6.1 CmdDeclareVariable (0x5B)

Using the "CmdDeclareVariable" command, an ICU internal variable can be declared for the use within mathematical expressions. The command immediately responds with no additional parameters (acknowledge). Execution starts immediately. The command has the following sequence of parameters:

Length	Type	Parameter
1	int	variable index*
1	int	variable type**
2	int	initial value

* the variable index starts with 1 (NOT zero based)
** NOT supported yet since all variables are of the type integer. Set to "1" for future compatibility.

C.6.2 CmdFreeVariable (0x5C)

Using the "CmdFreeVariable" command, a previously declared ICU internal variable can be freed. The command immediately responds with no additional parameters (acknowledge). Execution starts immediately. The command has the following sequence of parameters:

Length	Type	Parameter
1	int	variable index

C.6.3 CmdWriteVariable (0x5D)

Using the "CmdWriteVariable" command, a new value can be written to a previously declared ICU internal variable. The command immediately responds with no additional parameters (acknowledge). Execution starts at the time, specified by the parameter "time stamp". The command has the following sequence of parameters:

Length	Type	Parameter
4	int	time-stamp
1	int	variable index
2	int	variable value

C.6.4 CmdReadVariable (0x5E)

Using the ”CmdReadVariable” command, the value of a previously declared ICU internal variable can be read. The command responds upon execution with the variable index and the requested value as additional parameters. Issued while recording a protocol, the command responds immediately with no additional parameters (acknowledge). Execution starts at the time, specified by the parameter ”time stamp”. The command has the following sequence of parameters:

Length	Type	Parameter
4	int	time-stamp
1	int	variable index

The command’s response has the following sequence of parameters:

Length	Type	Parameter
4	int	time-stamp
1	int	variable index
2	int	variable value*

* for now only variables of the type integer are supported

C.7 Experiment Timer:

The ICU provides an internal timer, which is based on the system clock and therefore absolutely synchronous to all ICU operations. This timer can be used to synchronize and/or correlate the firmware-based operation with PC based operation of an experiment to one overall time line. Timer values are in units of 10 µs. The timer is never reset automatically by the ICU thus granting complete user control.

C.7.1 CmdResetExperimentTimer (0x58)

Using the ”CmdResetExperimentTimer” command, the ICU internal experiment timer can be reset. The command responds upon execution with the last timer value prior to the reset as an additional parameter. Issued while recording a protocol, the command responds immediately with no additional parameters (acknowledge). Execution starts at the time, specified by the parameter ”time stamp”. The command has the following sequence of parameters:

Length	Type	Parameter
4	int	time-stamp

The command's response has the following sequence of parameters:

Length	Type	Parameter
4	int	experiment timer value

C.7.2 CmdSendExperimentTimerValue (0x59)

Using the "CmdSendExperimentTimerValue" command, the ICU internal experiment timer can be readout. The command responds upon execution with the current timer value as an additional parameter. Issued while recording a protocol, the command responds immediately with no additional parameters (acknowledge). Execution starts at the time, specified by the parameter "time stamp". The command has the following sequence of parameters:

Length	Type	Parameter
4	int	time-stamp

The command's response has the following sequence of parameters:

Length	Type	Parameter
4	int	experiment timer value

C.8 RS232:

The ICU can provide several EIA-232 interfaces, which can be used from within a protocol. The number of EIA-232 interfaces can be determined by reading the system configuration ("CmdGetSystemConfig"). Any number from zero to six is possible, depending on the existing hardware. Upon start-up all EIA-232 interfaces are configured with the following settings:

- baud rate: 2400 bits/s
- word length: 8 bit
- stop bits: 1
- parity: none
- hardware control: non
- read mode: buffer

C.8.1 CmdConfigUART (0x60)

Using the "CmdConfigUART" command, each EIA-232 interface (UART) can be reconfigured individually. The settings that were set last are valid until the ICU is powered off. The command immediately responds with no additional parameters. Execution starts at the time, specified by the parameter "time stamp". The command has the following sequence of parameters:

Length	Type	Parameter
4	int	time-stamp
1	int	UART number
1	int	baud rate
1	int	word length
1	int	stop bit
1	int	parity
1	int	read mode
1	int	force read parameter
3	int	read timeout

Possible settings for the individual configuration parameters:

- UART number: 1 to 6 depending on the existing hardware. Read the system configuration (”CmdGetSystemConfig”) to get the number of UARTs.
- baud rate: (decimal - bits/s)
  - 15 = 600 bits/s
  - 14 = 1200 bits/s
  - 7 = 1800 bits/s
  - 13 = 2400 bits/s
  - 6 = 3600 bits/s
  - 12 = 4800 bits/s
  - 5 = 7200 bits/s
  - 11 = 9600 bits/s
  - 4 = 14400 bits/s
  - 10 = 19200 bits/s
  - 3 = 28800 bits/s
  - 9 = 38400 bits/s
  - 2 = 57600 bits/s
  - 8 = 76800 bits/s
  - 1 = 115200 bits/s
  - 0 = 230400 bits/s
- word length: (decimal - bits)
  - 1 = 7 bits*
  - 0 = 8 bits
- stop bits: (decimal - bit(s))
  - 1 = 1 bit
  - 2 = 2 bits
- parity: (decimal - bool)
  - 0 = no
  - 1 = yes*

- read mode: (decimal - mode number)
  - 0 = disable: Data sent from a device connected to the specified UART is discarded. Using the ”CmdReadUART” command will result in a response with no parameters. In this mode, the parameters ”force read parameter” and ”read timeout” have no effect.
  - 1 = buffer: Data sent from a device connected to the specified UART is buffered in a software buffer internal to the ICU. Using the ”CmdReadUART” this buffer can be read resulting in a buffer flush. An error message is generated, when a buffer overflow occurs (see section on ICU Error Responses). In this mode, the parameters ”force read parameter” and ”read timeout” have no effect.
  - 2 = end-byte: Data sent from a device connected to the specified UART is buffered in a software buffer internal to the ICU. An error message is generated, when a buffer overflow occurs (see section on ICU Error Responses). Upon a ”CmdReadUART” command a response to the host PC is generated when a data byte within the ICU’s software buffer equals the specified ”force read parameter” **. If the above does not apply, a response to the host PC is generated when an incoming data byte equals the specified ”force read parameter” within a timeout specified by ”read timeout” **. If none of the above does apply, after the specified timeout, the current content of the software buffer is sent to the host PC regardless whether a data byte equals the specified ”force read parameter” or not, resulting in a buffer flush. The response has the structure of a regular response to the ”CmdReadUART” command.
  - 3 = number of bytes: Data sent from a device connected to the specified UART is buffered in a software buffer internal to the ICU. An error message is generated, when a buffer overflow occurs (see section on ICU Error Responses). Upon a ”CmdReadUART” command a response to the host PC is generated when the number of data bytes within the ICU’s software buffer equals the number specified by the ”force read parameter” **. If the above does not apply, a response to the host PC is generated when the number of incoming bytes equals the number specified by the ”force read parameter” within a timeout specified by ”read timeout” **. If none of the above does apply, after the specified timeout, the current content of the software buffer is sent to the host PC regardless of the current number of data bytes within the ICU’s software buffer, resulting in a buffer flush. The response has the structure of a regular response to the ”CmdReadUART” command.
  - 4 = reserved
  - 5 = automatic end-byte: Data sent from a device connected to the specified UART is buffered in a software buffer internal to the ICU. An error message is generated, when a buffer overflow occurs (see section on ICU Error Responses). An automatic response to the host PC is generated when an incoming data byte equals the specified ”force read parameter” **. The automatic response has the structure of a normal response to the ”CmdReadUART” command. Upon a ”CmdReadUART” command the current content of the software buffer is send to the host PC regardless whether a data byte equal to the specified ”force read parameter” was received or not, resulting in a buffer flush. In this mode, the parameter ”read timeout” has no effect.
  - 6 = automatic number of bytes: Data sent from a device connected to the specified

UART is buffered in a software buffer internal to the ICU. An error message is generated, when a buffer overflow occurs (see section on ICU Error Responses). An automatic response to the host PC is generated when the number of data bytes within the internal software buffer equals the number specified by the ”force read parameter” **. The automatic response has the structure of a normal response to the ”CmdReadUART” command. Upon a ”CmdReadUART” command the current content of the software buffer is send to the host PC regardless of the current number of data bytes within the software buffer, resulting in a buffer flush. In this mode, the parameter ”read timeout” has no effect.

- 7 = automatic byte-timeout***: Data sent from a device connected to the specified UART is buffered in a software buffer internal to the ICU. An error message is generated, when a buffer overflow occurs (see section on ICU Error Responses). When a data byte is received, an automatic timeout specified by ”read timeout” ** is started. If another data byte is received within the timeout, the timeout timer is reset. This applies to all further data bytes that are received from a device connected to the specified UART. An automatic response to the host PC is generated, when the timeout expires, resulting in a buffer flush. The automatic response has the structure of a normal response to the ”CmdReadUART” command. In this mode, the parameter ”force read parameter” has no effect.

- force read parameter: (decimal - hex-byte): this parameter is only used in read modes 2, 3, 5 and 6. In all other modes this parameter has no effect. Depending on the selected ”read mode” is has the following two different meanings:
  - modes 2 and 5: Incoming data bytes are compared with the ”force read parameter” and a response to the host PC is initiated containing all previously sent data bytes, including the data byte that initiated the response.
  - modes 3 and 6: When the number of data bytes in the ICU’s internal software buffer is equal to the number that is specified by the ”force read parameter”, a response to the host PC is initiated containing all previously sent data bytes, including the data byte that initiated the response.
- read timeout: (decimal - 10 µs): this parameter is only used in read modes 2, 3 and (temporarily) 7. In all other modes this parameter has no effect. ”read timeout” specifies the timeout after which a response to the host PC is definitely generated regardless of the comparison results with the ”force read parameter”. The timeout starts at the execution time of the ”CmdReadUART” command (modes 2 and 3) or at the time a data byte, which was sent by a device connected to the specified UART was received (mode 7).

* NOT supported yet; send ”0” for future compatibility
** see description of a later parameter
*** subject to change: functionality is not fully tested

C.8.2 CmdWriteUART (0x61)

Using the ”CmdWriteUART” command any number of arbitrary bytes and/or characters can be send to an external device from within a protocol via the specified EIA-232 interface (UART).

The communication settings, which were set last, are used (see ”CmdConfigUART”). The command immediately responds with no additional parameters. Execution starts at the time, specified by the parameter ”time stamp”. The command has the following sequence of parameters:

Length	Type	Parameter
4	int	time-stamp
1	int	UART number
x		arbitrary parameters that are to be send

C.8.3 CmdReadUART (0x62)

Responses of external devices connected to any of the ICU's EIA-232 interfaces are handled depending on the current read mode. The communication settings, which were set last (see ”CmdConfigUART”), are used for reading responses. Unless responses are immediately passed to the host PC, they are locally buffered. Using the ”CmdReadUART” command the ICU's UART buffer of each existing EIA-232 interface can be read individually. The command responds upon execution with the content (if any) of the specified UART buffer as additional parameters. Issued while recording a protocol, the command responds immediately with no additional parameters (acknowledge). Execution starts at the time, specified by the parameter ”time stamp”. The command has the following sequence of parameters:

Length	Type	Parameter
4	int	time-stamp
1	int	UART number

The command's response has the following sequence of parameters:

Length	Type	Parameter
1	int	UART number
x		arbitrary parameters (received bytes)

C.9 Analog I/O:

The ICU can provide several analog input and/or output channels, which can be set or read individually from within a protocol. The number of analog channels as well as the characteristics (in/out, Umin, Umax, R = resolution) can be determined by reading the system configuration (see ”CmdGetSystemConfig”). Upon start-up all analog interfaces are configured with the following settings:
analog in: depending on external voltage
analog out: ~$0V$

The channel number that selects the analog channel for a write or read operation starts counting at one (first channel = 1).

For setting or reading an anlog channel a 2 byte integer parameter is used that correlates to the voltage level. The unit of this parameter is ”converter counts”. For an $R = 16$-bit converter $Umin$ therefore corresponds to 0 and $Umax$ to $(2^6 - 1) = 65535 = 0xFFFF$.

C.9.1 CmdWriteAnalogOut (0x63)

Using the ”CmdWriteAnalogOut” command, any of the analog output channels can be set individually. The specified analog output channel can be set to any voltage level between $Umin$ and $Umax$ with resolution R. The command immediately responds with no additional parameters. Execution starts at the time, specified by the parameter ”time stamp”. The command has the following sequence of parameters:

Length	Type	Parameter
4	int	time-stamp
1	int	channel number
2	int	voltage level

C.9.2 CmdReadAnalogOut (0x64)

Using the ”CmdReadAnalogOut” command, the current value of any of the analog output channels can be read individually. The returned value corresponds to the voltage level as described above. The command responds upon execution with the analog channel number and the requested value as additional parameters. Issued while recording a protocol, the command responds immediately with no additional parameters (acknowledge). Execution starts at the time, specified by the parameter ”time stamp”. The command has the following sequence of parameters:

Length	Type	Parameter
4	int	time-stamp
1	int	channel number

The command’s response has the following sequence of parameters:

Length	Type	Parameter
1	int	channel number
2	int	voltage level

C.9.3 CmdReadAnalogIn (0x65)

The command responds upon execution with the channel number and the voltage level of the specified analog input channel as additional parameters. Issued while recording a protocol, the command responds immediately with no additional parameters (acknowledge). Execution starts at the time, specified by the parameter ”time stamp”. The command has the following sequence of parameters:

Length	Type	Parameter
4	int	time-stamp
1	int	channel number

The command's response has the following sequence of parameters:

Length	Type	Parameter
1	int	channel number
2	int	voltage level

C.10 Digital I/O:

The ICU supports several banks of digital in- and/or outputs, each with eight bits. The following is the list of existing I/O lines for a Basic iMIC ICU:

bank	intern - DIn	extern - DIn	intern - DOut	extern - DOut
1	BP_DIO1-8	bit 0-7*	BP_DIO1-8	bit 0-7*
2	BP_DIO9-16	bit 0-7**	BP_DIO9-16	bit 0-7**
3	BP_DIN1-8	bit0 = DIn1, bit1 = DIn2, bit2 = TrigI, bit3 = PiezoClip, bit4 = Reserved, bit5 = Reserved, bit6 = Reserved, bit7 = Reserved	BP_DOUT1-8	bit0 = DOut1, bit1 = DOut2, bit2 = TrigO, bit3 = PiezoCtrl, bit4 = Reserved, bit5 = Reserved, bit6 = Reserved, bit7 = Reserved
4	FP_DIO17-20	bit0 = DIO17**, bit1 = DIO18*, bit2 = DIO19*, bit3 = DIO20*, bit4 = Reserved, bit5 = Reserved, bit6 = Reserved, bit7 = Reserved	FP_DIO17-20	bit0 = DIO17**, bit1 = DIO18*, bit2 = DIO19*, bit3 = DIO20*, bit4 = Reserved, bit5 = Reserved, bit6 = Reserved, bit7 = Reserved

* currently pre-configured as a digital OUTPUT;
** currently pre-configured as a digital INPUT

C.10.1 CmdWriteDigitalOut (0x66)

After configuring a digital I/O bank as output, the values of the individual I/O lines can be set with the "CmdWriteDigitalOut" command. The command immediately responds with no additional parameters. Execution starts at the time, specified by the parameter "time stamp". The command has the following sequence of parameters:

Length	Type	Parameter
4	int	time-stamp
1	int	bank
1	int	line-mask
1	int	value-mask

The parameters, bank, line-mask and value-mask specify the I/O line(s), whose levels are to be set. The parameter ”bank” obviously specifies the I/O line(s) bank. The parameter ”line-mask” selects the I/O line(s) whose level(s) are to be set. The parameter ”value-mask” specifies the value(s) of the selected I/O line(s).

Example:
bank = 0x01 selects bank 1.
line-mask = 0x35 = 0011 0101B selects I/O lines 1, 3, 5 and 6.
value-mask = 0x14 = 0001 0100B sets I/O line 1 low, sets I/O line 3 high, sets I/O line 5 high and sets I/O line 6 low.

C.10.2 CmdReadDigitalIn (0x67)

Permanent input lines or I/O lines that were previously configured as inputs can be read by issuing a ”CmdReadDigitalIn” command. The command responds upon execution with the requested bit-pattern of the specified I/O bank as an additional parameter. Issued while recording a protocol, the command responds immediately with no additional parameters (acknowledge). Execution starts at the time, specified by the parameter ”time stamp”. The command has the following sequence of parameters:

Length	Type	Parameter
4	int	time-stamp
1	int	bank
1	int	line-mask

The parameters bank and line-mask specify the I/O line(s), whose levels are to be read. The parameter ”bank” obviously specifies the I/O line(s) bank. The parameter ”line-mask” selects the I/O line(s) whose level(s) are to be read.

Note: When reading a single bit, it might be desirable to have either a ’0’ or a ’1’ as the result of a ”CmdReadDigitalIn” command. In this case the returned value can be transformed by the C-statement: (Value $>$ 0).

The command’s response has the following sequence of parameters:

Length	Type	Parameter
1	int	bank
1	int	value-mask

Abkürzungsverzeichnis

ADC	**A**nalog to **D**igital **C**onverter
AGND	**A**nalog **G**rou**ND**
AOTF	**A**cousto-**O**ptical **T**unable **F**ilter
API	**A**pplication **P**rogram **I**nterface
ASAM	**A**nalog-**S**ignal **A**cquisition **M**odule
ASCII	**A**merican **S**tandard **C**ode for **I**nformation **I**nterchange
ASI	**A**dvanced **S**erial **I**nterface
Bit	**Bi**nary digi**t**; kleinste Einheit einer Datenmenge
BIZ	**B**io**I**maging **Z**entrum der LMU
BNC	**B**ayonet **N**eill **C**oncelman - Steckverbinder; benannt nach den Erfindern Paul Neill (Bell Labs) und Carl Concelman (Amphenol)
Byte	Byte; Speichermengeneinheit, 1 Byte entsprechen 2^3 Bit = 8 Bit
CAN	**C**ontroller **A**rea **N**etwork
CCD	**C**harge-**C**oupled **D**evice
CPLD	**C**omplex **P**rogrammable **L**ogic **D**evice
CRC	**C**yclic **R**edundancy **C**heck
DAC	**D**igital to **A**nalog **C**onverter
DMA	**D**irect **M**emory **A**ccess
DGND	**D**igital **G**rou**ND**
DIN	**D**eutsche **I**ndustrie **N**orm
DIP	**D**ual **I**n-line **P**ackage
DLL	**D**ynamic **L**ink **L**ibrary
DPSSL	**D**iode **P**umped **S**olid **S**tate **L**aser
DSC	Digitale Scanner Kontrollsteuerung; Englisch: **D**igital **S**can **C**ontrol
DSP	**D**igitaler **S**ignal **P**rozessor
D-Sub	**D** **Sub**miniature; D-förmiger Steckverbinder nach MIL Standard 24308
EEPROM	**E**lectrically **E**rasable **P**rogrammable **R**ead **O**nly **M**emory
EIA	**E**lectronic **I**ndustries **A**lliance; Die EIA-232 Schnittstelle ist besser bekannt unter ihrem ursprünglichen Namen „RS232 Schnittstelle“
EMBL	**E**uropean **M**olecular **B**iology **L**aboratory
FIFO	**F**irst **I**n – **F**irst **O**ut
FLIP	**F**luorescence **L**oss **I**n **P**hotobleaching
FRAP	**F**luorescence **R**ecovery **A**fter **P**hotobleaching

FRET	**F**luorescence **R**esonance **E**nergy **T**ransfer
GFP	**G**reen **F**luorescent **P**rotein
GND	**G**rou**ND**
GPE	**G**raphischer **P**rotokoll **E**ditor
GUI	**G**raphical **U**ser **I**nterface
HCS	**H**igh **C**ontent **S**creening
HE	**H**öhen **E**inheit; Einheit für die Höhe einer „Frontplatte“: 1HE ≙ 1¾Zoll = 44,45 mm
HTS	**H**igh **T**hroughput **S**creening
I²C	**I**nter-**I**ntegrated **C**ircuit
ICU	**I**ntegration **C**ontrol **U**nit
IDES	**I**nter**D**igitated **E**lectrode **S**tructures
IEEE	**I**nstitute of **E**lectrical and **E**lectronics **E**ngineers
iMIC	**i**maging **MIC**roscope
IMR	**I**ntelligent **M**icroplate **R**eader
IT	**I**solated-**T**arget
I/O	**I**nputs/**O**utputs
kHz	**K**ilo **H**er**z**; Einheit für Taktraten, entspricht 10^3 Herz
LabVIEW	**Lab**oratory **V**irtual **I**nstrument **E**ngineering **W**orkbench; graphische Programmiersprache von National Instruments, Austin, Texas, USA
LCD	**L**iquid **C**rystal **D**isplay
LED	**L**ight **E**mitting **D**iode
LME	**L**ehrstuhl für **M**edizinische **E**lektronik
LMU	**L**udwig-**M**aximilians-**U**niversität München
LSM	**L**aser-**S**canning **M**icroscopy
MB	**M**ega **B**yte; Speichermengeneinheit, 1 MB entsprechen 10^6 Bytes
MHz	**M**ega **H**er**z**; Einheit für Taktraten, entspricht 10^6 Herz
MISO	**M**aster **I**n; **S**lave **O**ut
mm	**m**illi **M**eter; Längeneinheit: 1mm entspricht 10^{-3} Meter
MOSI	**M**aster **O**ut; **S**lave **I**n
ms	**m**illi **S**ekunde; Zeiteinheit: 1ms entspricht 10^{-3} Sekunden
nm	**n**ano **M**eter; Längeneinheit: 1nm entspricht 10^{-9} Meter
NME	**N**ew **M**olecular **E**ntities
O_2	Summenformel des molekularen Sauerstoffs
OEM	**O**riginal **E**quipment **M**anufacturer

PC	**P**ersonal **C**omputer
PCB	**P**rinted **C**ircuit **B**oard
PCI	**P**eripheral **C**omponent **I**nterconnect
pH-Wert	Maß für die Stärke der sauren bzw. basischen Wirkung einer Lösung: negative dekadische Logarithmus der Oxoniumionenkonzentration
RJ	**R**egistered **J**ack; nach Standard der US-amerikanischen Federal Communications Commission (FCC)
s	**S**ekunde
SDK	**S**oftware **D**evelopment **K**it
SEROS	**S**mall **E**mbedded **R**eal-Time **O**perating **S**ystem
SMB	**S**ub**M**iniature version **B**
SPI	**S**erial **P**eripheral **I**nterface
SRAM	**S**tatic **R**andom **A**ccess **M**emory
TDI	**T**ime **D**elayed **I**ntegration
TE	**T**eil **E**inheit; Einheit für die Breite einer „Frontplatte“: 1 TE ≙ $1/5$ Zoll = 5,08 mm
TIRF	**T**otal **I**nternal **R**eflection **F**luorescence
TMSI	**T**ime **M**ultiplexed **S**erial **I**nterface
TTCAN	**T**ime-**T**riggered communication on **CAN**
TUM	**T**echnische **U**niversität **M**ünchen
UART	**U**niversal **A**synchronous **R**eceiver **T**ransmitter
µHTS	mikro (**µ**) **H**igh **T**hroughput **S**creening
µl	mikro (**µ**) **L**iter; Volumeneinheit: 1 µl entspricht 10^{-6} Liter
µm	mikro (**µ**) **M**eter; Längeneinheit: 1 µm entspricht 10^{-6} Meter
µs	mikro (**µ**) **S**ekunde; Zeiteinheit: 1 µs entspricht 10^{-6} Sekunden
USB	**U**niversal **S**erial **B**us
V	**V**olt;internationale Einheit der elektrischen Spannung
VBA	**V**isual **B**asic for **A**pplications
VDE	**V**erband **D**eutscher **E**lektriker; seit 1998: Verband der Elektrotechnik, Elektronik und Informationstechnik e.V.
XML	E**x**tensible **M**arkup **L**anguage
ZB	**Z**ell-**B**asiert

Abbildungsverzeichnis

Tabellenverzeichnis

Literaturverzeichnis

[1] ABDULLAHA, M. ; MOHAMAD-SALEH, J. ; FATHINUL-SYAHIR, A. ; MOHD-AZEMI, B. : Discrimination and classification of fresh-cut starfruits (Averrhoa carambola L.) using automated machine vision system. In: *Journal of Food Engineering* 76 (2006), Oktober, Nr. 4, S. 506–523

[2] ALBERTS, B. ; ROBERTS, K. ; LEWIS, J. ; RAFF, M. ; WALTER, P. ; JOHNSON, A. : *Molecular Biology of the Cell*. 4th. Garland Pub, März 2002

[3] AMNIS: Time Delay Integration: enabling high sensitivity detection for Imaging-in-Flow on the ImageStream® 100 Cell Analysis System. In: *Amnis - Technology Report*, 2004

[4] ANTONARAKIS, S. : 10 Years of Genomics, Chromosome 21, and Down Syndrome. In: *Genomics* 51 (1998), Juli, Nr. 1, S. 1–16

[5] ASHKIN, A. ; DZIEDZIC, J. M. ; BJORKHOLM, J. E. ; CHU, S. : Observation of a single-beam gradient force optical trap for dielectric particles. In: *Optics Letters* 11 (1986), Mai, Nr. 5, S. 288–290

[6] ASHKIN, A. ; DZIEDZIC, J. M. ; YAMANE, T. : Optical trapping and manipulation of single cells using infrared laser beams. In: *Nature* 330 (1987), Dezember, S. 769–771

[7] AUER, M. ; MOORE, K. J. ; MEYER-ALMES, F. J. ; GUENTHER, R. ; POPE, A. J. ; STOECKLI, K. A.: Fluorescence correlation spectroscopy: lead discovery by miniaturized HTS. In: *Drug Discovery Today* 3 (1998), Oktober, Nr. 10, S. 457–465

[8] AXELROD, D. ; THOMPSON, N. L. ; BURGHARDT, T. P.: Total internal inflection fluorescent microscopy. In: *Journal of Microscopy* 129 (1983), Januar, Nr. 1, S. 19–28

[9] AXELROD, D. : Total Internal Reflection Fluorescence Microscopy in Cell Biology. In: *Traffic* 2 (2001), November, Nr. 11, S. 764

[10] BACKER, R. : *Assembler - Maschinennahes Programmieren von Anfang an*. 1. Rowohlt TB., 2003

[11] BALIS, F. M.: Evolution of Anticancer Drug Discovery and the Role of Cell-Based Screening. In: *Journal of the National Cancer Institute* 94 (2002), Januar, Nr. 2, S. 78–79

[12] BARABASZ, A. ; FOLEY, B. ; OTTO, J. C. ; SCOTT, A. ; RICE, J. : The Use of High-Content Screening for the Discovery and Characterization of Compounds That Modulate Mitotic Index and Cell Cycle Progression by Differing Mechanisms of Action. In: *ASSAY and Drug Development Technologies* 4 (2006), April, Nr. 2, S. 153–163

[13] BELKEBIR, K. ; CHAUMET, P. C. ; SENTENAC, A. : Superresolution in total internal reflection tomography. In: *Journal of the Optical Society of America* 22 (2005), September, Nr. 9, S. 1889–1897. – abbreviation: J. Opt. Soc. Am.

[14] BERGMANN, L. ; SCHAEFER, C. ; NIEDRIG, H. : *Lehrbuch der Experimentalphysik*. 10. Berlin : Gruyter, Oktober 2004 (3)

[15] BERNEY, C. ; DANUSER, G. : FRET or No FRET: A Quantitative Comparison. In: *Biophysical Journal* 84 (2003), Juni, S. 3992–4010

[16] BERNS, M. W. ; TADIR, Y. ; LIANG, H. ; TROMBERG, B. : Laser scissors and tweezers. In: *Methods Cell Biology* 55 (1998), S. 71–98

[17] BICKER, G. : Transmitter-induced calcium signalling in cultured neurons of the insect brain. In: *Journal of Neuroscience Methods* 69 (1996), Oktober, Nr. 1, S. 33–41

[18] BISSANTZ, C. ; FOLKERS, G. ; ROGNAN, D. : Protein-Based Virtual Screening of Chemical Databases. 1. Evaluation of Different Docking/Scoring Combinations. In: *Journal of Medicinal Chemistry* 43 (2000), Nr. 25, S. 4759–4767

[19] BISWAS, S. ; AKEY, J. M.: Genomic insights into positive selection. In: *Trends in Genetics* 22 (2006), August, Nr. 8, S. 437–446

[20] BLAKE, R. A.: Cellular screening assays using fluorescence microscopy. In: *Current Opinion in Pharmacology* 1 (2001), Oktober, Nr. 5, S. 533–539

[21] BOWEN, W. P. ; PAYNE, S. L.: High-content screening in oncology using fluorescence microplate cytometry. In: *Nature Methods* 2 (2005), Dezember, S. 1–2

[22] BRAECKMANS, K. ; PEETERS, L. ; SANDERS, N. N. ; DE SMEDT, S. C. ; DEMEESTER, J. : Three-Dimensional Fluorescence Recovery after Photobleaching with the Confocal Scanning Laser Microscope. In: *Biophysical Journal* 85 (2003), Nr. 4, S. 2240–2252

[23] BRAKENHOFF, G. J. ; BLOM, P. ; BARENDS, P. : Confocal scanning light microscopy with high aperture immersion lenses. In: *Journal of Microscopy* 117 (1979), S. 219–232

[24] BRISCHWEIN, M. ; GROTHE, H. ; OTTO, A. M. ; RESSLER, J. ; LOB, V. ; WIEST, J. ; WOLF, B. : Möglichkeiten und Grenzen der Mikrosensortechnologie in zellulärer Diagnostik und Pharmascreening. In: *Chemie Ingenieur Technik* 77 (2005), November, Nr. 12, S. 1955–1959

[25] BURDINE, L. ; KODADEK, T. : Target identification in chemical genetics: the (often) missing link. In: *Chemistry and Biology* 11 (2004), Mai, S. 593–597

[26] BUTCHER, E. C. ; BERG, E. L. ; KUNKEL, E. J.: Systems biology in drug discovery. In: *Nature Biotechnology* 22 (2004), Oktober, Nr. 10, S. 1253–1259

[27] CAMPBELL, A. K.: Book Review: Intracellular Calcium–They Think It's All Over! It Isn't Quite Yet! In: *Science STKE* 1999 (1999), November, Nr. 7, S. 1

[28] CARPENTER, A. E. ; SABATINI, D. M.: Systematic genome-wide screens of gene function. In: *Nature Reviews Genetics* 5 (2004), Januar, Nr. 1, S. 23–33

[29] COMLEY, J. : High content screening: emerging importance of novel reagents/probes and pathway analysis. In: *Drug Discovery World* Summer (2005), S. 31–53

[30] COMLEY, J. ; FOX, S. : Growing market for high content analysis tools. In: *Drug Discovery World* Spring (2004), S. 25–34

[31] CONSTANS, A. : Automated Microscopy gets a new shape. In: *The Scientist* 18 (2004), März, Nr. 6, S. 41

[32] CRAIG, A.-M. ; MALEK, M. : Market structure and conduct in the pharmaceutical industry. In: *Pharmacology and Therapeutics* 66 (1995), Nr. 2, S. 301–337

[33] DELISI, C. : Genomes: 15 Years Later. In: *Human Genome News* 11 (2001), Juli, Nr. 3-4, S. 5–6

[34] Kap. Two-Photon Molecular Excitation in Laser-Scanning Microscopy In: DENK, W. ; PISTON, D. W. ; WEBB, W. W.: *Handbook of Biological Confocal Microscopy*. New York : Plenum Press, 1995

[35] DENK, W. ; STRICKLER, J. H. ; WEBB, W. W.: Two-photon laser scanning fluorescence microscopy. In: *Science* 248 (1990), April, Nr. 4951, S. 73–76

[36] DESHPANDE, R. R. ; HEINZLE, E. : On-line oxygen uptake rate and culture viability measurement of animal cell culture using microplates with integrated oxygen sensors. In: *Biotechnology Letters* 26 (2004), Mai, Nr. 9, S. 763–767

[37] DIMASI, J. A. ; HANSEN, R. W. ; GRABOWSKI, H. G.: The price of innovation: new estimates of drug development costs. In: *Journal of Health Economics* 22 (2003), März, Nr. 2, S. 151–185

[38] DOVE, A. : Screening for content—the evolution of high throughput. In: *Nature Biotechnology* 21 (2003), S. 859–864

[39] DREWS, J. : Drug Discovery: A Historical Perspective. In: *Science* 287 (2000), März, S. 1960–1964

[40] DUMONT, J. E. ; PÉCASSE, F. ; MAENHAUT, C. : Crosstalk and specificity in signalling: Are we crosstalking ourselves into general confusion? In: *Cellular Signalling* 13 (2001), Juli, Nr. 7, S. 457–463

[41] EGGERT, U. S. ; MITCHISON, T. J.: Small molecule screening by imaging. In: *Current Opinion in Chemical Biology* 10 (2006), Juni, Nr. 3, S. 232–237

[42] EGNER, A. ; ANDRESEN, V. ; HELL, S. W.: Comparison of the axial resolution of practical Nipkow-disk confocal fluorescence microscopy with that of multifocal multiphoton microscopy: theory and experiment. In: *Journal of Microscopy* 206 (2002), April, Nr. 1, S. 24

[43] EHRET, R. ; BAUMANN, W. ; BRISCHWEIN, M. ; SCHWINDE, A. ; STEGBAUER, K. ; WOLF, B. : Monitoring of cellular behaviour by impedance measurements on interdigitated electrode structures. In: *Biosensors & Bioelectronics* 12 (1997), Nr. 1, S. 29–41. – 0956-5663 Journal Article

[44] EHRET, R. ; BAUMANN, W. ; BRISCHWEIN, M. ; SCHWINDE, A. ; WOLF, B. : On-line control of cellular adhesion with impedance measurements using interdigitated electrode structures. In: *Medical & Biological Engineering & Computing* 36 (1998), Nr. 3, S. 365–370

[45] EMMERT-BUCK, M. R. ; BONNER, R. F. ; SMITH, P. D. ; CHUAQUI, R. F. ; ZHUANG, Z. ; GOLDSTEIN, S. R. ; WEISS, R. A. ; LIOTTA, L. A.: Laser Capture Microdissection. In: *Science* 274 (1996), November, Nr. 5289, S. 998

[46] ENTZEROTH, M. : Emerging trends in high-throughput screening. In: *Current Opinion in Pharmacology* 3 (2003), Oktober, Nr. 5, S. 522–529

[47] FOG, A. ; P.BUCK, R. : Electronic Semiconducting Oxides as pH Sensors. In: *Sensors and Actuators* 5 (1984), S. 137–146

[48] FOLCH, A. ; TONER, M. : Microengineering of cellular interactions. In: *Annual Review of Biomedical Engineering* 2 (2000), August, S. 227–256

[49] FOX, S. ; FARR-JONES, S. ; YUND, M. A.: High Throughput Screening for Drug Discovery: Continually Transitioning into New Technology. In: *Journal of Biomolecular Screening* 4 (1999), August, Nr. 4, S. 183–186

[50] FRANKLIN, B. ; LABAREE, L. W. (Hrsg.) ; BELL, W. J. (Hrsg.) ; BOATFIELD, H. C. (Hrsg.) ; FINEMAN, H. H. (Hrsg.): *The Papers of Benjamin Franklin: January 1, 1745 through June 30, 1750*. Bd. 3. Yale University Press, Januar 1961

[51] GEDDES, L. A. ; BAKER, L. E.: *Principles of Applied Biomedical Instrumentation*. 3. Wiley-Interscience, August 1989

[52] GEISLER, T. ; LOB, V. ; WOLF, B. ; UHL, R. : Control-Architecture for Cell-Based High-Content and High Throughput Screening. In: *IFMBE Proceedings* Bd. 14, 2006, S. 2152–2155

[53] GEISLER, T. ; LOB, V. ; WOLF, B. ; UHL, R. : Microscope-based High-Throughput and High-Content Screening with the “Intelligent Microplate Reader”. In: *Conference Guide 9th MipTec 2006*, 2006

[54] GEISLER, T. ; RESSLER, J. ; HARZ, H. ; WOLF, B. ; UHL, R. : Automated Multiparametric Platform for High-Content and High-Throughput Analytical Screening on Living Cells. In: *IEEE Transactions on Automation Science and Engineering* 3 (2006), April, Nr. 2, S. 169–176

[55] GEISLER, T. ; WOLF, B. ; UHL, R. : Modular Real-Time Control Platform for Cell-Based High-Content and High-Throughput Screening. In: *BMT Proceedings*, 2006

[56] GIULIANO, K. A. ; DEBIASIO, R. L. ; DUNLAY, R. T. ; GOUGH, A. ; VOLOSKY, J. M. ; ZOCK, J. ; PAVLAKIS, G. N. ; TAYLOR, D. L.: High-Content Screening: A New Approach to Easing Key Bottlenecks in the Drug Discovery Process. In: *Journal of Biomolecular Screening* 2 (1997), Nr. 4, S. 249–259

[57] GIULIANO, K. A. ; HASKINS, J. R. ; TAYLOR, D. L.: Advances in High Content Screening for Drug Discovery. In: *ASSAY and Drug Development Technologies* 1 (2003), August, Nr. 4, S. 565–577

[58] GRIER, D. G.: A revolution in optical manipulation. In: *Nature* 424 (2003), August, S. 810–816

[59] GRIMNES, S. ; ØRJAN MARTINSEN: *Bioimpedance and Bioelectricity Basics*. Academic Press, März 2000

[60] GROSCHE, J. ; MATYASH, V. ; MÖLLER, T. ; VERKHRATSKY, A. ; REICHENBACH, A. ; KETTENMANN, H. : Microdomains for neuron–glia interaction: parallel fiber signaling to Bergmann glial cells. In: *Nature Neuroscience* 2 (1999), Februar, Nr. 2, S. 139–143

[61] GUSTAFSSON, M. G. L.: Surpassing the lateral resolution limit by a factor of two using structured illumination microscopy. In: *Journal of Microscopy* 198 (2000), Mai, Nr. 2, S. 82

[62] GUSTAFSSON, M. G. L.: Nonlinear structured-illumination microscopy: Wide-field fluorescence imaging with theoretically unlimited resolution. In: *Proceedings of the National Academy of Sciences of the USA* 102 (2005), S. 13081–13086

[63] HANN, M. M. ; OPREA, T. I.: Pursuing the leadlikeness concept in pharmaceutical research. In: *Current Opinion in Chemical Biology* 8 (2004), Juni, Nr. 3, S. 255–263

[64] HARAGUCHI, T. ; DING, D. Q. ; YAMAMOTO, A. ; KANEDA, T. ; KOUJIN, T. ; HIRAOKA, Y. : Multiple-color fluorescence imaging of chromosomes and microtubules in living cells. In: *Cell Structure and Function* 24 (1999), Oktober, Nr. 5, S. 291–298

[65] HAUPTS, U. ; ASHMAN, S. ; TURCONI, S. ; BINGHAM, R. ; WHARTON, C. ; HUTCHINSON, J. ; CAREY, C. ; MOORE, K. J. ; POPE, A. J.: Single-Molecule Detection Technologies in Miniaturized High-Throughput Screening: Fluorescence Intensity Distribution Analysis. In: *Journal of Biomolecular Screening* 8 (2003), Nr. 1, S. 19–33

[66] HEINTZMANN, R. ; JOVIN, T. M. ; CREMER, C. : Saturated patterned excitation microscopy-a concept for optical resolution improvement. In: *Journal of the Optical Society of America* 19 (2002), August, Nr. 8, S. 1599–1609

[67] HERTZBERG, R. P. ; POPE, A. J.: High-throughput screening: new technology for the 21st century. In: *Current Opinion in Chemical Biology* 4 (2000), August, Nr. 4, S. 445–451

[68] HOLST, G. C.: *Electro-Optical Imaging System Performance*. 2. Jcd Publishing and Spie Optical Engineering P, April 2000

[69] HOOD, L. ; HEATH, J. R. ; PHELPS, M. E. ; LIN, B. : Systems Biology and New Technologies Enable Predictive and Preventative Medicine. In: *Science* 306 (2004), Oktober, Nr. 5696, S. 640–643

[70] HOPPE, A. ; CHRISTENSEN, K. ; SWANSON, J. A.: Fluorescence Resonance Energy Transfer-Based Stoichiometry in Living Cells. In: *Biophysical Journal* 83 (2002), Dezember, S. 3652–3664

[71] HUANG, S.-H. ; TSENG, F.-G. : Development of a monolithic total internal reflection-based biochip utilizing a microprism array for fluorescence sensing. In: *Journal of Micromechanics and Microengineering* 15 (2005), Dezember, Nr. 12, S. 2235–2242

[72] HUANG, S. ; INGBER, D. E.: The structural and mechanical complexity of cell-growth control. In: *Nature Cell Biology* 1 (1999), September, S. E131–E138

[73] ISENBERG, G. ; BIELSER, W. ; MEIER-RUGE, W. ; REMY, E. : Cell surgery by laser micro-dissection: a preparative method. In: *Journal of Microscopy* 107 (1976), Nr. 1, S. 19–24

[74] JAHR, I. : *Lexikon der industriellen Bildverarbeitung*. Spurbuchverlag, Oktober 2003

[75] JAISWAL, J. K. ; MATTOUSSI, H. ; MAURO, J. M. ; SIMON, S. M.: Long-term multiple color imaging of live cells using quantum dot bioconjugates. In: *Nature Biotechnology* 21 (2002), Januar, S. 47–51

[76] JESSE, K. : *Femtosekundenlaser: Einführung in die Technologie der ultrakurzen Lichtimpulse*. Springer, März 2005

[77] JÄGER, S. ; GARBOW, N. ; KIRSCH, A. ; PRECKEL, H. ; GANDENBERGER, F. U. ; HERRENKNECHT, K. ; RÜDIGER, M. ; HUTCHINSON, J. P. ; BINGHAM, R. P. ; RAMON, F. ; BARDERA, A. ; MARTIN, J. : A Modular, Fully Integrated Ultra-High-Throughput Screening System Based on Confocal Fluorescence Analysis Techniques. In: *Journal of Biomolecular Screening* 8 (2003), Dezember, Nr. 6, S. 648–659

[78] JOHN, G. T. ; HEINZLE, E. : Quantitative screening method for hydrolases in microplates using pH indicators: determination of kinetic parameters by dynamic pH monitoring. In: *Biotechnology and Bioengineering* 72 (2001), März, Nr. 6, S. 620–627

[79] JOHN, G. T. ; GOELLING, D. ; KLIMANT, I. ; SCHNEIDER, H. ; HEINZLE, E. : PH-sensing 96-well microtitre plates for the characterization of acid production by dairy starter cultures. In: *Journal of Dairy Research* 70 (2003), August, Nr. 3, S. 327–333

[80] JOHN, G. T. ; KLIMANT, I. ; WITTMANN, C. ; HEINZLE, E. : Integrated optical sensing of dissolved oxygen in microtiter plates: a novel tool for microbial cultivation. In: *Biotechnology and Bioengineering* 81 (2003), März, Nr. 7, S. 829–836

[81] KAARTINEN, J. ; HÄTÖNEN, J. ; HYÖTYNIEMI, H. ; MIETTUNEN, J. : Machine-vision-based control of zinc flotation—A case study. In: *Control Engineering Practice* 14 (2006), Dezember, Nr. 12, S. 1455–1466

[82] KATZ, L. C. ; DALVA, M. B.: Scanning laser photostimulation: a new approach for analyzing brain circuits. In: *Journal of Neuroscience Methods* 54 (1994), Oktober, Nr. 2, S. 205–218

[83] KERNIGHAN, B. W. ; RITCHIE, D. : *The C. Programming Language*. 2. Prentice Hall, April 1988

[84] KERR, R. ; LEV-RAM, V. ; BAIRD, G. ; VINCENT, P. ; TSIEN, R. Y. ; SCHAFER, W. R.: Optical Imaging of Calcium Transients in Neurons and Pharyngeal Muscle of C. elegans. In: *Neuron* 26 (2000), Juni, Nr. 3, S. 583–594

[85] KIRSON, E. D. ; YAARI, Y. : A novel technique for micro-dissection of neuronal processes. In: *Journal of Neuroscience Methods* 98 (2000), Juni, Nr. 2, S. 119–122

[86] KITANO, H. : Computational systems biology. In: *Nature* 420 (2002), November, S. 206–210

[87] KÖNIG, K. : Multiphoton microscopy in life sciences. In: *Journal of Microscopy* 200 (2000), November, Nr. 2, S. 83–104

[88] KOECHNER, W. : *Solid-State Laser Engineering*. Springer, 1999

[89] KRAUS, M. ; WOLF, B. : *Structured Biological Modelling- A new approach to biophysical cell biology*. Boca Raton, Florida : CRC Press, Inc., Juli 1995. – . Boca Raton, Florida: CRC Press, Inc. S

[90] LANDER, E. S. ; WEINBERG, R. A.: Genomics: journey to the center of biology. In: *Science* 287 (2000), März, Nr. 5459, S. 1777–1782

[91] LANGER, T. ; HOFFMANN, R. : Virtual Screening An Effective Tool for Lead Structure Discovery. In: *Current Pharmaceutical Design* 7 (2001), Mai, Nr. 7, S. 509–527

[92] LAWRENZ, W. : *CAN Controller Area Network. Grundlagen und Praxis*. 4. Hüthig, Juli 2000

[93] LENZ, G. R. ; NASH, H. M. ; JINDAL, S. : Chemical ligands, genomics and drug discovery. In: *Drug Discovery Today* 5 (2000), April, Nr. 4, S. 145–156

[94] LIOTTA, L. A. ; FERRARI, M. ; PETRICOIN, E. : Clinical proteomics: Written in blood. In: *Nature* 425 (2003), Oktober, Nr. 905, S. 905

[95] LOB, V. ; BRISCHWEIN, M. ; H. ; GROTHE ; KAUFMANN, J. R. K. ; WOLF, B. : Cell-based Assays: Mikrosensorarray-basiertes Screening an lebenden Zellen und Geweben. In: *BIOspektrum* Sonderausgabe 11 (2005), S. 511–512

[96] LOB, V. ; GEISLER, T. ; RESSLER, J. ; GROTHE, H. ; BRISCHWEIN, M. ; WOLF, B. : Multiparametric automated screening system based on a 24-Well-Microplate for living cells and tissues. In: *Conference Guide 9th MipTec 2006*, 2006

[97] LOB, V. ; GEISLER, T. ; WOLF, B. ; UHL, R. : Multiparametric automated screening system based on a 24-Well-Microplate for living cells and tissues. In: *IFMBE Proceedings* Bd. 14, 2006, S. 557–560

[98] MAIT, J. N. ; ATHALE, R. ; VAN DER GRACHT, J. : Evolutionary paths in imaging and recent trends. In: *Optics Express* 11 (2003), September, Nr. 18, S. 2093–2101

[99] MAJOR, J. : What is the Future of High Throughput Screening? In: *Journal of Biomolecular Screening* 4 (1999), Juni, Nr. 3, S. 119

[100] MARTINSEN, P. : A time-delayed integration spectrometer. In: *Measurment Science and Technology* 13 (2002), S. 1280–1283

[101] MASTERS, B. R. ; THAER, A. A.: Real-time scanning slit confocal microscopy of the in vivo human cornea. In: *Applied Optics* 33 (1994), Februar, Nr. 4, S. 695–701

[102] MATTHEAKIS, L. C. ; OHLER, L. D.: Seeing the light: calcium imaging in cells for drug discovery. In: *Drug Discovery Today* 1 (2000), Juni, Nr. 1, S. 12–19

[103] MCCORMICK, T. (Hrsg.): *The Essentials of Microbiology*. Research & Education Association, Januar 1998

[104] MCNALLY, J. G. ; KARPOVA, T. ; COOPER, J. ; CONCHELLO, J. A.: Three-Dimensional Imaging by Deconvolution Microscopy. In: *Methods* 19 (1999), S. 373–385

[105] MEYVIS, T. K. L. ; SMEDT, S. C. D. ; OOSTVELDT, P. V. ; DEMEESTER, J. : Fluorescence Recovery After Photobleaching: A Versatile Tool for Mobility and Interaction Measurements in Pharmaceutical Research. In: *Pharmaceutical Research* 16 (1999), August, Nr. 8, S. 1153–1162

[106] MITCHELL, P. : Microfluidics—downsizing large-scale biology. In: *Nature Biotechnology* 19 (2001), August, Nr. 8, S. 717–721

[107] MOORE, K. ; REES, S. : Cell-Based Versus Isolated Target Screening: How Lucky Do You Feel? In: *Journal of Biomolecular Screening* 6 (2001), April, Nr. 2, S. 69–74

[108] MOTRESCU, R. E.: *Analysis of Biological Signals with Multifunctional Bioelectronic Sensor Chips on Living Cells*, Technische Universität München - Heinz-Nixdorf Lehrstuhl für Medizinische Elektronik, Diss., November 2004

[109] MULLER, N. J.: *Bluetooth*. 1. mitp, 2001

[110] MUNCK, S. ; UHL, R. ; HARZ, H. : A ratiometric imaging method for mapping ion flux densities. In: *Cell Calcium* 31 (2002), Januar, Nr. 1, S. 27–35

[111] MUNCK, S. ; BEDNER, P. ; BOTTARO, T. ; HARZ, H. : Spatiotemporal properties of cytoplasmic cyclic AMP gradients can alter the turning behaviour of neuronal growth cones. In: *European Journal of Neuroscience* 19 (2004), Februar, Nr. 4, S. 791

[112] MYERS, S. ; BAKER, A. : Drug discovery—an operating model for a new era. In: *Nature Biotechnology* 19 (2001), August, S. 727–730

[113] NAKANO, A. : Spinning-disk Confocal Microscopy — A Cutting-Edge Tool for Imaging of Membrane Traffic. In: *Cell Structure and Function* 27 (2002), Nr. 5, S. 349–355

[114] NATURE: Lab automation. In: *Nature* 411 (2001), S. 869–870

[115] NEIL, M. A. A. ; JUSKAITIS, R. ; WILSON, T. : Method of obtaining optical sectioning by using structured light in a conventional microscope. In: *Optics Letters* 22 (1997), Dezember, Nr. 24, S. 1905–1907

[116] NEOGY, S. ; WALTER, J. ; SEEBACHER, C. ; UHL, R. : A novel CCD-based dynamic slit confocal microscope. In: *Proceedings of the Focus on Microscopy*, 2006

[117] NETTEN, H. ; VAN VLIET, L. J. ; BODDEKE, F. R. ; DE JONG, P. ; YOUNG, I. T.: A Fast Scanner for Fluorescence Microscopy using a 2-D CCD and Time Delayed Integration. In: *BioImaging* 2 (1994), Nr. 4, S. 184–192

[118] NICOUD, J. : Peripheral interface standards for microprocessors. In: *Proceedings of the IEEE* 64 (1976), Juni, Nr. 6, S. 896–904

[119] NOWAKOWSKI, J. ; HERRENKNECHT, K. : Ultra-high–throughput gene silencing for rapid, economical functional genomics studies. In: *QIAGEN News* 3 (2004), Oktober, Nr. 3, S. 76–78

[120] OAI, E. . *Evotec OAI: From target to IND*. Biotech for Pharma. November 2003

[121] O'LEARY, P. : Machine vision for feedback control in a steel rolling mill. In: *Computers in Industry* 56 (2005), Dezember, Nr. 8-9, S. 997–1004

[122] OTTO, A. M. ; BRISCHWEIN, M. ; MOTRESCU, E. ; WOLF, B. : Analysis of drug action on tumor cell metabolism using electronic sensor chip. In: *Archiv der Pharmazie* 337 (2004), Dezember, Nr. 12, S. 682–686

[123] PAWLEY, J. B. ; PAWLEY, J. B. (Hrsg.): *Handbook of Biological Confocal Microscopy*. New York : Plenum Press, 1995

[124] PERIASAMY, A. : Fluorescence resonance energy transfer microscopy: a mini review. In: *Journal of Biomedical Optics* 6 (2001), Juli, Nr. 3, S. 287–291

[125] PHRMA: Appendix: Detailed Results from the PhRMA Annual Membership Survey. In: *PhRMAannual survey*, 2003

[126] PUCADYIL, T. J. ; CHATTOPADHYAY, A. : Confocal Fluorescence Recovery After Photobleaching of Green Fluorescent Protein in Solution. In: *Journal of Fluorescence* 16 (2006), Januar, Nr. 1, S. 87–94

[127] RABINOWITZ, M. B. ; WETHERILL, G. W. ; KOPPLE, J. D.: Kinetic analysis of lead metabolism in healthy humans. In: *The Journal of Clinical Investigation* 58 (1976), August, Nr. 2, S. 260–270

[128] RESSLER, J. ; GROTHE, H. ; BRISCHWEIN, M. ; WOLF, B. : Sensor-supported 24-well-plates: Multifunctional tools for biological and biomedical HTS applications. In: *Drug Plus International* 3 (2004), September, Nr. 3, S. 19–21

[129] RESSLER, J. ; GROTHE, H. ; MOTRESCU, E. ; WOLF, B. : New concepts for chip-supported multi-well-plates: Realization of a 24- well-plate with integrated impedance-sensors for functional cellular screening applications and automated microscope aided cell-based assays. In: *Proceedings of the 26th Annual International Conference of the IEEE EMBC* 1 (2004), September, Nr. 3, S. 2074–2077

[130] RIBBECK, K. ; GÖRLICH, D. : Kinetic analysis of translocation through nuclear pore complexes. In: *The EMBO Journal* 20 (2001), März, Nr. 6, S. 1320–1330

[131] ROST, F. W. D.: *Fluorescence Microscopy*. 1. Cambridge University Press, Juli 1995

[132] ROY, L. ; DELBOS, M. ; PAILLOLE, N. ; DURAND, V. ; VOISIN, P. : Comparaison de systèmes d'analyse d'images cytologiques en dosimétrie biologique. In: *Radioprotection* 38 (2003), Nr. 3, S. 323–340

[133] SCHNEIDER, G. ; BÖHM, H.-J. : Virtual screening and fast automated docking methods. In: *Drug Discovery Today* 7 (2002), Januar, Nr. 1, S. 64–70

[134] SCHUNCKA, C. ; JOHANNESA, T. ; VARGAB, D. ; LÖRCHA, T. ; PLESCHA, A. : New developments in automated cytogenetic imaging: unattended scoring of dicentric chromosomes, micronuclei, single cell gel electrophoresis, and fluorescence signals. In: *Cytogenetic and Genome Research* 104 (2004), Nr. 1-4, S. 383–389

[135] SEEBACHER, C. ; WALTER, J. ; UHL, R. : New concepts for sensitive multicolor confocal microscopy. In: *Proceedings of the Focus on Microscopy*, 2006

[136] SEGALL, M. : Smart high-throughput screening. In: *Drug Discovery Today* 8 (2003), Februar, Nr. 4, S. 160–161

[137] SEKAR, R. B. ; PERIASAMY, A. : Fluorescence resonance energy transfer (FRET) microscopy imaging of live cell protein localizations. In: *The Journal of Cell Biology* 160 (2003), März, Nr. 5, S. 629–633

[138] SHEETZ, M. P.: *Laser Tweezers in Cell Biology*. Academic Press San Diego, 1998

[139] SHEPPARD, C. ; SHOTTON, D. ; SHEPPARD, C. : *Confocal Laser Scanning Microscopy*. BIOS Scientific Publishers, Februar 1998

[140] SILVERMAN, L. ; CAMPBELL, R. ; BROACH, J. R.: New assay technologies for high-throughput screening. In: *Current Opinion in Chemical Biology* 2 (1998), Juni, Nr. 3, S. 397–403

[141] SIMONE, N. L. ; BONNER, R. F. ; GILLESPIE, J. W. ; EMMERT-BUCK, M. R. ; LIOTTA, L. A.: Laser-capture microdissection: opening the microscopic frontier to molecular analysis. In: *Trends in Genetics* 14 (1998), Juli, Nr. 7, S. 272–276

[142] SISCHKA, A. : *Aubau einer optischen Pinzette*, Universität Bielefeld, Diplomarbeit, 2002

[143] SITTAMPALAM, G. S. ; KAHL, S. D. ; JANZEN, W. P.: High-throughput screening: advances in assay technologies. In: *Current Opinion in Chemical Biology* 1 (1997), Oktober, Nr. 3, S. 384–391

[144] SJAASTAD, M. D. ; ANGRES, B. ; LEWIS, R. S. ; NELSON, W. J.: Feedback regulation of cell-substratum adhesion by integrinmediated intracellular Ca2+ signaling. In: *Proceedings of the National Academy of Sciences of the USA* 91 (1994), Mai, S. 8214–8218

[145] SLIMANE, T. A. ; FONTANGES, P. ; TRUGNAN, G. : GFP, FRAP, FLIP, FRET, PRIM, FLASH ect. . . . New microscopic techniques using new fluorescent probes. An overview. In: *Biology of the Cell* 91 (1999), Juni, Nr. 3, S. 227–228

[146] SMITH-MORROW, J. A.: Proteomics Deals With New Realities. In: *American Biotechnology Laboratory* 23 (2005), August, Nr. 9, S. 10–11

[147] SPERLING, K. : From proteomics to genomics. In: *Electrophoresis* 22 (2001), August, Nr. 14, S. 2835–2837

[148] STAHL, W. : What is the Future of High Throughput Screening? In: *Journal of Biomolecular Screening* 4 (1999), Juni, Nr. 3, S. 117–118

[149] STEPHENS, D. J. ; ALLAN, V. J.: Light Microscopy Techniques for Live Cell Imaging. In: *Science* 300 (2003), April, Nr. 5616, S. 82–86

[150] STOCK, K. ; SAILER, R. ; STRAUSS, W. S. L. ; LYTTEK, M. ; STEINER, R. ; SCHNECKENBURGER, H. : Variable-angle total internal reflection fluorescence microscopy (VA-TIRFM): realization and application of a compact illumination device. In: *Journal of Microscopy* 211 (2003), Juli, Nr. 1, S. 19–29

[151] STRAUB, M. ; LODEMANN, P. ; HOLROYD, P. ; JAHN, R. ; HELL, S. W.: Live cell imaging by multifocal multiphoton microscopy. In: *European Journal of Cell Biology* 79 (2000), Oktober, S. 726–734

[152] SU, J. ; HUANG, C. ; TARNG, Y. : An automated flank wear measurement of microdrills using machine vision. In: *Journal of Materials Processing Technology* 180 (2006), Dezember, Nr. 1-3, S. 328–335

[153] SU, Z. ; TIAN, G. Y. ; GAO, C. : A machine vision system for on-line removal of contaminants in wool. In: *Mechatronics* 16 (2006), Juni, Nr. 5, S. 243–247

[154] SUN, D. ; COHEN, S. ; MANI, N. ; MURPHY, C. ; ROTHSTEIN, D. : A pathway-specific cell based screening system to detect bacterial cell wall inhibitors. In: *The Journal of antibiotics* 55 (2002), März, Nr. 3, S. 279–287

[155] SUNDBERG, S. A.: High-throughput and ultra-high-throughput screening: solution- and cell-based approaches. In: *Current Opinion in Biotechnology* 11 (2000), Februar, Nr. 1, S. 47–53

[156] SZÖLLOSI, J. ; DAMJANOVICH, S. ; MÁTYUS, L. : Application of fluorescence resonance energy transfer in the clinical laboratory: Routine and research. In: *Cytometry (Communications in Clinical Cytometry)* 34 (1998), August, Nr. 4, S. 159–179

[157] TAMMELA, P. : *Screening Methods for the Evaluation of Biological Activity in Drug Discovery*, University of Helsinki, Diss., 2004

[158] TAO, Y. ; HEINEMANN, P. H. ; VARGHESE, Z. ; MORROW, C. T. ; SOMMER, H. J.: Machine vision for color inspection of potatoes and apples. In: *Transactions of the ASAE* 38 (1995), Nr. 5, S. 1555–1561

[159] TERSKIKH, A. ; FRADKOV, A. ; ERMAKOVA, G. ; ZARAISKY, A. ; TAN, P. ; KAJAVA, A. V. ; ZHAO, X. ; LUKYANOV, S. ; MATZ, M. ; KIM, S. ; WEISSMAN, I. ; SIEBERT, P. : "Fluorescent Timer": Protein That Changes Color with Time. In: *Science* 290 (2000), November, Nr. 5496, S. 1585–1588

[160] THOMAS, D. ; LIPP, P. ; TOVEYA, S. C. ; BERRIDGEA, M. J. ; LIC, W. ; TSIEN, R. Y. ; BOOTMAN, M. D.: Microscopic properties of elementary Ca2+ release sites in non-excitable cells. In: *Current Biology* 10 (2000), Januar, Nr. 1, S. 8–15

[161] TOTOK-STORB, B. : Cellular Interactions. In: *The Journal of The American Society of Hematology* 72 (1988), August, Nr. 2, S. 373–385

[162] TRUGNAN, G. ; FONTANGES, P. ; DELAUTIER, D. ; AIT-SLIMANE, T. : FRAP, FLIP, FRET, BRET, FLIM, PRIM... new techniques for a colourful life. In: *Médecine sciences (Paris)* 20 (2004), November, S. 1027–1034

[163] TSIEN, R. Y.: The Green Fluorescent Protein. In: *Annual Review of Biochemistry* 67 (1998), Juli, S. 509–544

[164] TWYMAN, R. M.: *Principles of proteomics*. BIOS Scientific Publishers New York, Oktober 2004

[165] Kap. Arc lamps and monochromators for fluorescence microscopy In: UHL, R. : *Imaging Neurons: A Laboratory Manual*. Cold Spring Harbor Laboratory Press, Juli 2000, S. 2.1–2.8

[166] UHL, R. ; HARZ, H. ; WALTER, J. ; SCHULZE, S. : Mikroskopische Objekte in verschiedenen Dimensionen. In: *LaborPraxis* 1 (2005), January/February, S. x2–x4

[167] UHL, R. ; WALTER, J. ; SEEBACHER, C. ; HOCHE, R. : Imaging microscopy redifined. In: *Proceedings of the Focus on Microscopy*, 2006

[168] VERKHUSHA, V. V. ; LUKYANOV, K. A.: The molecular properties and applications of Anthozoa fluorescent proteins and chromoproteins. In: *Nature Biotechnology* 22 (2004), März, Nr. 3, S. 289–296

[169] WALTERS, W. ; M.T.STAHL ; MURCKO, M. : Virtual screening-an overview. In: *Drug Discovery Today* 3 (1998), April, Nr. 4, S. 160–178

[170] WANGA, M. ; YAO, S. ; MADOU, M. : A long-term stable iridium oxide pH electrode. In: *Sensors and Actuators B: Chemical* 81 (2002), Januar, Nr. 2-3, S. 313–315

[171] WIEST, J. ; STADTHAGEN, T. ; SCHMIDHUBER, M. ; BRISCHWEIN, M. ; RESSLER, J. ; RAEDER, U. ; GROTHE, H. ; MELZER, A. ; WOLF, B. : Intelligent Mobile Lab for Metabolics in Environmental Monitoring. In: *Analytical Letters* 39 (2006), August, Nr. 8, S. 1759–1771

[172] WIEST, J. ; BRISCHWEIN, M. ; GROTHE, H. ; ANELIE M. OTTO, B. W.: Planar Microsensors for measurement of cellular respiration. In: *SENSOR 2005 Proceedings II* 2 (2005), S. 249–254

[173] WIEST, J. ; BRISCHWEIN, M. ; GROTHE, H. ; WOLF, B. : Miniaturisierbare, biokompatible Gelöst-Sauerstoff-Sensoren. In: *Chemie Ingenieur Technik 2005* 77 (2005), Nr. 8, S. 1115–1116

[174] WIEST, J. ; BRISCHWEIN, M. ; RESSLER, J. ; OTTO, A. M. ; GROTHE, H. ; WOLF, B. : Cellular Assays with Multiparametric Bioelectronic Sensor Chips. In: *CHIMIA International Journal for Chemistry* 59 (2005), Mai, Nr. 5, S. 243–246

[175] WILLIAMS, M. : Systems and integrative biology as alternative guises for pharmacology: Prime time for an iPharm concept? In: *Biochemical Pharmacology* 71 (2005), Dezember, Nr. 12, S. 1707–1716

[176] WILSON, T. ; MASTERS, B. R.: Confocal microscopy: Introduction to the feature issue. In: *Applied Optics* 33 (1994), Nr. 4, S. 565

[177] WINKLER, T. ; KETTLING, U. ; KOLTERMANN, A. ; EIGEN, M. : Confocal fluorescence coincidence analysis: An approach to ultra high-throughput screening. In: *Proceedings of the National Academy of Sciences of the USA* 96 (1999), Februar, Nr. 4, S. 1375–1378

[178] Kap. Lab-on-a-chip Systems for Cellular Assays In: WOLF, B. ; BRISCHWEIN, M. ; GROTHE, H. ; STEPPER, C. ; RESSLER, J. ; WEYH, T. : *BioMEMS*. Springer-Verlag, 2006, S. 269–308

[179] WOLF, B. ; BRISCHWEIN, M. ; OTTO, A. ; GROTHE, H. : Chip statt Maus: Zur Bedeutung multiparametrischer biohybrider Bauelemente in Toxikologie und Pharmascreening. In: *Technisches Messen* 71 (2003), Dezember, Nr. 12, S. 553–556

[180] WOLF, B. ; GEISLER, T. ; LOB, V. ; BRISCHWEIN, M. ; RESSLER, J. ; WIEST, J. : Chip statt Maus: Mikrosensorarrays zu Chemikalienprüfung. In: *Nachrichten aus der Chemie* 54 (2006), Februar, S. 115–120

[181] WOLF, D. E.: Designing, building, and using a fluorescence recovery after photobleaching instrument. In: *Methods in Cell Biology* 30 (1989), S. 271–306

[182] YAO, S. ; WANG, M. ; MADOU, M. : A pH Electrode Based on Melt-Oxidized Iridium Oxide. In: *Journal of the Electrochemical Society* 148 (2001), S. H29–H36

[183] ZIMMERMANN, W. ; SCHMIDGALL, R. : *Bussysteme in der Fahrzeugtechnik. Protokolle und Standards*. Vieweg, 2006

Publikationsliste

Veröffentlichungen:

1. Thomas Geisler, Bernhard Wolf, Rainer Uhl,
   "Modular Real-Time Control Platform for Cell-Based High-Content and High-Throughput Screening"
   BMT Proceedings
   ISSN 0939-4990, September 2006

2. Thomas Geisler, Volker Lob, Bernhard Wolf, Rainer Uhl,
   "Control-Architecture for Cell-Based High-Content and High Throughput Screening"
   IFMBE Proceedings
   Volume 14 (ISSN 1727-1983), pp. 2152 - 2155, August 2006

3. Volker Lob, Thomas Geisler, Bernhard Wolf, Rainer Uhl,
   "Multiparametric automated screening system based on a 24-Well-Microplate for living cells and tissues"
   IFMBE Proceedings
   Volume 14 (ISSN 1727-1983), pp. 557 - 560, August 2006

4. Thomas Geisler, Johann Ressler, Hartmann Harz, Bernhard Wolf, Rainer Uhl,
   "Automated Multiparametric Platform for High-Content and High-Throughput Analytical Screening on Living Cells"
   IEEE Transactions on Automation Science and Engineering
   Volume 3, Issue 2 (ISSN 1545-5955), pp. 169 - 176, April 2006

5. Bernhard Wolf, Thomas Geisler, Volker Lob, Martin Brischwein, Johann Ressler, Joachim Wiest,
   "Chip statt Maus: Mikrosensorarrays zu Chemikalienprüfung"
   Nachrichten aus der Chemie
   Volume 54 (ISSN 1439-9598), pp. 115 - 120, Februar 2006

6. Thomas Geisler, Bernhard Wolf, Rainer Uhl,
   "Modular Control-System for Automated Analytical Bio Imaging and Integration into High-Content and High-Throughput Screening Setups"
   Biomedizinische Technik
   Band 50, Ergänzungsband 1 (ISSN 0939-4990), Teil 1, pp. 1384-1385, September 2005

7. Thomas Geisler, Theodore W. Manikas,
   "Autonomous Robot Navigation System Using a Novel Value Encoded Genetic Algorithm"
   Proceedings on the 45th IEEE International Midwest Symposium on Circuits and Systems
   Volume 3 (ISBN 0-7803-7523-8), pp. 45 - 48, August 2002

Fachvorträge:

1. *BMT 2006*
   Gemeinsame Jahrestagung der Deutschen, Österreichischen und Schweizerischen Gesellschaften für Biomedizinische Technik 2006
   06. - 09. September 2006; ETH Zürich, Schweiz

   Thomas Geisler, Bernhard Wolf, Rainer Uhl,
   "Modular Real-Time Control Platform for Cell-Based High-Content and High-Throughput Screening"
   Session: Sensorik
   Zeit: Freitag, 08. September 2006, 16:00 - 18:00

2. *World Congress on Medical Physics and Biomedical Engineering 2006*
   Internationale Konferenz und Ausstellung
   27. August - 01. September 2006; Konferenzzentrum COEX Seoul, Korea

   Thomas Geisler, Volker Lob, Bernhard Wolf, Rainer Uhl,
   "Control-Architecture for Cell-Based High-Content and High Throughput Screening"
   Track: Image Processing, Analysis, and Visualization (Track 14)
   Session: In-vitro Imaging (Session 2)
   Zeit: Montag, 28. August 2006, 14:40 - 16:10

3. *MipTec 2006*
   Internationale Konferenz und Ausstellung
   7-10 Mai 2006; Konferenzzentrum Basel, Schweiz

   Thomas Geisler, Volker Lob, Bernhard Wolf, Rainer Uhl,
   "Microscope-based High-Throughput and High-Content Screening with the "Intelligent Microplate Reader""
   Session: Drug Discovery Technologies V: Image Analysis Applications
   Zeit: Donnerstag, 11. Mai 2006, 11:30 - 12:00

Wissenschaftliche Poster:

1. *MipTec 2006*
   Internationale Konferenz und Ausstellung
   7-10 Mai 2006; Konferenzzentrum Basel, Schweiz

   Thomas Geisler, Bernhard Wolf, Rainer Uhl,
   "Control-Architecture for Microscope-based High Content High Throughput Screening"
   MipTec 2006, Tagungsband, Nr. P130, 2006

2. *MipTec 2006*
   Internationale Konferenz und Ausstellung
   7-10 Mai 2006; Konferenzzentrum Basel, Schweiz

   Volker Lob, Thomas Geisler, Johann Ressler, Helmut Grothe, Martin Brischwein, Bernhard Wolf,
   "Multiparametric Automated Screening System Based on a 24-Well-Microplate for Living Cells and Tissues"
   MipTec 2006, Tagungsband, Nr. P72, 2006

3. *BMT 2005*
   39. Jährlicher Kongress der Deutschen Gesellschaft für Biomedizinische Technik
   14-17 September 2005; Konferenzzentrum Nürnberg, Deutschland

   Thomas Geisler, Bernhard Wolf, Rainer Uhl,
   "Modular Control-System for Automated Analytical Bio Imaging and Integration into High-Content and High-Throughput Screening Setups"
   Biomedizinische Technik, Band 50, Ergänzungsband 1, Teil 1, pp. 1384-1385, 2005

4. *MipTec 2005*
   Internationale Konferenz und Ausstellung
   09-12 Mai 2005; Konferenzzentrum Basel, Schweiz

   Thomas Geisler, Rainer Uhl,
   "Imaging Control Platform for Automated High-Throughput and High-Content Screening under the Microscope"
   MipTec 2005, Tagungsband, p. 117, 2005

Money never made a man happy yet, nor will it. There is nothing in its nature to produce happiness. The more a man has, the more he wants. Instead of its filling a vacuum, it makes one. If it satisfies one want, it doubles and trebles that want another way.

Better is little with the fear of the Lord, than great treasure, and trouble therewith.

– Benjamin Franklin –