Nr.64a /Sept.1999

ISSN 1022-161

PCNEWS

Mikro 2

educ@tion

ADIM WIEN GRAZ · CC · CE RETZ · IT · MCCA · OeCAC · PCC-S · PCCTC-I

Sonderheft SIEMENS

Mikrocontroller Starterkits & EXBO

in Forschung Ausbildung & Industrie

Infineon technologies

SEPTEMBER 1999

SONDERHEFT MIKRO 2

PCNEWS 64a

9771022161000 64

ATS 60,50
€ 4,4

Verlagspostamt 1100 Wien, P.b.b. 8 8 8 5 0 W 8 6 U

SIEMENS

C166 Family

Infineon technologies

Der SIEMENS Halbleiter-Bereich heißt jetzt INFINEON

Members of C166 Family	Type	Max CPU Clock	Oscillator	Instruction Cycle Time	ROM/Flash	RAM (Including XRAM)	Linear Address Space for Code & Data	I/O Lines	ADC Inputs Resolution	Timers/Counters (16-Bit)	Capture Compare Unit (Channels)	PWM (Outputs)	Interrupt Vectors/Levels	Serial I/O	Real Time Clock	CAN Interface 2.0B active	Hardware Power Down	Watchdog Timer	Osc Watchdog	On-Chip Bootstrap Loader	Packaging
C161	C161V-L16M	16 MHz	prescaler/ direct input	125 ns	–	1 KB	4 MB	63	–	3	–	–	14/16	USART + SSC	–	–	–	✓	–	–	P-MQFP-80
	C161K-L16M												20/16				✓			✓	
	C161O-L16M					2 KB															
	C161RI-L16M					3 KB	8 MB	76	4/8 Bit	5			27/16	USART + SSC + I²C	✓		yes + Power saving				P-MQFP-100
	C161RI-L16F																				P-TQFP-100
C163	C163-LF	20 MHz	PLL/ prescaler direct input	100 ns	–	1 KB	16 MB	77	–	5	–	–	20/16	USART + SSP	–	–	✓	✓	✓	–	P-TQFP-100
	C163-L25F	25 MHz		80 ns																	
	C163-16F25F*	25 MHz		80 ns	128 KB Flash																
C164	C164CI-8EM*	20 MHz	PLL/presc. direct input	100 ns	64 K OTP	2 KB	4 MB	69	8/10 Bit	5	8	6	32/16	USART + SSC	✓	✓	yes + Powers.	✓	✓	✓	P-MQFP-80
	C164CI-8RM*	25 MHz		80 ns	64 K ROM	3 KB															
C165	C165-LM	20 MHz	prescaler direct input	100 ns	–	2 KB	16 MB	77	–	5	–	–	28/16	USART + SSC	–	–	✓	✓	–	✓	P-MQFP-100
	C165-L25M	25 MHz		80 ns																	
	C165-RM	20 MHz		100 ns	4 KB ROM																
	C165-LF	20 MHz		100 ns	–																P-TQFP-100
	C165-L25F	25 MHz		80 ns																	
C166	SAB80C166-M	20 MHz	prescaler	100 ns	–	1 KB	256 KB	76	10/ 10 Bit	7	16	–	32/16	2 x USART	–	–	✓	✓	–	✓	P-MQFP-100
	SAB80C166-M25	25 MHz		80 ns																	
	SAB83C166-5M	20 MHz		100 ns	32 KB ROM																
	SAB83C166-5M25	25 MHz		80 ns																	
	SAB88C166-5M	20 MHz		100 ns	32 KB Flash																
	SAB80C166W-M	20 MHz	direct input	100 ns	–																
	SAB83C166W-5M				32 KB ROM																
	SAB88C166W-5M				32 KB Flash																
C167	C167-LM	20 MHz	prescaler	100 ns	–	2 KB	16 MB	111	16/ 10 Bit	9	32	4	66/16	USART + SSC	–	–	✓	✓	–	✓	P-MQFP-144
	C167S-4RM		PLL/ direct input		32 KB ROM														✓		
	C167SR-LM				–	4 KB										✓			–		
	C167CR-LM																				
	C167CR-L25M	25 MHz		80 ns																	
	C167CR-4RM	20 MHz	PLL/presc. direct input	100 ns	32 KB ROM														✓		
	C167CR-16RM		PLL/ direct input		128 KB ROM														–		
	C167CR-16FM*				128 KB Flash																

* in preparation

Published by Semiconductor Group

Siemens Aktiengesellschaft

Ordering No. B 158-H6957-G5-X-7600
Printed in Germany
PS 11985. (8541)

Inhalt

Webversion

http://pcnews.at/ins/pcn/64a/~64a.htm

LIESMICH

KOCHREZEPTE

SOFTWARE

HARDWARE

PROJEKTE

GRUNDLAGEN

UNTERHALTSAMES

Mikrocontroller in PCNEWS

PCNEWS	Seite	Thema
38	43	Mikrocontroller
38	63	Buchliste
51	90	Karusell-1
52	46	Karusell-2
52	76	OOP - vom Assembler über C zu C++
54	117	Unabhängigkeitserklärung eines Mikrocontrollers
57	86	DAVE & Starterkits
58	114	Starterkit
59	93	C161-Starterkit
59a		Sonderausgabe Mikro-1
60	97	Mikrocontroller im Internet und Literatur
60	102	Erfolgreich starten
61	97	Monitor für Mikrocontroller
61	104	EXBO-1
62	102	Flash Tools
62	106	EXBO-2

Impressum

Diese Sonderausgabe der PCNEWS wird von SIEMENS herausgegeben und im PCNEWS-Eigenverlag hergestellt. Die Auflage ist 11000.

Anschrift

Siemens AG Österreich
Bauelemente und Sondertechnik
Erdberger Lände 26, A-1031 Wien
E-Mail: *wilhelm.brezovits@siemens.at*

PCNEWS-Eigenverlag
Siccardsburggasse 4/1/22, A-1100 Wien
Tel.: 01-604 5070, DW-2
E-Mail: *pcnews@pcnews.at*
WWW: *http://pcnews.at/.*

Autorinnen und Autoren

Berger Christian 47

Karikaturist und Comiczeichner

E✉ *karicartoons@magnet.at*

Brezovits Wilhelm Ing. Jg.1968 4,6,21,66,68

Produktspezialist für Mikrocontroller

Firma Siemens AG

Absolvent HTL-Mödling, E5b, 1987

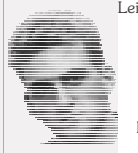
Interessen C, C++ und µC-C/C++

Hobbies Blockflöte, Rad

Privates Verheiratet, 3 Söhne

E✉ *Wilhelm.Brezovits@siemens.at*

Gausterer Robert Ing. Jg.1967 62



Leitung Marketing & Vertrieb

Firma A&R TECH

Werdegang HTL, Burisch Elektronik Bauteile Vertrieb, Produktmanagement, Bereichsleitung, Marketing & Verkaufsmanagementlehrgangs

Hobbies Mountainbiken, Laufen, Gesang, Motorradfahrer, Billard

Privates verheiratet

E✉ *rg.artech@aon.at*

Hintenaus Peter Dipl.-Ing.Dr. 60

FH-Lehrer für Industrielle Elektronik in Kapfenberg

Hochschule Fachhochschule Kapfenberg

Werdegang Promotion im Bereich Symbolic Computation am RISC Linz; Assistant Professor für Computer Science an der Kent State University, Ohio (USA)

Absolvent Uni Linz Informatik

Hobbies segelfliegen, radfahren, Uhren

Privates ledig

E✉ *Peter.Hintenaus@fh-joanneum.at*

Kliegelhöfer Peter Dipl.-Ing. (FH) 34

Application Engineering Microcontrollers

Firma Infineon AG

Interessen Elektronik, Computer

Hobbies Musik, Fahrrad fahren, Wandern, Reisen

E✉ *peter.kliegelhoefer@infineon.com*

Kral Anton Ing. Jg.1968 54

Techniker

Hochschule UNI Linz, Inst. f. praktische Informatik

E✉ *kral@ssw.uni-linz.ac.at*

🌐 *http://www.ssw.uni-linz.ac.at/*

Krammer Hermann Dr. Jg.1951 56

Lehrer für Elektronik

Schule HTBLA Braunau

Club PCCTGM

Hobbies Klavier, Querflöte

Privates verheiratet, 2 Kinder

E✉ *h.krammer@eduhi.at*

Krause Werner Mag. Jg.1955 U1,U4

Lehrer für Bildnerische Erziehung

Schule GRG Wien 23 Alterlaa

Absolvent Hochschule f. Angewandte Kunst

Interessen CorelDraw, PhotoShop, Painter

Hobbies Fotografieren, Modellbahnbau, Coverbilder für PCNEWS

Privates verheiratet, 2 Kinder

E✉ *w.krause@chello.at*

Maresch Helfrid a.o.Univ.Prof. Dipl.-Ing.Dr. 60,61

Studiengangsleiter der „Industriellen Elektronik“

Hochschule Fachhochschule Kapfenberg

Werdegang Studium der Technischen Physik, Promotion und Habilitation im Bereich der Medizinschen Informatik

Absolvent TU Graz

Hobbies laufen, fliegen (CPL)

Privates verheiratet, 5 Kinder

E✉ *Helfrid.Maresch@fh-joanneum.at*

Perschl Christian Jg.1975 34,41,72

Student der Informatik

Schule TU-Wien

Club PCCTGM

Absolvent TGM-N93B

Interessen Computer

Hobbies Musik

E✉ *e9327470@stud1.tuwien.ac.at*

🌐 *http://stud1.tuwien.ac.at/~e9327470*

Prandstetter Gottfried Jg.1951 40

Lehrer für fachpraktischen Unterricht an der HTL-Leonding seit 1986; Entwicklung und Kleinstserien von Maschinensteuerungen und Messtechnik.

Schule HTL Leonding

Interessen Entwicklung von Hard- und Software

E✉ *g.prandstetter@eduhi.at*

Radl Reinhard Ing. Jg.1954 62

Technischer Leiter

Firma A&R TECH

Werdegang Kiepe Electric Hardwareentwicklung, Siemens SVT Hard/Softwareentwicklung, igm Robotersysteme AG

Absolvent HTL Mödling 1974

Hobbies Amateurfunker (Call OE3RDW), Motorradfahrer, Jukeboxen

Privates verheiratet, 1 Sohn

E✉ *rr.artech@aon.at*

Resel Manfred Ing. 53,55

Lehrer für Werkstättenlabor und PRT

Schule HTBLA-Hollabrunn, Regelungstechnik

E✉ *manfred.resel@r.htl-hl.ac.at*

Schultes Renate Jg.1957 63

Dozentin für Mikroelektronik

Firma MicroConsult

E✉ *r.schultes@microconsult.de*

Thieme Andreas Dipl.-Ing. Jg.1969 57

Engineering, Softwareentwicklung

Firma Windtec

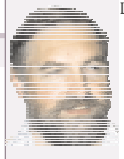
Absolvent TU Chemnitz, D, 1995

Interessen Alternative Energien, Umrichtertechnik

Privates verheiratet

E✉ *Andreas.Thieme@mail.windtec.com*

Tomaselli Werner Mag. 59



Lehrer für Mathematik, Physik und Informatik

Schule HTL Bregenz

E✉ *bzkaud@htlb.vol.at*

Waldner Walter Dr. 8,26,44,49

Lehrer für EDV und Technische Informatik

Schule HTL Klagenfurt Mössingerstraße

Club PCCTGM

E✉ *walter.waldner@telekabel.at*

🌐 *http://www.htblmo-klu.ac.at/lernen/*

Inserenten

vivi

DIE SCHNELLSTEN!

An unserem leistungsstarken Housing-Knoten “VIVI” steht Ihr Server oder Modem an einem der schnellsten Punkte im Österreichischen Internet.

ATnet Housing bietet folgende Möglichkeite :

- Internationale Anbindung ans Internet über mehrere unterschiedliche Netz-Anbindungen garantiert schnelle Verbindungen und hohe Ausfallssicherheit.
- Schnellste nationale Anbindung ans Österreichische Internet durch Verkehraustausch-Abkommen mit praktisch allen Österreichischen Internet-Service-Anbietern.
- Direkte Anbindung an unseren leistungsfähigen Backbone
- Sie können mit Ihrem Server z. B. einen Web-Server betreiben, Webspace vermieten; einen eigenen Mail-Server, Chat-Server, FTP-Server, Audio-Server, Video-Server machen.
- Die Anbindung erfolgt an einen der schnellsten Punkte im Österreichischen Internet - mit mehreren Hundert Mbit Anbindung ans Internet.
- Zusatzservices wie Domain-Name-Services, Backup-Mailserver ("Secondary MX"), Backup-Nameserver ("Secondary NS"), Mail-Relaying (SMTP-Forwarding) sind möglich und inkludiert.
- Überwachung Ihres Servers 24 Stunden am Tag, mit Mail/SMS-Benachrichtung bei Ausfall ("Watchdog")
- at-net stellt Ihnen International gültige IP Adressen in ausreichendem Rahmen zur Verfuegung (Sie bekommen von uns die Daten, die für die Netzwerkeinrichtung notwendig sind.)
- Ein Wählleitungszugang zur Wartung ist auf Wunsch inkludiert.
- Sie können auf Ihrem Server beliebige Software und Hardware verwenden.
- Ihr Server steht in einem eigens adaptierten Serverhousing-Zentrum mit Hochgeschwindigkeitsanbindung, hohen Sicherheitsstandards, Zutrittskontrolle, unterbrechungsfreier Stromversorgung, Überspannungsschutz, Filter gegen Stromschwankungen, Brandmeldeanlage, Wassermeldeanlage, Voll-Klimatisierung und ist damit zu den bestmöglichen Bedindungen mit dem Internet verbunden.
- Gegen Voranmeldung haben Sie fuer geringfügige Arbeiten innerhalb unserer Geschäftszeiten (werktags, 09.00-18.00 Uhr) Zugang zu Ihrem Server
- Zusatzdienste wie Telefonleitungen, etc. sind möglich
- Serverstandort / Standort fuer Standleitungsanbindung: 1010 Wien, Landesgerichtsstrasse

http://www.at-net.at/

Die einmaligen Kosten sind abhängig von einem: 10 Mbit/s oder 100 Mbit/s Anschluss.
Der 10 Mbit/s Anschluss [10base] kostet ATS 12.000,- (EUR 872.07).
Der 100 Mbit/s Anschluss [100base] kostet ATS 36.000,-- (EUR 2616.22).

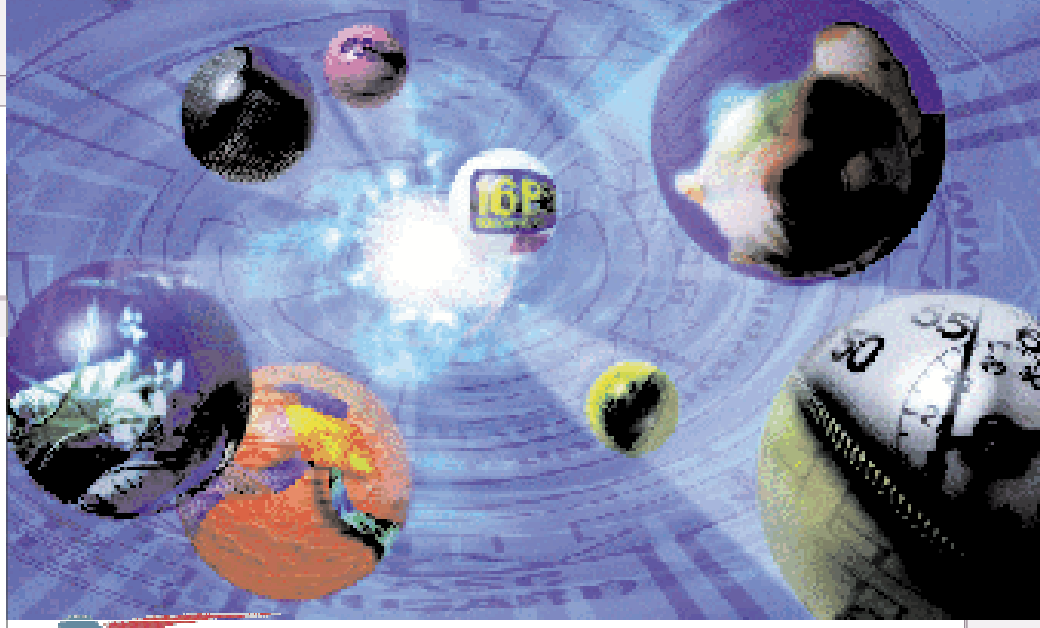
Die laufenden Kosten sind abhängig von: Gehäuseart und übertragener Datenmenge.
Ein Mini-Tower kostet ATS 2.400,-- (EUR 174.41) [Rechner + Messpauschale].
Die Kosten für Datenmenge sind ATS 500,-- (EUR 36.34) pro GB.

Kontaktieren Sie uns unter:

ATnet
Alxingerstraße 37/1a A-1100 Wien
Tel: 01 605 52-87 Fax: 01 605 52-88
email: info@atnet.at

ATnet
INTERNET WORKS

Liebe Leser!



Sie halten soeben die zweite Sonderausgabe der PCNEWS zum Thema INFINEON Mikrocontroller in Ihren Händen.

Diese Ausgabe bietet Ihnen die Möglichkeit, Ihre Erfahrungen mit unseren Produkten sowie die realisierten Anwendungen und eventuellen Konzeptüberlegungen einem weiten Anwenderkreis vorzustellen.

Dieses Heft soll Ihnen aber auch einen Überblick über die Aktivitäten von Universitäten, Fachhochschulen, HTL's, Konsulenten, Ingenieurbüros, Entwicklern, Firmen und Privatpersonen in ganz Österreich rund um die INFINEON Mikrocontrollerfamilien geben.

Schwerpunkt

Der Schwerpunkt dieser Ausgabe ist eine pädagogisch aufbereitete Artikelserie zur Inbetriebnahme des INFINEON 16 bit C167CR-Mikrocontrollerstarterkits. Es wäre für uns eine besondere Freude, wenn dieser Sonderdruck auch als ergänzender Unterrichtsbehelf Verwendung finden kann und als "Kochrezept" zu einem Erfolgserlebnis für den Studierenden führt.

Infineon technologies

Einladung

Da wir bereits an der nächsten Ausgabe arbeiten, möchten wir Sie einladen, uns einen Beitrag über Ihre Applikationen oder aber auch nur grundsätzliche Überlegungen zur Verfügung zu stellen.

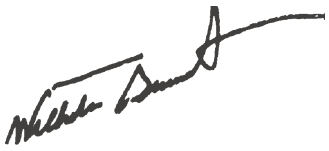
Interessant wäre daher eine Seite mit Bild und Text

- der Personen, die sich mit dem INFINEON μC beschäftigen und den gemachten Erfahrungen,
- des Gerätes, wo der μC eingebaut ist und welche Aufgaben er darin übernimmt.

Herzlichen Dank

Bei den Autoren der Beiträge dieser Ausgabe möchten wir uns herzlich für die Mühe bedanken und würden uns freuen, wenn wir auch in Zukunft weitere Informationen bekommen.

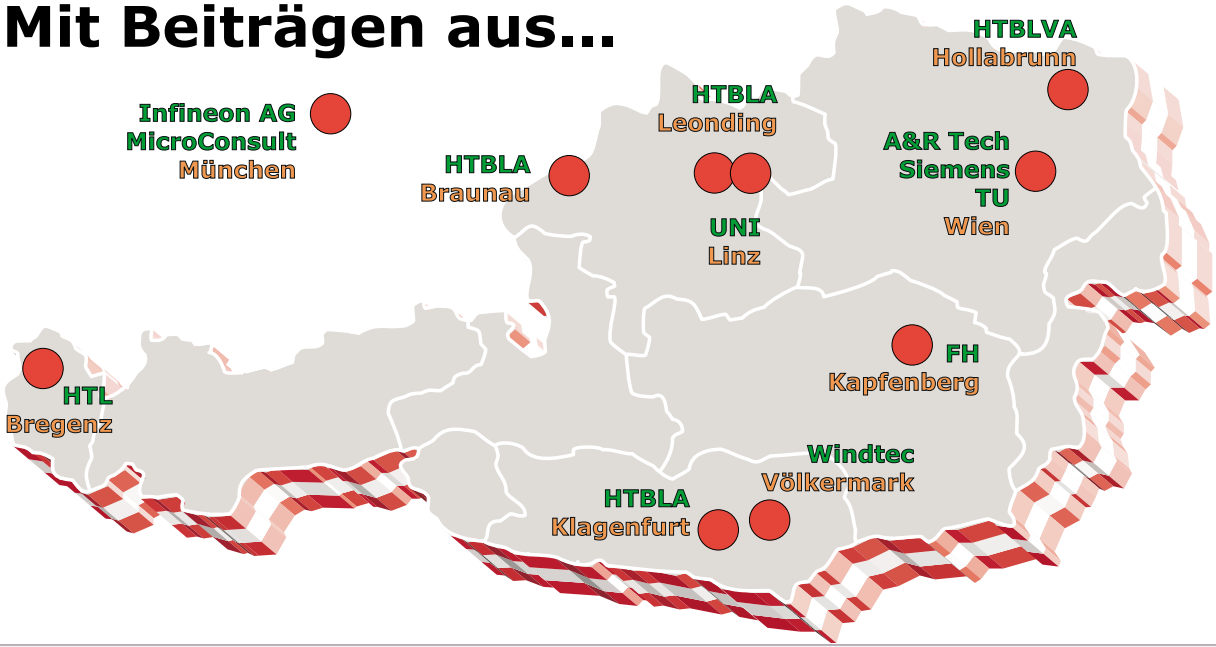
Ihr



Wilhelm Brezovits

Siemens AG Österreich
Bauelemente und Sondertechnik
Erdbergerlände 26
A-1031 Wien

Mit Beiträgen aus...



Weblinks

Wilhelm Brezovits

Mikrocontroller		
Infineon Mikrocontroller Homepage	Startseite im Internet für die (Siemens)/Infineon Mikrocontroller	http://www.infineon.com/microcontrollers/
DAvE (Digitaler Applikationsingenieur)	Updates für die DAvE-CD	http://www.infineon.com/DAvE.html
Development Tools Partners Magazine	Kostenloses Abo der Toolpartnerzeitschrift CONTACT	http://www.spacetools.com/
TriCore (32 bit µC,µP,DSP)	Symbiose aus Mikroprozessor (µP), Mikrocontroller (µC) und DSP (Digital Signal Prozessor).	http://www.tri-core.com/
Mikrocontroller Universität	Viele wertvolle Online-Kurse	http://www.mfuniversity.com/
Compiler-Hersteller		
Keil		http://www.keil.com/
Tasking		http://www.tasking.com/products/80C166/
Embedded C++		http://www.caravan.net/ec2plus/
Universitäten, FH's und HTL's		
Klagenfurter Unterrichtsserver	Hier findet man alles zur Inbetriebnahme des C167CR Starterkits u.v.m.	http://www.htblmo-klu.ac.at/lernen/siemens/index.htm
TU Wien Institut für Computertechnik	Hier findet man ein Online Skriptum für den C167CR	http://mc.ict.tuwien.ac.at/
TU Graz Institut für Elektronik	Hier findet man C167CR Kerne (Minimodule) und Kernöl	http://www-ife.tu-graz.ac.at/Elektronik/Roehrer/Grurf/index.htm
Uni Linz Institut für prakt. Informatik	div. Mikrocontrollerprojekte	http://ssw.uni-linz.ac.at/General/Staff/AK/
Uni Salzburg, Institut für Computerwissenschaften	C167CR Fußballroboter	http://www.cosy.sbg.ac.at/~robolab/
HTL Braunau	div. Mikrocontrollerprojekte	http://www.asn-linz.ac.at/schule/htlbraunau/lehrer/krammer/index.htm
Schule für Mikroelektronik		
MicroConsult	Hier findet man Kurse für Ausbildung und Weiterbildung	http://www.microconsult.de/
Bücher		
Otmar Feger	Hardware + Software Verlag	http://www.otmar-feger.de/
Starterkits		
Starterkits	Bestellen von Siemens/Infineon Mikrocontrollerstarterkits	http://www.mtm.at/starterkit.htm
MINIMON	Dieses Tool muss man haben!	http://stud1.tuwien.ac.at/~e9327470/minimon/minimon.htm
Diverses		
Hitex, C166 Applikationen	Ein Blick lohnt sich!	http://www.hitex.demon.co.uk/c166/miscdocs.htm
Hitex, Insider's Guide To Planning 166 Family Designs	Ein Blick lohnt sich!	http://www.hitex.demon.co.uk/book166/166des19-a.html
Willert Software Tools	Expertenforum	http://www.willert.de/forum/
MTM SYSTEME	Toolpartner in Österreich	http://www.mtm.at/
WALTER REKIRSCH	Toolpartner in Österreich	http://www.rekirsch.com/

VERITAS

Datensicherung mit einem verläßlichen Werkzeug - damit Ihre Daten auch am Tag X zur Verfügung stehen!

Wie sicher sind Ihre Daten am PC?

Viele Anwender standen schon vor dem Problem, aus heiterem Himmel den Zugang zu ihrer Festplatte zu verlieren - dem einzigen Medium, auf dem sich ihre Daten befanden. Es kommt einfach immer wieder vor, daß Festplatten ohne Vorankündigung defekt werden und es nicht mehr möglich ist, auf die gespeicherten Informationen zuzugreifen.

Datenverlust rechtzeitig vorbeugen

Führen Sie mit einer geeigneten, komfortablen Software regelmäßig Sicherungsläufe durch, um stets gewappnet zu sein, wenn ein Störfall passiert. Backup Exec von Veritas ist komfortabel und trotz seiner hohen Leistungsfähigkeit einfach in der Bedienung. Automatische Sicherungsläufe - auch nach komplexer Definition - können zu beliebigen oder festgelegten Zeiten durchgeführt werden.

Ihre Sicherung ist derzeit zu kompliziert oder nicht zufriedenstellend?

Nützen Sie Ihre alten Sicherungsbänder von Windows NT oder von ArcServe - Sie können diese in Backup Exec einfach weiterverwenden!

Einfache Sicherungsfunktionen sind im Standard-Sicherungsprogramm von Windows NT vorhanden, die Rücksicherung ist aber nur in einfachster Form möglich, da man auf den Bändern nicht richtig suchen kann, u.a.. Mit Backup Exec hingegen haben Sie vollen Überblick über den Zustand Ihrer Bänder, diese werden mit genauen Angaben verwaltet, z.B. darüber, wieviel Daten gesichert wurden, wieviele Lesefehler aufgetreten sind, wieviel Platz noch auf dem Band ist, etc., etc. Unzuverlässige Bänder können erkannt und ausgetauscht werden.

Backup Exec erkennt auch Viren

Eine wichtige Funktion von Backup Exec ist auch, daß die Software in der Lage ist, Viren zu erkennen. Befallene Dateien werden (je nach Anweisung) markiert oder auch bereinigt, auf Wunsch mitgesichert.

RESTORE - die große Bedeutung des Rücksicherns

Das Sichern ist die wichtige Voraussetzung, um Daten später rückholen zu können - für den Zeitpunkt, wenn Daten benötigt werden, die sich (nur mehr) auf einem Datenband befinden. Hier wird erst die Bedeutung des verläßlichen Rücksicherns deutlich.

So funktioniert das Restore von Backup Exec

Wie kann man vorgehen, wenn z.B. Fehler passiert sind, jemand unabsichtlich eine Datei gelöscht hat und nun genau diese Datei möglichst schnell wieder da sein sollte?

Mit Backup Exec suchen Sie aus der Übersicht der Bänder bzw. Medienübersicht die benötigte Datei heraus, markieren diese und lassen sie rücksichern. Da es wirklich so einfach funktioniert, stellt das Rücksichern kein Problem dar.

Zusatzmodul Intelligent Disaster Recovery (IDR)

Hatten Sie schon einmal einen Win NT Workstation oder Server Systemabsturz?

Diejenigen, die bereits so einen Systemabsturz erlebt haben und anschließend alles wieder zum Laufen bringen mußten, wissen, daß es Stunden (sehr oft auch die Nacht, da die Geräte am nächsten Tag wieder zur Verfügung stehen müssen) dauer Daten nicht zurückspielen und muß das System neu aufsetzen. Auch mit einer Notfallsdiskette bedeutet das im Idealfall 2 Stunden Arbeit, im Normalfall dauert es viel länger.

Rasches Recovery mit IDR

IDR erstellt einen "Schnappschuß" der Konfiguration eines gesamten Systems und kann mit einer speziell erstellten Diskette + Bandsicherung - oder (neu!) einer CD, das System in kürzester Zeit - etwa 20 Minuten - wieder aufsetzen!

Veritas Backup Exec ist für verschiedene Betriebssysteme und Netzwerke verfügbar, bitte fragen Sie uns nach:

Backup Exec für Windows NT (Single oder Multi-Server), Desktop 98 für Einzel-PCs mit Windows 98, Backup Exec für Netware (Single oder Multi-Server), dem Zusatzmodul IDR u.v.a.

NEU: Für Schulen und Universitäten sind VERITAS Produkte zu speziell günstigen Preisen erhältlich!

Fragen Sie nach Veritas Sicherungs-Software für Firmen und Schulen (Universitäten) bei:

PABLITOS Software GmbH

Edelsbachstraße 50
8063 Eggerdorf
bei Graz

Tel: 03117-51 01 – 0
Fax: 03117-51 01 – 90
E-Mail: veritas@pablitos.co.at
Internet: http://www.pablitos.co.at

ERFOLGREICH STARTEN

MIT DEM INFINEON C167-STARTERKIT UND DEM SOFTWARE-ENTWICKLUNGS SYSTEM VON KEIL

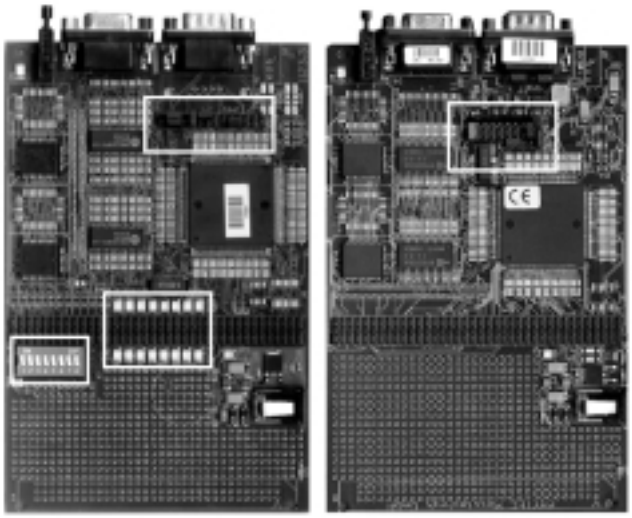
Walter Waldner

1. Einleitung

Mit dem **Infineon C167-Starterkit** erhält man alles, um die Welt der 16-Bit-Mikrocontroller erforschen zu können. Das Kit enthält:

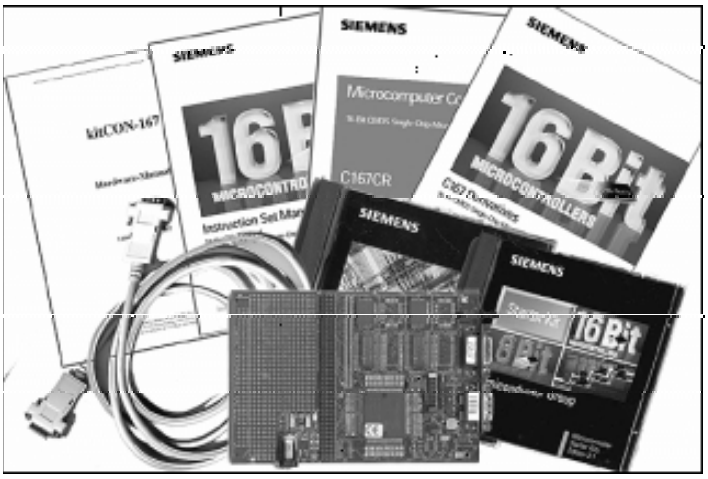
- das kitCON-167 Evaluation-Board von Phytec mit dem Infineon 16-Bit-Mikrocontroller C167CR-LM, 64KB RAM und 256 KB Flash-Memory
- Eine CD-ROM mit Software und Dokumentationen
- Manuals zum C167-Mikrocontroller
- serielles Kabel zur Verbindung des kitCON-Boards mit dem PC

Das Starterkit wurde Mitte 1998 neu aufgelegt. Diese Ausgabe enthält gegenüber der 1997er-Ausgabe ein geringfügig geändertes kitCON-167-Board.



Das neuere kitCON-167-Board (nachfolgend als kitCON-167-1998 bezeichnet) ist im Bild links abgebildet. Es enthält gegenüber dem kitCON-167-1997 zusätzlich 16 Leuchtdioden, die mit Port 2 des Controllers verbunden sind. Auch die Funktion der Jumper wurde etwas geändert. Neu hinzugekommen ist der Schalter SW3, über den unter anderem der Bootstrap-Modus aktiviert wird. Die nachfolgenden Ausführungen sind für beide Board-Varianten gültig. Auf die wenigen Unterschiede zwischen den beiden Platinen wird an gegebener Stelle hingewiesen werden.

Dieser Artikel beschreibt, wie Sie ihre ersten einfachen Programme auf der Starterkit-Hardware zum Laufen bringen - und das, ohne vorher die Assemblersprache des Prozessors lernen zu müssen. Heute werden Mikrocontroller fast ausschließlich in Hochsprache programmiert. Die hardwarenahe Programmiersprache C hat sich in diesem Bereich als sehr geeignet erwiesen und auch für die C166-Familie gibt es eine Reihe von leistungsfähigen Software-Entwicklungsumgebungen für C. Auf der Starterkit-CD-ROM sind einige Demoversionen dieser Werkzeuge enthalten, die unterschiedliche Einschränkungen gegenüber den (meist recht teuren) Vollversionen aufweisen. Meine Wahl fiel auf die Toolkette der Firma KEIL - ein professionelles Entwicklungssystem mit Assembler, Compiler, Linker/Locator, Intel-Hex-Konverter und Debugger/Simulator. Die KEIL-Demoversion erlaubt es, Programme bis zu 4 KB Codegröße zu übersetzen, zu linken und in die Ziel-Hardware zu laden, was für viele Experimente zum Kennenlernen aller Komponenten des C167-Mikrocontrollers völlig ausreichend ist.



Aller Anfang ist schwer - das gilt auch für die erste Inbetriebnahme des Starterkits. Zunächst geht es darum, auf der umfangreichen CD-ROM die richtigen Verzeichnisse und Dateien zu finden und die Keil-Software auf dem PC zu installieren. Danach sind Compiler, Linker und Debugger zu konfigurieren. Schließlich gibt es noch verschiedene Möglichkeiten, das Programm auf dem kitCon167-Board zum Laufen zu bringen.

Dieses Dokument soll Ihnen helfen, sich in dieser Vielfalt zurecht zu finden und nach Ihren ersten Experimenten werden Sie wahrscheinlich meine Meinung teilen, dass das C167-Starterkit ein ganz fantastisches Paket ist, mit dem man hervorragend arbeiten kann, das leistungsfähig und doch einfach zu programmieren ist, wenn man weiß, wie.

2. Erforderliche Vorkenntnisse

Zunächst sollten Sie Teile der Dokumentationen lesen, die Sie mit dem Starter-Kit erhalten haben:

- das kitCON-167-Hardware-Manual
- zumindest die Kapitel “Introduction”, “Architectural Overview”, “Memory Organization”, “Central Processing Unit”, “Parallel Ports” des C167-User-Manuals

3. Die Software-Entwicklungsumgebung von KEIL (“KEIL Tool-Kette”) PK166

Die KEIL-Tool-Kette (Professional Developers Kit PK166 Version 3.xx) stellt ein vollständiges Entwicklungssystem für alle Mikrocontroller der C166-Familie dar. Sie enthält unter anderem einen C166-Assembler, einen für den C167-Mikrocontroller zugeschnittenen und optimierten C-Compiler, einen Linker/Locater und einen Simulator/Debugger.

Der Anwender arbeitet dabei unter Windows 3.x / 9x /NT mit einer integrierten Entwicklungsumgebung (μVision 1.xx), die das Projektmanagement übernimmt und die erforderlichen Schritte vom Source-Code bis zur lauffähigen Version weitgehend automatisiert, nachdem die Oberfläche entsprechend konfiguriert wurde.

Die Source-Files können in Assembler- oder C-Code erstellt werden. Die Übersetzung in den C-167-Objectcode erfolgt durch den Assembler A166 bzw. durch den ANSI-C-Compiler C166. Der Übersetzungsvorgang wird in einer List-Datei (Extension .LST)

Ihr Start in das

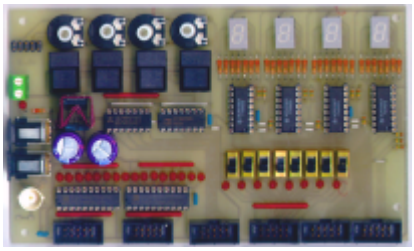
Infineon technologies

Mikrocontroller-Universum

INFINEON (Siemens) Starterkit's

sind für folgende Mikrocontroller erhältlich:
C504, C505C, C515C, C161, C163, C164CI und C167CR
sowie C541USB und verschiedene OTP (PRIME) Versionen.

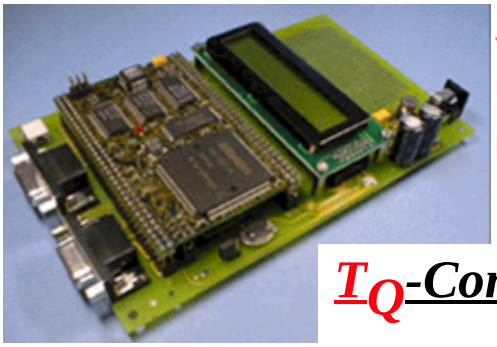
Alle Siemens Starterkit´s sind mit "ready to use" Board, Software (C-Compiler, Debugger usw. in eingeschränkter Form auf CD-ROM) , Kabel und umfangreicher Beschreibung ausgestattet.



EXBO - Die perfekte Erweiterung Ihres Starterkits

Das Demoboard kommt unbestückt mit Bauanleitung.

Funktionalität: 4x Poti für Analog In; 4x Taster entprellt; 4x SiebenSeg.Anzeige; 8x Schalter mit opt.Kontrolle; 16 LED's; Spannungs Stabilisator; Oszi-Anschluß; Verbindung über Pfostenstecker



TQ - Starterkit's für die professionelle Anwendung

Mit Minimodule schneller zum Erfolg.

Die TQ-Starterkits eröffnen Ihnen die Welt der C167CR Minimodule.

Ausgerüstet sind die TQ-Starterkits mit einem Display, serieller und CAN-Schnittstelle, Reset-Taster, 16 LED's, Batterie zur Pufferung des SRAM's und einem Steckernetzgerät. Monitorprogram, Flash Programmierung und Demo-Compiler runden das Angebot ab.

TQ-Components

TASKING
Quality Development Tools Worldwide

C/C++/EC++ - Compiler und Debugger

Der Standard für Entwicklungstools der C166 Familie

MTM SYSTEME
Messen Testen Messen

Ing. Gerhard Muttenthaler
Hirschstettnerstraße 21
A-1220 Wien

(+43 1 2032814
7 +43 1 2021303
e-✉ g.muttenthaler@mtm.at

genaue Informationen unter
. ***www.mtm.at***

Name:
Firma:
Abteilung:
Straße:
PLZ/Ort:
Tel/Fax:
e-mail:
Signum:

Ich bestelle wie folgt:

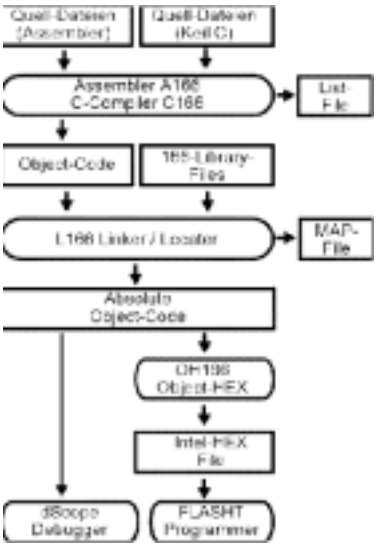
Stück	*Artikel*	*Preis (ATS)*
....	Starterkit C504	1.775,-
....	Starterkit C504-PRIME	3.540,-
....	Starterkit C505C	1.775,-
....	Starterkit C505C-PRIME	3.540,-
....	Starterkit C513	1.775,-
....	Starterkit C513-PRIME	3.540,-
....	Starterkit C515C	1.775,-
....	Starterkit C515C-PRIME	3.540,-
....	Starterkit C541USB	1.990,-
....	CAN Starterkit Adapter	1.600,-
....	OTP- Adapter MQFP44	1.420,-
....	OTP- Adapter MQFP80	1.600,-

Stück	*Artikel*	*Preis (ATS)*
....	Starterkit C161	1.990,-
....	Starterkit C163	1.990,-
....	Starterkit C164CI	1.990,-
....	Starterkit C164CI-PRIME	3.540,-
....	Starterkit C167CR	1.990,-
....	EXBO	300,-
....	Minimodul TQM167CA0	3.020,-
....	Minimodul TQM167LCB0	2.500,-
....	Starterkit STK167CA0	5.600,-
....	Starterkit STK167CB0	5.600,-

Bei den beschriebenen Produkten können keine Rabatte gegeben werden. Alle Lieferungen können nur per Nachnahme, Selbstabholung mit Barzahlung oder gegen Vorauskasse erfolgen. Für Verpackung und Versand müssen wir ATS 150,- in Rechnung stellen. Die Preise sind exklusive 20% Mehrwertsteuer.

MESSTECHNIK - ENTWICKLUNGSTOOLS - INDUSTRIECOMPUTER - DIENSTLEISTUNGEN

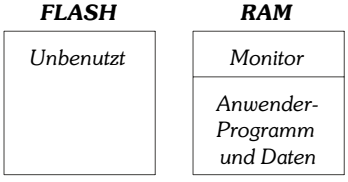
dokumentiert. Die so erzeugten Object-Files (relocateable = verschiebbar) können nicht direkt ausgeführt werden. Der L166-Linker/Locater verknüpft die Object-Dateien und Bibliotheksfunktionen, löst Symbole auf und erzeugt absolute Object-Dateien (mit absoluten Adressen innerhalb des 16 MB großen Adressraum des C167-Mikrocontrollers). Das Protokoll des Link/Locator-Laufes kann in der Map-Datei (Extension .M66) betrachtet werden. Die Object-Datei kann anschließend mit dem dScope-Debugger (und einem Monitorprogramm, das auf dem C167 läuft) in das SRAM geladen und ausgeführt werden. Alternativ läßt sich ein Programm aber auch in das Flash-Memory laden. Dazu verwendet man das Tool FLASHT, das als Eingabeformat allerdings eine Intel-Hex-Datei (ASCII-Format) erfordert. Diese Umwandlung erledigt das Programm OH166.



4. Wohin mit dem Programm ?

Eine ganz wichtige Frage, die den Programmentwicklungsprozess wesentlich beeinflusst, ist die Entscheidung, wohin das vom Anwender entwickelte Programm gespeichert werden soll. Das kitCON bietet ein SRAM und ein Flash-Memory an. Dementsprechend gibt es im wesentlichen zwei Möglichkeiten. Das Laden des Anwendungsprogramms auf die Ziel-Hardware erfolgt dabei stets über die serielle Leitung. Das kitCON-167 Board wird mit dem seriellen Kabel (liegt dem Starterkit bei) mit der COM-Schnittstelle des PCs verbunden. Am Evaluation-Board wird das Kabel an den Stecker neben dem Reset-Schalter angeschlossen. Ein Tipp: wird das Kabel zuerst mit der seriellen Schnittstelle des PCs verbunden, passt das andere Ende des Kabels ohnehin nur in die eine der beiden Anschlussbuchsen des kitCON.

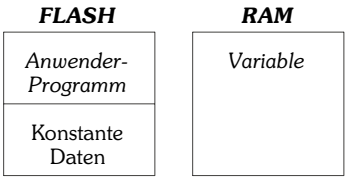
4.1 Möglichkeit 1 - RAM



Während der Entwicklungsphase einer Anwendung, insbesondere aber auch in der Ausbildung (Schulbetrieb), ist es sinnvoll, diese Variante zu wählen, da Programmänderungen sofort ins SRAM geschrieben werden können. Eine wesentliche Rolle spielt dabei der Monitor. Der Monitor ist Bestandteil der Toolkette und wird mit dem Keil-Simulator-Debugger dScope und dem in den C167-Mikrocontroller integrierten Bootstrap-Loader (über die serielle Verbindung) in das RAM des Phytec-Boards geladen. Diese Software erlaubt es, anschließend unser Anwendungsprogramm in das RAM zu laden und zu starten. Darüber hinaus kommuniziert das Monitor-Programm mit der dScope-Oberfläche und ermöglicht so ein sehr komfortables Debuggen unserer Software (Einzelschritt-Abarbeitung, Betrachten von Speicherinhalten, Anzeige von Special Function Registern und vieles mehr).

Ist unser Programm fertig entwickelt, kann es in das Flash-Memory programmiert werden. Das ist Möglichkeit 2 mit folgender Aufteilung.

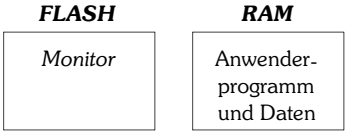
4.2 Möglichkeit 2 - FLASH Memory



Diese Variante hat natürlich den Vorteil, unser Programm nun permanent gespeichert ist und automatisch startet, sobald das kitCon-Board mit Spannung versorgt wird oder der RESET-Knopf gedrückt wird. Das Programmieren des Flash-EEPROMs wird nicht direkt durch das Keil-Entwicklungssystem unterstützt. Dazu muss das DOS-Programm `FLASHT.EXE` verwendet werden, das sich ebenfalls auf der Starterkit-CD-ROM befindet. Da der Flash-Speicher etwa 100.000 Schreibzyklen verträgt, kann dennoch bei Bedarf jederzeit eine Neuprogrammierung erfolgen.

Schließlich gibt es noch eine Variante zur Möglichkeit 1:

4.3 Möglichkeit 3 - FLASH / RAM



In dieser Variante wird das Monitor-Programm in das Flash-Memory geladen und steht somit ohne Neuladen nach jeder Spannungsunterbrechung zur Verfügung. Da der Monitor aber ein relativ kleines Programm ist (ca 5 kB), das schnell über die serielle Schnittstelle übertragen wird, bringt dies nur geringe Geschwindigkeitsvorteile. Will man allerdings das RAM auf Grund großer Datenmengen bis zum letzten Byte nutzen, ist das Laden des Monitors in den Flash-Speicher eine sinnvolle Methode.

Zu den verschiedenen Methoden des Arbeitens und Konfigurierens des Monitors habe ich ein eigenes Dokument erstellt (***siehe [1]***). Ich werde hier daher auf die verschiedenen Varianten nicht mehr näher eingehen, sondern beschreibe den direktesten und einfachsten Weg zum ersten lauffähigen C167-Programm.

5. Installation der KEIL-Toolkette

Zur Installation legen Sie die Starter-Kit-CD-ROM ein und starten die Datei `SETUP.EXE` im Verzeichnis `X:\CDROM\3RDTOOLS\KEIL\C166`

Während der Installation werden Sie nach einem Ziel-Verzeichnis gefragt. Für die folgenden Ausführungen wird angenommen, dass Sie die Standardvorgabe eingeben: `C:\C166EVAL`

Das gesamte System benötigt etwa 6.5 MB auf Ihrer Festplatte. Bei diesem Entwicklungssystem handelt es sich um eine Demoversion, die den vollen Leistungsumfang anbietet - nur die Codegröße der Programme ist mit 4 kB beschränkt, was für das

Kennenlernen des C167-Mikrocontrollers eine wenig relevante Einschränkung ist.

Nach der Installation des Demo-Pakets müssen nun noch die richtigen Dateien für den Monitor in das Verzeichnis BIN kopiert werden. Der Monitor ist jenes Programm, das in den Speicher des Evaluation-Boards geladen wird und das Laden von Anwendungsprogrammen, sowie das Debuggen dieser Programme ermöglicht, indem es über die serielle Schnittstelle mit dem PC kommuniziert (Variante 1, wie in ***4.1.*** beschrieben).

Auf der CD-ROM findet man an vielen Stellen Monitor-Programme, aber nur eines funktioniert mit dem Phytec kitCON-Board. Und diese Dateien befinden sich etwas versteckt (und ohne entsprechend Hinweise) im Verzeichnis

```
CDROM\STARTKIT\SK_167\MONITOR\KEIL
```

Kopieren Sie aus diesem Verzeichnis die beiden Dateien `BOOT` und `MONITOR` in das BIN-Verzeichnis des KEIL-Software-Pakets. Falls Sie bei der Installation die Default-Vorgabe übernommen haben, ist dies das Verzeichnis

```
C:\C166EVAL\BIN
```

Verbinden Sie nun eine serielle Schnittstelle des PCs (z.B. COM1) mit der Buchse P1 des kitCON-167-Boards. Ein Tipp: stecken Sie das Kabel zuerst an Ihrem PC ein, dann passt es am kitCON-Board ohnehin nur in die Buchse P1.

Die serielle Verbindung zwischen PC und dem C167-Board erfüllt mehrere Aufgaben:

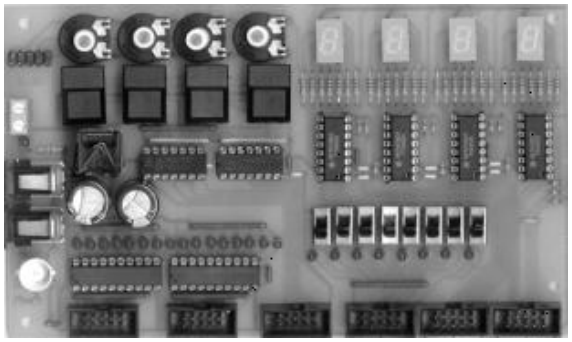
- Programme können damit wahlweise in das RAM oder das Flash-Memory des Boards geladen werden
- Bei der Arbeit mit dem Debugger werden Kommandos und Daten zwischen PC und Board transferiert
- Die C-Library des Keil-Systems enthält Routinen, mit denen User-Programme Daten über die serielle Schnittstelle senden und empfangen können

Wir wollen nun anhand eines einfachen Beispiels die Schritte vom Source-Code bis zum Ausführen des Anwendungsprogramms auf der Zielhardware zeigen. Wie unter ***4.1*** und ***4.2*** beschrieben, gibt es zwei grundsätzlich verschiedene Möglichkeiten, die beide hier dargestellt werden.

6. LED-Lauflicht - das erste C167-Projekt

6.1 Die Schaltung

Ein Mikrocontroller-Board ohne Peripherie (Sensorik, Ein- und Ausgabekomponenten) macht wenig Sinn. Insbesondere zum Experimentieren und Kennenlernen des Aufbaus und der Funktionsweise des Mikrocontrollers und seiner Bestandteile sind solche Zusatzkomponenten wünschenswert.



Ein **Experimentierboard** (EXBO), das genau auf die Phytec-kitCON-167-Platine zugeschnitten ist, ist inzwischen entwickelt worden und kann nachgebaut werden. Informationen über dieses Board und die zugehörigen Dateien zum Laden auf Ihren PC finden Sie auf meinen Webseiten im Internet (siehe ***[5]***). Das vorige Bild zeigt das Experimentierboard EXBO.

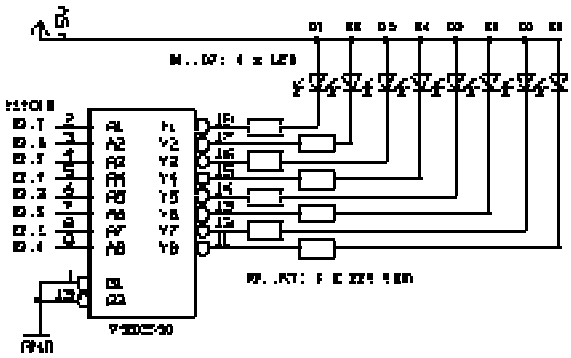
Wir werden ein Lauflicht-Programm schreiben und anhand dieses Beispiels die wesentlichen Schritte der Software-Entwicklung beschreiben.

Für dieses Projekt nehmen wir an, dass 8 Leuchtdioden an das Port 2 (Pins 0 bis 7) des kitCON-C167 angeschlossen sind.

Besitzer des kitCON-167-1998 haben solche, mit dem Port 2 verbundene Leuchtdioden, bereits auf dem Phytec-Board (siehe kitCON Hardware-Manual Version 2.0 7/1998). Die Kathoden der (low-current) LEDs sind am kitCON-167-1998 direkt mit den Pins des Ports 2 verbunden. Die Anoden sind über Vorwiderstände an 5 Volt geführt. Beachten Sie, dass die LEDs daher genau dann leuchten, wenn die entsprechenden Pins des Ports 2 LOW-Pegel aufweisen.

Wenn Sie mit EXBO arbeiten, ist lediglich eine Verbindung mit Flachbandkabel zwischen der LED-Steckerleiste des EXBO und dem kitCON-167erforderlich. Die LEDs am EXBO leuchten genau dann, wenn der Eingangspegel HIGH ist.

Sie können aber auch die einfache, nachfolgend abgebildete Schaltung aufbauen und mit den Pins 0 bis 7 des Ports 2 des kitCON-167-Boards verbinden.



Diese Schaltung verwendet den CMOS-Baustein 74HC540 (8fach invertierender Buffer/Line Driver mit Tristate-Ausgängen) zur Ansteuerung der Leuchtdioden. Die Pins des C167 erlauben nur einen Sink-/Source-Strom von etwa 1 mA, was für Standard-LEDs viel zu wenig ist. Die Ausgänge des 74HC540 hingegen können mit maximal 35 mA belastet werden. Der Strom durch die Leuchtdioden wird durch Widerstände von jeweils 220 Ohm auf ca. 15 mA beschränkt. Bei dieser Schaltung leuchten die LEDs genau dann, wenn der Eingangspegel HIGH ist.

6.2 Die µVision-Oberfläche (Version 1.xx)

Das Lauflicht-Programm wird mit der Sprache C unter der µVision-Oberfläche von KEIL entwickelt. Starten Sie nun das Programm µVision. Das Setup-Programm sollte eine Programmgruppe eingerichtet haben (unter Windows 95 im Start-Menü). Ansonsten finden Sie µVision unter dem Dateinamen `UVW166E.EXE` im Verzeichnis `C:\C166EVAL\BIN`.

Wie andere Programmierumgebung auch, verwaltet µVision Projekte. Zu einem **Projekt** gehören sämtliche Source-Dateien, Bibliotheken und Compiler/Linker-Einstellungen. Aus diesen Projekt-Elementen erstellt µVision ein lauffähiges C167-Programm, wobei die in 3 beschriebenen Stufen weitgehend automatisiert durchlaufen werden. Auch den dScope-Simulator/Debugger können wir über die µVision-Oberfläche aufrufen.

Unser Lauflicht-Projekt wird nur aus einer einzigen Quell-Datei (einem C-Programm) bestehen.

Wählen Sie im Menü: ***FILE - NEW***. Ein Editor-Fenster öffnet sich. Speichern Sie die noch leere Datei gleich einmal mit der Extension .C ab, um den Syntax-Check und die Farbkennung der C-Sprachelemente zu aktivieren. Für die weiteren Ausführungen nehmen wir an, dass Sie der Datei den Namen `LLICHT.C` gegeben haben und unter `C:\166EVAL` ein Verzeichnis `PROJECTS` angelegt haben, in das `LLICHT.C` gespeichert wird.

6.3 Der C-Source-Code

Hier ist nun der Source-Code für unser Lauflicht-Programm:

```
// Lauflicht ueber Port 2 // Walter Waldner, 1998/07
#include <reg167.h>

void warten(unsigned int w);
const unsigned int dauer = 0x2000;

void main(void)
{
  unsigned int x;

  // Pins 2.0 bis 2.7 als Ausgänge
  DP2 = 0x00FF;
  ODP2 = 0x0000;
  // Timer 3 konfigurieren
  T3CON = 0x0007;
  while (1)
  {
    for (x=1; x<=0x0080; x=x<<1)
    {
      P2 = x;
      warten(dauer);
    }
    for (x=0x0040; x>=0x0002; x=x>>1)
    {
      P2 = x;
      warten(dauer);
    }
  }
}

void warten(unsigned int w)
{
  T3 = 0;
  T3R = 1;
  while (T3 <= w);
  T3R = 0;
}
```

ACHTUNG: Wenn Sie für das Projekt die am kitCON-167-1998 vorhandenen LEDs verwenden, sind die beiden Anweisungen

```
P2 = x;
```

durch die Anweisung

```
P2 = x ^0x00FF;
```

zu ersetzen, da wir möchten, dass die LEDs leuchten, wenn die entsprechenden Bits in der Variablen `x` den Wert 1 haben (siehe Punkt ***6.1***).

Sehen wir uns nun den Source-Code und die Besonderheiten des Keil-Compilers an:

Alle SFRs (special function registers), aber auch Bitgruppen oder einzelne Bits der Register können über die Namen angesprochen werden, die im User-Manual des C167 definiert sind. Dazu ist lediglich die Header-Datei `reg167.h` mit der `#include`-Anweisung zu laden. Der Keil-C-Compiler entspricht dem ANSI-Sprachumfang. Jeder Programmierer mit C-Erfahrung wird sich schnell zurechtfinden.

Die 8 Leuchtdioden sind für unser Experiment mit den Pins 0 bis 7 (Low-Byte) des 16-Bit-Ports P2 verbunden. Diese Pins sind zunächst als Ausgänge zu definieren. Dazu wird in das Direction-Register DP2 der hexadezimale Wert 0x00FF geschrieben (ein 1-Wert auf der Bitposition b schaltet das Pin b des Ports als Ausgang). Das ODP2-Register wird auf 0 gesetzt. Damit werden die als Ausgang definierten Pins des Ports 2 in den Push-pull-Modus geschalten (1-Werte würden den Open-drain-Modus wählen). Die beiden for-Schleifen erzeugen für die Variable x der Reihe nach die Bit-Kombinationen 00000001, 00000010, 00000100, .., 10000000, 01000000 ... 00000010. Da ein 1-Wert einer leuchtenden Diode entspricht, erhalten wir so den gewünschten "Lauflicht"-Effekt. Durch die Anweisung `P2 = x` wird der Wert von x in das Port2-Register geschrieben und die entsprechenden Pegel erscheinen am Ausgang.

Nach jedem Schreibvorgang müssen wir eine Warteschleife einbauen, um den Laufeffekt überhaupt beobachten zu können. Wir könnten dies etwa durch eine for-Schleife der Art
`(for y=0; y<=30000; y++);`
erreichen. Viel attraktiver ist es allerdings, einen der zahlreichen Timer des C167 dafür zu verwenden. Für unser Beispiel setzen wir den Timer T3 ein. Durch Einschreiben eines bestimmten Wertes in das `T3CON`-Register (Timer 3 configuration register) können wir die Arbeitsweise von T3 festlegen. Lesen Sie dazu im User-Manual das ***Kapitel 9*** (General Purpose Timer Units). `T3CON = 0x0007` gibt an, dass das Zählregister T3 mit der durch 1024 geteilten internen Taktfrequenz der C167-CPU (20 MHz) angesteuert wird und aufwärts zählen soll (siehe ***Seite 9-5*** im C167 User Manual Version 2.0).

Das Unterprogramm `warten(unsigned int w)` setzt das Zählregister `T3` auf 0 und startet anschließend den Zählvorgang, indem das Run-Bit (`T3R`) auf 1 gesetzt wird. Die `while`-Schleife schafft eine Verzögerung, bis `T3` den Wert des Parameters w überschritten hat. Anschließend wird der Timer gestoppt (`T3R = 0`). `T3R` ist ein Beispiel für den Zugriff auf ein einzelnes Bit eines 16-Bit-Registers. Ein Blick in `reg167.h` zeigt, wie man solche symbolische Namen vereinbaren kann:

```
sbit T3R = T3CON^6;
```

Mit dem Datentyp `sbit` ist es möglich, einzelnen Bits eines `SFR` einen symbolischen Namen zu geben. User-Manual ***Seite 9-3*** zeigt die Bitzuordnungen für das `T3CON`-Register. `T3R` ist das Bit 6 dieses 16-Bit-Speichers.

Damit ist unser erstes Keil-Programm hinreichend beschrieben. Wir speichern das Programm mit ***FILE - SAVE*** ab (den Namen `LLICHT.C` haben wir ja bereits vergeben).

6.4 Definition eines Projekts

Bevor wir das Programm nun erfolgreich übersetzen und starten können, muss zunächst ein PROJEKT definiert werden. Wie in anderen Programmier-Umgebungen wird auch im Keil-System dabei eine `.PRJ`-Datei erstellt, die Informationen darüber enthält, welche Quell-Dateien (Assembler und/oder C) zum Projekt gehören und welche Einstellungen für den Compiler und Linker/Locater beim Erstellen des lauffähigen Programms verwendet werden sollen.

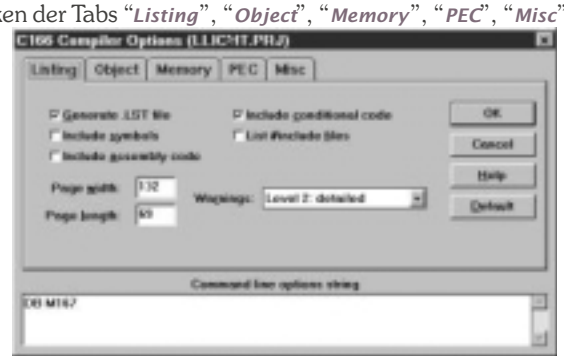
Wählen Sie im Menü den Punkt "***Project - New Project***". Geben Sie dem Projekt den Namen `LLICHT.PRJ` und speichern Sie diese (wie auch `LLICHT.C`) in das Verzeichnis `C:\C166EVAL\PROJECTS`.

In der nun erscheinenden klicken Sie auf "***Add***". In diese Liste tragen wir alle Dateien ein, die zu diesem Software-Projekt gehören. In unserem Fall ist dies nur das C-Source-Programm `LLICHT.C`. Achten Sie bitte darauf, dass das Feld "***Include in Link/Lib***" angekreuzt ist. Da unser einfaches Beispiel keine weiteren Quelldateien benötigt, drücken wir nun den Knopf "***Save***".

6.4.1 Compiler-Optionen

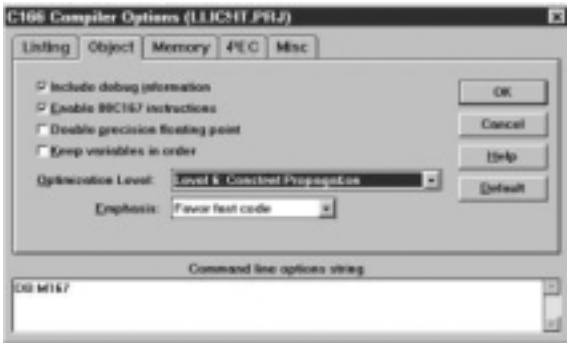
Nun müssen die Einstellungen für den C-Compiler vorgenommen werden. Wählen Sie im Menü "***Options - C166 Compiler***".

Es erscheint eine Dialogbox mit mehreren Seiten, die durch Anklicken der Tabs "***Listing***", "***Object***", "***Memory***", "***PEC***", "***Misc***" aufgerufen werden können.

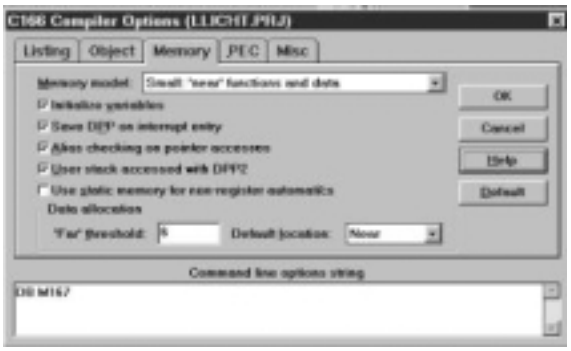


Unter "***Listing***" können Sie Parameter für die Listing-Datei angeben, die vom Compiler erzeugt wird. Für unser Projekt würde diese Datei `LLICHT.LST` heißen. Ein Blick in diese Datei offenbart, was der Compiler aus unserem Source-Programm macht. Als Einsteiger können Sie diese Datei und auch die Einstellungen getrost ignorieren.

Wichtiger ist schon die nächste Dialogbox "***Object***". Klicken Sie "***Include debug information***" und "***Enable 80C167 instructions***" an. Auf diesem Formular wird allgemein die Object-Code-Generierung gesteuert. Auch können Präferenzen für die Code-Optimierung gewählt werden.



Unter "***Memory***" wird das Speichermodell ausgewählt, das beim Compilieren und Linken verwendet werden soll.



Als Speichermodell wählen wir "***SMALL***", was bedeutet, dass der Code unseres Programms nicht größer als 64 KB sein darf und alle Programmverzweigungen (Unterprogramm-Aufrufe, bedingte und unbedingte Sprünge) innerhalb des 64 KB-Segmentes durchgeführt werden. Da die Demo-Version ohnehin ein 4 kB-Limit vorgibt, sind andere Speichermodelle nicht sinnvoll. Wählen Sie die Optionen, wie im obigen Screenshot dieser Dialogbox ersichtlich.

Die Dialogboxen "***PEC***" und "***Misc***" sind für unser erstes Beispielprogramm irrelevant. Sie können also jetzt den Knopf "***OK***" anklicken und die Einstellung der Compiler-Optionen beenden.

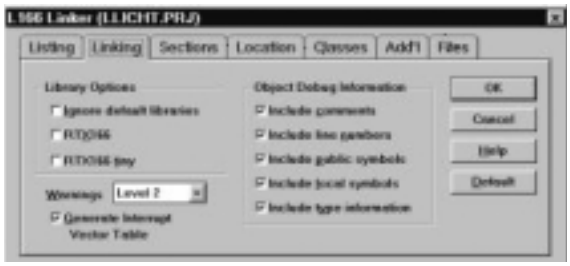
6.4.2 Linker-Optionen

Nun folgen die Einstellungen für den 166-Linker/Locater. Der Linker/Locater fügt die vom Compiler übersetzten Dateien und die Bibliotheksfunktionen zusammen und löst Symbole (Namen von Variablen, Konstanten, Unterprogrammen) in Adressen auf. Dazu muss er insbesondere Informationen darüber haben, in welche Speicherbereiche Code und Daten gelegt werden dürfen.

Wählen Sie im Menü den Punkt "***Options - L166 Linker***". Es erscheint eine Dialogbox mit den Tabs "***Listing***", "***Linking***", "***Sections***", "***Location***", "***Classes***", "***Add'l***" und "***Files***".

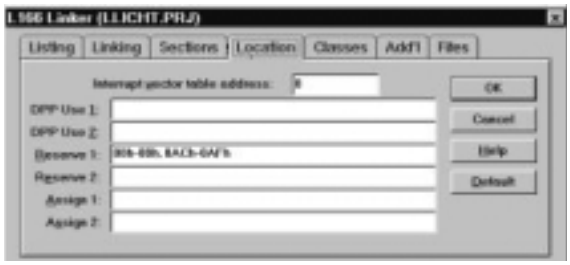
Das Formular "***Listing***" erlaubt die Angabe von Optionen für die vom Linker erzeugte Mapping-Datei. Die Voreinstellungen (siehe Screenshot) passen für unser Vorhaben.

Die folgende Seite ist Link-Optionen und Debug-Informationen gewidmet. Die Standardvorgaben laut Screenshot sollten unverändert übernommen werden.



Die Seite "***Sections***" können Sie für unser Beispiel leer lassen.

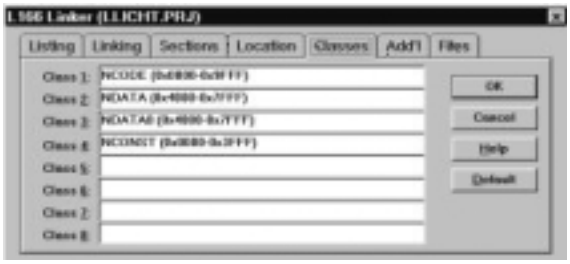
Nun folgt die Seite "***Location***". In der Zeile "***Reserve 1***" ist einzugeben:



08h-0Bh, 0ACh-0AFh

Diese Adressbereiche werden dadurch reserviert. Es handelt sich um Bereiche der Interrupt-Vektor-Tabelle. Das Monitor-Programm verwendet den NMI-Interrupt und den Receive-Interrupt der seriellen Schnittstelle. Aus diesem Grund dürfen diese Adressen nicht für den Link / Locate - Vorgang unseres Programmes verwendet werden.

Die nächste Seite "***Classes***" definiert die Adressbereiche für den Programm-Code (NCODE), die initialisierten und nicht-initialisierten Daten (NDATA0 und NDATA), sowie für die Konstanten (NCONST). Diese Angaben werden vom Linker verwendet, wenn dieser Symbole in absolute Adressen wandelt.



ACHTUNG: Die Eintragungen auf dieser Seite hängen davon ab, ob das Programm mit dScope/Monitor in das SRAM oder mit FLASHT in das Flash-Memory geladen werden soll.

Für die in ***4.1*** beschriebene Möglichkeit 1 (Programm wird vom Debugger in das SRAM geladen) können folgende Zeilen eingetragen werden:

NCODE (0x0000-0x9FFF)
NDATA (0x4000-0x7FFF)
NDATA0 (0x4000-0x7FFF)
NCONST (0x0000-0x3FFF)

Für nähere Informationen verweise ich auf meinen Artikel zum KEIL-Monitor ***[1]***.

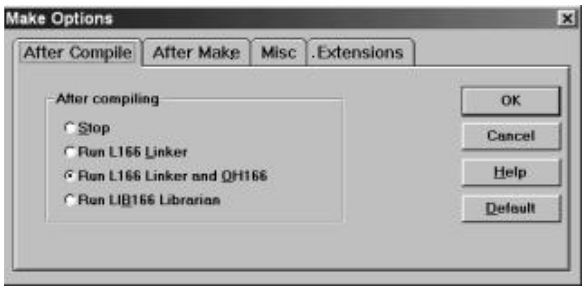
Die Einstellungen für das Laden eines Anwendungsprogramms in den Flash-Speicher beschreiben wir später (***siehe Kapitel 8***).

Damit sind die Linker-Optionen vollständig festgelegt.

6.4.3 Weitere Einstellungen

Wenn Sie Ihr Anwendungsprogramm später in den Flash-Speicher des Phytec-Boards laden möchten, ist dazu eine Datei im Intel-Hex-86-Format erforderlich. Dazu sind folgende Einstellungen erforderlich:

Wählen Sie den Menüpunkt ***Options – Make ....*** In der darauf erscheinenden Dialogbox klicken Sie auf der Seite ***After Compile*** die Option ***Run L166 Linker and OH166*** an.



Anschließend wird unter dem Menüpunkt ***Options - OH166 Object-Hex Converter*** eingestellt, welches Format die generierte HEX-Datei haben soll. Für unsere Zwecke ist als ***Output File Format*** die Option ***Intel Hex-86*** zu aktivieren.

6.4.4 Build Project

Nun kommen wir zum spannenden Augenblick. Das Keil-System ist bereit, unser Projekt in ein lauffähiges Programm zu übersetzen. Klicken Sie dazu den Menüpunkt ***Project - Make: Build Project*** an. Assembler, Compiler, Linker/Locater werden automatisch aufgerufen und die Ausgabe-Dateien werden generiert. In unserem Fall sind das die Dateien

LLICHT.OBJ	Object-Datei zum C-Sourceprogramm
LLICHT.LST	List-Datei (für den Compiler-Lauf)
LLICHT	die absolute Object-Datei
LLICHT.M66	Linker-Map-Datei

Eventuell wird auch LLICHT.H86 (die Intel-Hex-86-Datei) generiert (***gemäß Abschnitt 6.4.3***).

Beachten Sie bitte: die absolute Object-Datei, die wir nun anschließend mit Hilfe des Monitors in die Zielhardware laden werden, hat keine File-Extension.

7. Der Simulator/Debugger dScope/tScope

Ein sehr interessantes, umfangreiches (aber auch komplexes) Tool in der Keil-Software-Kette ist der Simulator/Debugger (dScope).

Starten Sie das Programm durch die Auswahl des Menüpunktes "***Run dScope Debugger***" (auf der Festplatte heißt das Programm DSW166.EXE und es befindet sich im Verzeichnis C:\C166EVAL\BIN). Mit dScope können Sie Ihr Anwendungsprogramm entweder am PC simulieren oder aber über ein Monitorprogramm in die Zielhardware laden und die üblichen Funktionen eines Debuggers verwenden. Dazu gehören:

- Schrittweiser Programmablauf
- Setzen / Löschen von Breakpoints
- Memory-Dumps
- Anzeigen von SFR-Inhalten
- De-Assemblieren von Code
- und vieles mehr

Wir werden uns hier auf das Arbeiten mit **dScope** als Debugger für das Programm in der Zielhardware beschränken (in dieser Form heißt der Debugger **tScope**).

7.1 Monitor und Anwendungsprogramm in die Zielhardware laden (SRAM)

Vergewissern Sie sich, dass Sie das Phytec kitCON-Board mit dem seriellen Kabel, das dem Starterkit beiliegt, verbunden ist. Für die folgenden Aktionen muss am kitCON-Board der **Bootstrap-Modus** aktiviert werden. Der Bootstrap-Modus erlaubt über eine fest in den C167 programmierte Codesequenz das Laden von 32 Byte Programmcode. Dieser 32 Byte Code kann dann weitere Programmteile laden. Hier ist nun zu unterscheiden, welche kitCON-Variante Sie in Ihrem Starterkit vorgefunden haben. Auf dem kitCON-167-1998 ist der **Schalter 1** von **SW3** auf **ON** zu stellen. Besitzer des kitCON-167-1997 müssen den Jumper **JP2** mit dem (roten) Stecker schliessen (Pins 1 und 2 dieses Jumpers verbinden). Drücken Sie in beiden Fällen danach die **RESET**-Taste am kitCON-Board.

Der tScope-Debugger benutzt den Bootstrap-Mechanismus, um das Monitorprogramm in das SRAM des Phytec-Boards zu laden. Zunächst wird über die C167-Bootsequenz das Programm BOOT geladen und gestartet. Diese Software lädt anschließend das wesentlich größere Programm MONITOR. Wie in ***Kapitel 5*** beschrieben wurde, müssen sich die beiden absoluten Object-Dateien `BOOT` und `MONITOR` im Verzeichnis `C166EVAL\BIN` befinden. Das Monitorprogramm kommuniziert über die serielle Schnittstelle mit der tScope-Oberfläche. Kommandos können so an das Monitorprogramm gesandt und ausgeführt werden (z.B. das Laden eines Anwendungsprogramms). Außerdem erlaubt der Monitor einen Blick in das "Innenleben" des C167-Mikrocontrollers. Wird die Anwendungssoftware im Einzelschritt-Betrieb abgearbeitet oder erreicht die Ausführung einen vorher definierten Breakpoint, werden die Inhalte der SFRs an tScope übertragen, wo sie angezeigt werden können. Ebenso können vom Monitor die Werte von Speicherbereichen abgerufen oder auch verändert werden. Sie können auf diese Art genau mitverfolgen, wie Ihr Programm die Speicherzellen und Registern verändert.

7.1.1 Monitor laden

Wir laden nun den Monitor mit Hilfe des Bootstrap-Loaders in das SRAM des kitCON.

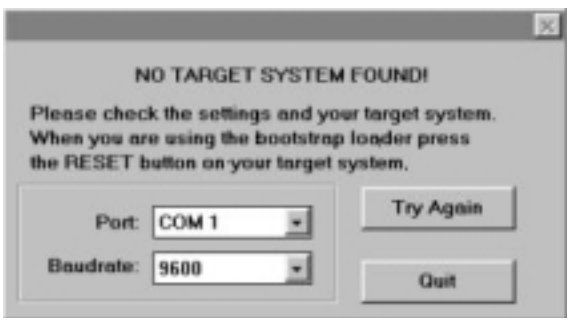
Erste Methode: Klicken Sie die Auswahlliste links oben an und wählen Sie `MON166.DLL`

Zweite Methode: Öffnen Sie das Command-Window (***View - Command Windows***) und geben Sie ein:

```
load mon166.dll
```

`MON166.DLL` ist die Schnittstelle zwischen dem Monitor-Programm (Target-Monitor), das auf dem Phytec-Board läuft und dem tScope-Debugger, der interaktiven PC-Oberfläche.

Es sollte ein Fenster mit einer Balkenanzeige erscheinen, die den Fortschritt des Ladevorgangs anzeigt.



Sollten Sie die Fehlermeldung "***NO TARGET SYSTEM FOUND***" erhalten, ist entweder die serielle Schnittstelle nicht richtig ausgewählt (COM1, COM2), oder aber die Baudrate falsch eingestellt. Sie sollten 9600 Baud verwenden, da der Monitor für diese Baudrate konfiguriert wurde. Im Artikel ***[1]*** wird beschrieben, wie höhere Geschwindigkeiten eingestellt werden können (dazu muss der MONITOR neu generiert werden).

ACHTUNG

Wir wollen an dieser Stelle nochmals darauf hinweisen, dass die Dateien `BOOT` und `MONITOR` aus dem Verzeichnis

```
CDROM\STARTKIT\SK_167\MONITOR\KEIL
```

der Starter-Kit-CD in das Verzeichnis

```
C166EVAL\BIN
```

der Keil-Installation kopiert werden müssen. Nach der Installation der Keil-Toolkette befinden sich dort Dateien `BOOT` und `MONITOR`, die nicht für das kitCON-Board gedacht sind. Auch in diesem Fall erhalten Sie die obige Fehlermeldung.

Im Command-Window sollte jetzt die Meldung erscheinen, wie sie im Bildschirmfoto am Beginn dieses Abschnitts zu sehen ist.

7.1.2 Anwendungsprogramm laden

Nun können wir unser Lauflicht-Programm in das SRAM laden. Auch hier gibt es zwei Methoden.

Erste Methode: Klicken Sie auf das linke Symbol ("geöffneter Ordner"). (siehe folgenden Screenshot)

Im darauf erscheinenden Dateiauswahlfenster selektieren Sie die absolute Object-Datei `LLICHT` (ohne Extension) aus unserem Projekt-Verzeichnis `C:\C166EVAL\PROJECTS\`

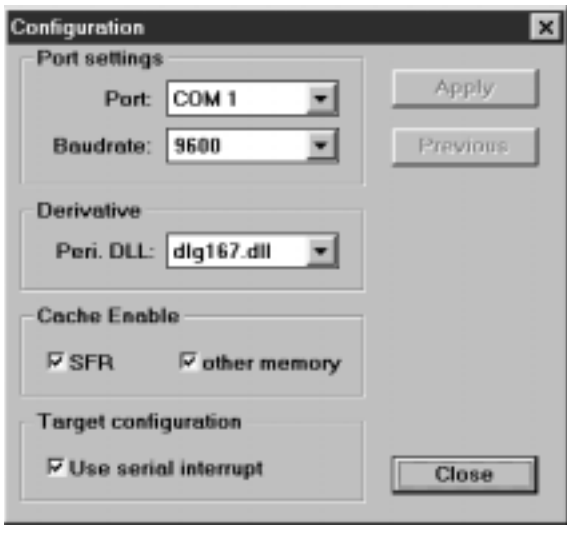
Zweite Methode: Geben Sie im Command-Window den Befehl

```
load LLICHT
```

ein.

7.1.3 Zielsystem wählen

Da die KEIL-Software sowohl für Systeme mit dem C166- als auch mit dem C167-Mikrocontroller entwickelt wurde, ist nun noch das Zielsystem auszuwählen. Wählen Sie den Menüpunkt ***Peripherals - Config*** und stellen Sie die Optionen so ein, wie dies das Bildschirmfoto zeigt.



WICHTIG: Unter "Derivative" ist als "Peri.DLL" die Datei DLG167.DLL zu wählen, da unser Zielsystem einen C167-Mikrocontroller enthält.

7.1.4 Anwendungsprogramm starten

Klicken Sie nun das Command-Window an und geben Sie dort den Befehl

```
g
```

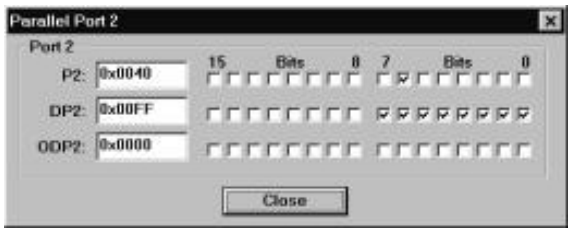
ein. Dieses Kommando (g = go) startet unser Anwendungsprogramm. Alternativ kann im Debug-Fenster der Menü-Punkt ***GO*** gewählt werden.

Die Leuchtdioden sollten im programmierten Muster aufleuchten.

Um das Programm anzuhalten, brauchen wir nur die ESC-Taste drücken (das Command-Fenster muss dazu allerdings das aktuelle Fenster sein - eventuell vorher anklicken). Auch der Menüpunkt "***STOP***" im Debug-Fenster hält ein Programm an.

Wenn das Programm angehalten wurde, kann der Inhalt von C167-Registern betrachtet werden. Unter dem Menü-Punkt ***View*** finden Sie zahlreiche Fenster, die zum Debuggen Ihres Programmes geöffnet werden können. Normalerweise ist zumindest das REGS-Fenster offen. Es zeigt die wichtigsten CPU-Register.

Die verschiedenen SFRs zur integrierten Peripherie des C167-Mikrocontrollers können Sie sich über den Menü-Punkt ***Peripherals*** anzeigen lassen. Unser Lauflicht-Programm arbeitet u.a. mit dem Port 2 und dem Timer 3. Sie können also beispielsweise die zugehörigen Fenster abrufen. Im ***Parallel Port 2*** - Window sollte genau das Bit angekreuzt sein, das HIGH-Pegel führt. Die damit verbundene Leuchtdiode sollte leuchten.



7.1.5 Automatisches Laden des Monitors und der Object-Datei

Die Keil-Oberfläche (µVision 1.xx) erlaubt es, den Monitor und das Object-File nach dem Aufruf von dScope automatisch in das RAM des kitCON-Boards zu laden. Dazu müssen wir lediglich eine kleine INI-Datei erstellen.

Beenden Sie dScope. Das Anwendungsprogramm muss vorher durch ESC unterbrochen werden. Sie befinden sich wieder in der µVision -Oberfläche. Wählen Sie ***File - New*** und geben Sie in das Editor-Fenster ein:

```
load mon166.dll
load llicht
```

Speichern Sie diese Datei unter dem Namen LLICHT.INI ab.

Nun geben Sie unter ***Options - dScope Debugger*** den Namen LLICHT.INI ein.

Wenn Sie jetzt ***Run - dScope Debugger*** anklicken, wird das Monitor- und das Object-Programm von dScope automatisch geladen. Befindet sich der Monitor noch im SRAM-Speicher, wird er nicht erneut geladen.

8. Programm-Entwicklung für das FLASH-Memory

Wie heute in der Praxis üblich, werden wir unsere Beispielprogramme in der höheren Programmiersprache C kodieren. Wenn C-Programme für PC-Systeme mit DOS / Windows erstellt werden, steht für deren Ausführung ein leistungsfähiges Betriebssystem zur Verfügung, das den Prozessor und die Peripherie bereits in einen vordefinierten Zustand gebracht hat.

Unser Mikrocontroller hingegen ist eine Stand-alone-Hardware. Für die geeignete Konfiguration der zahlreichen Komponenten unseres C167-Systems müssen wir daher selbst sorgen. Erst wenn gewisse SFRs (special function register) des C167 entsprechend gesetzt wurden, kann das Hauptprogramm main() unseres C-Sourcemoduls erfolgreich ablaufen. Unbedingte Voraussetzung dafür ist die korrekte Konfiguration des externen Buscontrollers, dessen Hauptaufgabe das Timing und die Adressierung (Chip-Auswahl) am externen Bus ist, an den die Flash- und SRAM-Speicherbausteine angeschlossen sind (START-UP-Konfiguration).

Die Startup-Konfiguration unseres Phytec-kitCon wird von einem Assembler-Modul vorgenommen, das wir mit dem leistungsfähigen Programm **DAvE** erstellen können. Es würde den Rahmen sprengen, hier auf alle Funktionen von DAvE einzugehen. Im wesentlichen ist DAvE eine Software, die es erlaubt, alle Komponenten des C167-Mikrocontrollers (oder anderer Infineon-µCs) zu konfigurieren und danach sowohl die erforderliche Assembler-Startup-Datei, aber auch C-Rahmenprogramme zu generieren. Wir werden uns hier zunächst auf die Erzeugung der Assembler-Startup-Datei beschränken.

8.1 Installation von DAvE (Version 1.x)

DAVE liegt dem Starterkit in Form einer eigenen CD-ROM bei. Legen Sie diese CD ein und starten Sie das Programm SETUP.EXE im Verzeichnis SETUP.

8.2. Generieren der STARTUP-Datei

Starten Sie nach der erfolgreichen Installation das Programm .EXE. Wählen Sie den Menüpunkt "***Project - New***" aus. Sie werden nun nach dem Mikrocontroller-Typ gefragt. Wählen Sie aus der Liste "***C167CR***" aus.

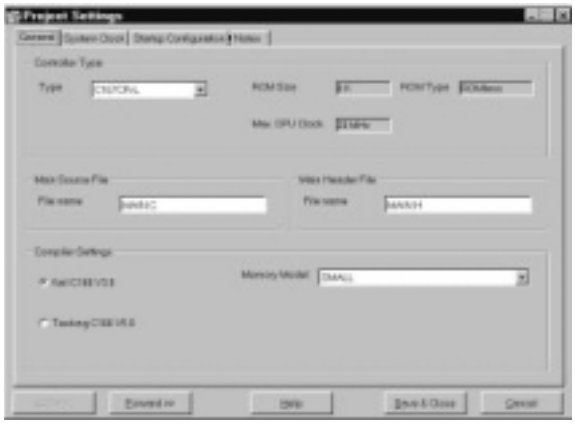
Anschließend müssen Sie Ihrem Projekt einen Namen geben. Wir wollen eine Startup-Datei erstellen, damit Anwendungsprogramme ins Flash programmiert und exekutiert werden können. Geben Sie als Name *PK167F* ein (PK für Phytec kitCON, F für Flash). Wir schlagen vor, dass Sie im Verzeichnis C:\C166EVAL\PROJECTS ein entsprechendes DAvE-Verzeichnis anlegen und dieDAvE-Projekt-Datei in dieses Verzeichnis speichern.

Auf der nun erscheinenden Formular-Seite wählen Sie unter "***Type***" den Chip "***C167CR-L***", der sich auf dem Phytec-kitCON-167-Board befindet.

Unter "***Compiler Settings***" klicken Sie die Option "***Keil C166 V3.0***" an. Im Eingabefeld "***Memory Model***" wird "***SMALL***" eingestellt. Danach drücken Sie auf den Knopf "***Forward >>***".

Wir kommen so auf die nächste Seite. Hier ist die Taktfrequenz einzustellen. Unser Board arbeitet mit einer externen Frequenz von 5 MHz, die durch die C167-interne PLL vervierfacht wird. Die CPU arbeitet somit intern mit 20 MHz. Ihre Einstellungen soll-

ten so vorgenommen werden, wie dies aus dem Screenshot ersichtlich ist.



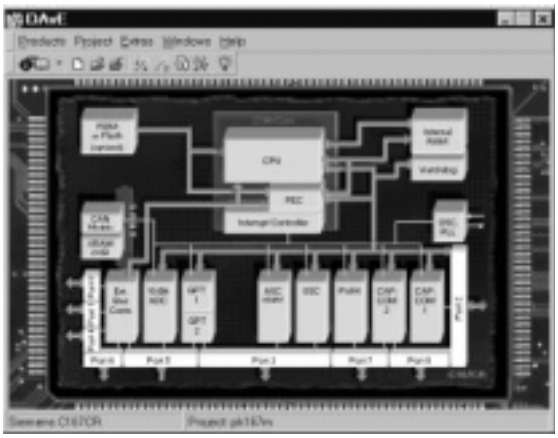
Weiter geht es wieder mit dem Button "***Forward >>***". Nun folgt die Einstellung der Startup-Konfiguration, die Sie gemäß dieses Bildschirmfotos eingeben sollten:

Aktiviert sein müssen folgende Optionen:

- Fetch code from external ROM
- Pin #WR and #BHE operate als #WRL and #WRH
- 4-Bit segment address
- 16 bit demultiplexed bus
- Five Chip Select Lines: #CS4 .. #CS0

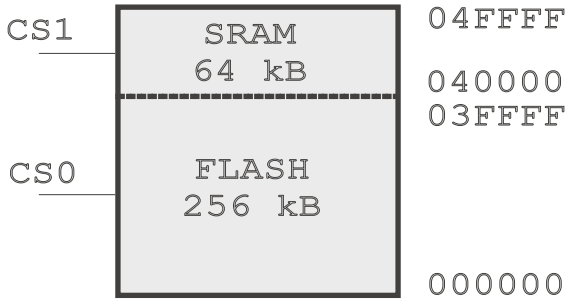
Damit ist die erste Serie von Einstellungen abgeschlossen. Drücken Sie den Knopf "***Save & Close***".

Nun kommen wir zur Konfiguration der C167-Komponenten. Wir sehen ein Bild über das "Innenleben" des C167-Mikrocontrollers. Bewegen Sie die Maus über die verschiedenen Symbole und drücken Sie die rechte Maustaste. Sie sehen, dass man durch Klick zur Konfiguration einer Komponente fortschreiten oder aber die entsprechene Seiten aus dem User-Manual anzeigen lassen kann (dazu muss der Acrobat Reader auf Ihrem System installiert sein, der sich ebenfalls auf der DAvE-CD-ROM befindet).



Für unser Lauflicht-Programm müssen wir nur eine Komponente konfigurieren - den "***External Bus Controller***".

Der "***External Bus Controller***" spielt in unserem System eine zentrale Rolle. Er ist dafür verantwortlich, dass der externe Speicher des kitCon korrekt aktiviert wird. Für ein tieferes Verständnis dieser Einheit lesen Sie bitte das kitCON-Handbuch und das ***Kapitel 8*** des User-Manuals.



Das Phytec kitCon-Board, das Sie mit dem Starterkit erhalten haben, verfügt über ein SRAM mit 64 KB (organisiert als 32 K x 16 bit) und ein FLASH-memory mit 256 KB (organisiert als 128 K x 16 bit). Das Flash-Memory wird durch CS0 (chip select 0), das RAM durch CS1 aktiviert. Der externe Bus-Controller ist äußerst flexibel und kann so konfiguriert werden, dass die externen Speicher mit dem korrekten Timing angesteuert werden. Insbesondere kann spezifiziert werden, in welchem Adressbereich des insgesamt 16 MB großen Adressraum welcher Speicher (oder auch Memory-mapped-Peripherie) liegen soll.

Im folgenden werden wir uns auf das SRAM- und das FLASH-memory des kitCONs beschränken. Wenn Sie weitere Speicher- oder andere Peripherie-Bausteine anschließen wollen, ist analog vorzugehen.

Im wesentlichen müssen wir nun mit Hilfe von DAvE die SFRs `SYSCON`, `BUSCON0`, `BUSCON1` und `ADDRSEL1` konfigurieren. Die `BUSCON`-Register legen das Timing und die Signalformen für die Ansteuerung der Speicherbausteine fest (`BUSCON0` für das FLASH memory - Speichertyp "ROM", `BUSCON1` für das SRAM - Speichertyp "RAM").

Das Register `ADDRSEL1` spezifiziert den Adressbereich für das SRAM. Der externe Buscontroller selektiert mit `CS1` das SRAM, wenn die CPU eine Adresse anspricht, die in dem durch den Wert von `ADDRSEL1` festgelegten Bereich liegt. Liegt die Adresse nicht in diesem Bereich, wird `CS0` und damit das Flash-Memory aktiviert.

Der externe Buscontroller wird natürlich nur dann tätig, wenn die angesprochene Adresse tatsächlich zu externem Speicher gehört und nicht etwa eine internes Register, internes RAM oder XRAM selektiert. Die Speicher-Organisation des C167 ist im ***Kapitel 3 des User-Manuals*** nachzulesen.

Wenn wir ein Programm in das Flash-Memory des Phytec-kitCON-167-Boards speichern möchten, haben wir für die Adresszuordnungen und damit insbesondere für die Konfiguration des Registers `ADDRSEL1` nicht viel Wahlmöglichkeiten. Der Code-Teil eines Programms muss beim C167 stets in einen Bereich gelegt werden, der bei der physikalischen Adresse 000000 beginnt, da dort die Interrupt-Vektortabelle gespeichert ist. Selbst wenn Sie keine Interrupt-Service-Routinen programmiert haben, muss zumindest der RESET-Vektor (auf die Adresse 0) geschrieben werden.

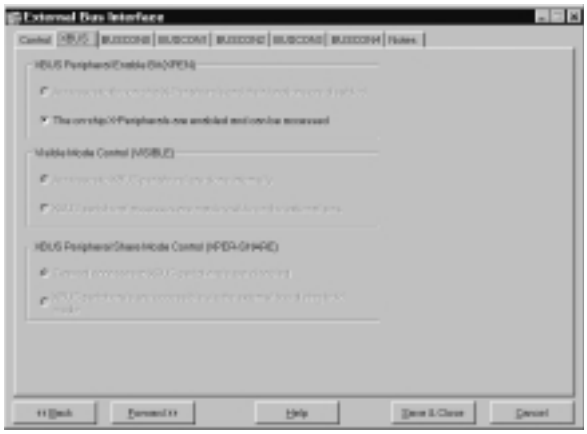
Aus diesen Überlegungen folgt, dass wir das Flash-Memory auf die Adresse 0 legen müssen. Das SRAM legen wir in den Bereich unmittelbar nach dem Flash.

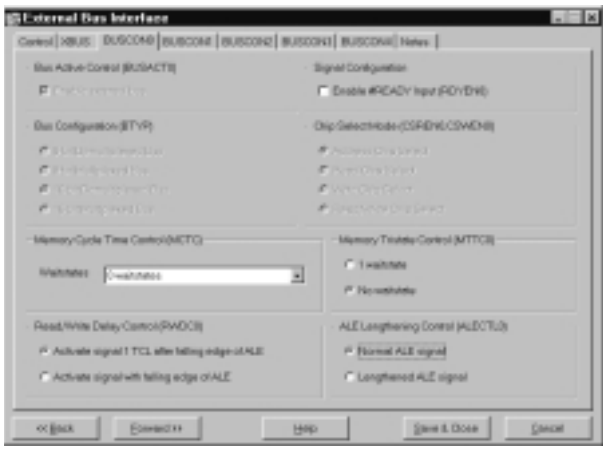
Speicherbereich	**Speichertyp**
000000-03FFFF	256 KB
040000-04FFFF	Flash-Memory 64 KB SRAM

Bemerkung: Wenn wir mit dem Monitor (und dScope) arbeiten und unser Programm in das SRAM speichern, ist eine andere Adresszuordnung erforderlich. In diesem Fall wird das Register `ADDRSEL1` von den Programmen `BOOT` und `MONITOR` so gesetzt, dass das SRAM auf der Adresse 0 liegt (***siehe Artikel [1]***).

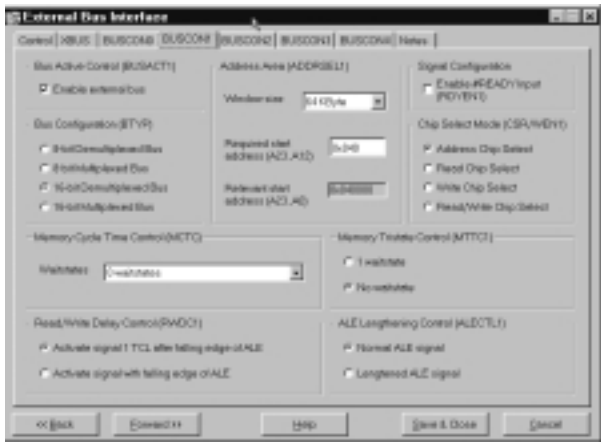
Nun aber zur Konfiguration der `BUSCON`- und `ADDRSEL`-Register. Bewegen Sie die Maus auf das Symbol "***Ext. Bus Contr.***" rechts unten, klicken Sie auf die rechte Maustaste und wählen Sie "***Configure***".

Stellen Sie der Reihe nach die Optionen auf den Formularseiten gemäß den folgenden Bildschirmfotos ein.





Das kitCON-167-Board hat nur zwei externe Speicherkomponenten (FLASH und RAM). Wir benötigen also keine weiteren Adressfenster. Aus diesem Grund werden für `BUSCON2`, `BUSCON3` und `BUSCON4` keine Einstellungen vorgenommen. Die dazugehörigen Seiten füllen Sie nicht aus, da die Standardvorgaben passen.



Einige Erläuterungen zu diesen Parametern:

Die wesentlichen Informationen für diese Einstellungen finden Sie im kitCON-167-Handbuch. Beide Speicherkomponenten (SRAM und FLASH) sind 16 Bit breit und werden über einen "demultiplexed bus" (getrennter Daten- und Adressbus) angesteuert. Die Parameter für das Bus-Timing sind im kitCON-Handbuch nachzulesen:

- 0 Waitstates (Memory Cycle Time Control)
- verzögertes R/W-Signal (Read/Write Delay Control)
- keine Tristate-Verzögerung (Memory Tristate Control)
- normales ALE-Signal (ALE Lengthening Control)
- kein READY-Signal

(Das ALE-Signal wird nur bei Multiplex-Bussen benötigt)

Drücken Sie "***Save & Close***", wenn Sie die vier Seiten "***Control***", "***XBUS***", "***BUSCON0***" und "***BUSCON1***" ausgefüllt haben.

Wir haben nun die Mindesteinstellungen für das Startup-File vorgenommen. DAvE kann daraus nun eine Assembler-Datei mit dem erforderlichen Startup-Code erzeugen. Dazu wählen wir im Menü den Punkt "***Project - Generate Code***". DAvE speichert eine Assembler-Datei mit dem Namen `START.ASM` in das Verzeichnis Ihrer Festplatte, das Sie vorher bei der Vergabe des Projekt-Namens gewählt haben. DAvE kann nun beendet werden. Es sei an dieser Stelle nochmals darauf hingewiesen, dass DAvE ein sehr umfangreiches Entwicklungstool ist, mit dem Sie sich eingehend beschäftigen sollten, wenn Sie später komplexere Projekte mit Ihrem C167-Mikrocontroller planen.

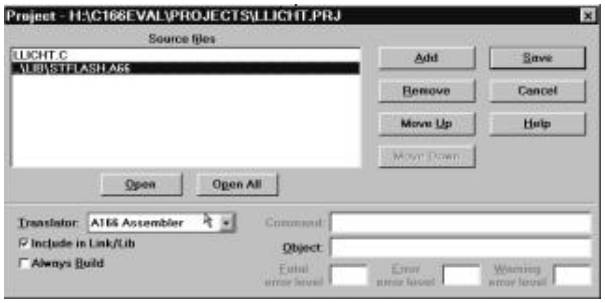
Kopieren Sie die von DAvE generierte Datei `START.ASM` in das LIB-Verzeichnis des KEIL-Systems. Wenn Sie bei der Installation das Default-Verzeichnis gewählt haben, ist dies `C:\C166EVAL\LIB`. Benennen Sie die Datei in `STFLASH.A66` um.

DAvE generiert nicht nur die Assembler-Datei `START.ASM`, sondern auch C-Dateien, die als "Rahmenprogramme" für die Software-Entwicklung verwendet werden können. Für unser einfaches Projekt bringen diese "Templates" keine großen Vorteile. Zum einen haben wir die C-Source-Datei `LLICHT.C` ja bereits erstellt und außerdem war es nur erforderlich, den externen Bus-Controller zu konfigurieren. Es sei aber noch einmal darauf hingewiesen, dass DAvE die Entwicklung umfangreicherer Projekte wesentlich erleichtert. Insbesondere wenn die verschiedenen internen C167-Komponenten Interrupts erzeugen sollen, sind die C-Rahmenprogramme, die von DAvE automatisch erstellt werden eine wesentliche Hilfe.

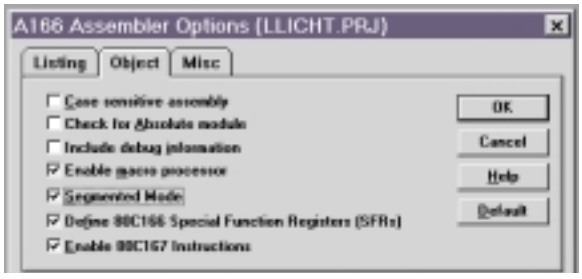
8.3 Einstellungen in der KEIL-Umgebung für das Speichern des Programms im FLASH

Damit unser Programm im Flash-Speicher laufen kann, müssen wir in der µVision-Umgebung einige wenige Änderungen gegenüber den im ***Abschnitt 6*** beschriebenen Einstellungen vornehmen.

Unser Projekt besteht jetzt nicht nur aus der C-Quelldatei .C, sondern auch aus der Assembler-Datei `STFLASH.A66`, die wir mit DAvE generiert haben. Fügen Sie die Datei `STFLASH.A66` Ihrem LLICHT-Projekt hinzu ("***Project - Edit Project - Add***"). Achten Sie darauf, dass die Option "***Include in Link/Lib***" aktiviert ist.



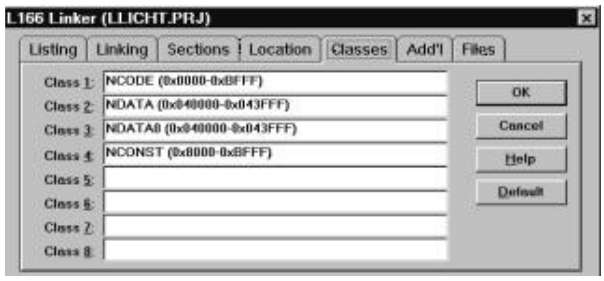
Da nun auch der Assembler am Erzeugen der Object-Datei beteiligt ist, sind auch für ihn Einstellungen unter dem Menüpunkt ***Options-A166-Assembler*** vorzunehmen. Klicken Sie, wie im folgenden Bild sichtbar, die Optionen ***Enable macro processor***, ***Segmented Mode***, ***Define 80C166 Special Function Registers*** und ***Enable 80C167 Instructions*** an.



Die Einstellungen für die CLASSES-Adressen sind wie folgt einzugeben ("***Options - L166 Linker***"):

```
NCODE       (0x0000-0xBFFF)
NDATA       (0x040000-0x043FFF)
NDATA0      (0x040000-0x043FFF)
NCONST      (0x8000-0xBFFF)
```

Der Programmcode (Class `NCODE`) und Konstante (Class `NCONST`) werden in das FLASH gespeichert, für Variable (Class `NDATA` und `NDATA0`) wird Speicher im SRAM reserviert. Konstante und Variable werden über die DPP-Register (siehe ***C167-Handbuch***) adressiert. Daher werden für sie jeweils 16 kB große Bereiche angegeben.



Für die Flash-Programmierung muss unser Programm im Intel-Hex-Format vorliegen. Diese Datei wird vom KEIL-Tool OH166 erzeugt. Damit dieser zusätzliche Schritt (***siehe Abschnitt 3***) durchgeführt wird, müssen wir unter "***Options - Make die Option Run L166 Linker and OH166***" anklicken. Die von OH166 erzeugte intel-Hex-Datei wird den Namen des Projekts mit der Dateiendung H86 tragen. In unserem Fall ist das `LLICHT.H86`.

Anschließend wählen wir im Menü "***Options - OH166 Object-Hex-Converter***" aus und stellen das Format Intel-HEX-86 ein.

Die Einstellungen sind damit abgeschlossen. Klicken Sie den Menüpunkt "***Project - Make: Build Project***" an und unsere Ausgabedatei `LLICHT.H86` wird gemäß unseren Angaben erstellt.

8.4 Das Flash-Programmiertool FLASHT

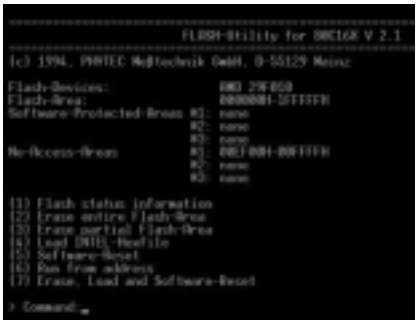
Zum Programmieren des Flash-Speichers benötigen wir ein entsprechendes Programm, das sich auf der Starterkit-CD-ROM im Verzeichnis

`CDROM\STARTKIT\SK_167\FLASH`

befindet. Kopieren Sie dieses Verzeichnis auf Ihre Festplatte. Das Flash-Tool heißt `FLASHT.EXE` und kann über die Batch-Dateien `FLASHT_1.BAT` (für COM1) bzw. `FLASHT_2.BAT` (für COM2) aufgerufen werden. Auf dem Phytec-kitCON-Board muss nun der Bootstrap-Modus aktiviert werden. Haben Sie das kitCON-167-1998, ist der Schalter 1 von SW3 auf ON zu stellen. Auf dem kitCON-167-1997 muss der rote Stecker so gesetzt werden, dass er die Pins 1 2 von Jumper JP2 schließt. Anschließend ist auf dem Board die RESET-Taste zu drücken.

FLASHT lädt über den Bootstrap-Mechanismus ein Monitor-Programm in den RAM-Speicher des Boards, das auch den Programmier-Algorithmus für das Flash enthält.

Das Flash-Tool von Phytec ist ein DOS-Programm, das aber auch in einem Fenster unter Windows 95/98 gestartet werden kann. Es bietet zahlreiche Optionen an, wie Sie auf dem folgenden Bildschirmfoto ersehen können.



Wählen Sie zunächst die Option 7 (***Erase, Load and Software Reset***). Das Flash-Memory wird gelöscht. Danach drücken wir die Taste F2 und geben den Namen der Intel-Hex-Datei ein, die in den Flash-Speicher geladen werden soll. In unserem Beispiel geben wir

`C:\166EVAL\PROJECTS\LLICHT.H86` ein.

Nach dem Laden des Programms führt das Flash-Monitor-Programm einen Software-Reset durch. Das Anwendungsprogramm sollte nun laufen. Da es sich nun resident im Flash befindet, kann es auch nach einer Unterbrechung der Spannungsversorgung jederzeit durch Drücken der Reset-Taste gestartet werden. Das Phytec-Board darf jedoch dazu NICHT in den Bootstrap-Modus geschalten werden. Auf dem kitCON-167-1998 ist der Schalter 1 von SW3 auf OFF zu stellen. Auf dem kitCON-167-1997 ist der (rote) Stecker von JP2 zu entfernen! Andernfalls würde durch ein Reset der Bootstrap-Loader und nicht unser Anwendungsprogramm gestartet werden.

9. STARTUP-Datei für RAM-Konfiguration

Es wird Ihnen wahrscheinlich aufgefallen sein, dass wir für das Arbeiten mit dem Debugger und das Laden unseres Anwendungsprogramms in das RAM keine DAvE-Startup-Datei generiert haben. Das ist auch nicht unbedingt erforderlich, da der KEIL-Linker beim Fehlen einer Startup-Datei im Projekt automatisch eine Default-Startroutine zum Programm linkt. Im Debug-Fenster von dScope sehen Sie nach dem Laden der Object-Datei auf der Adresse 0 (RESET-Vektor) den Aufruf

```
000000: JMPS C_STARTUP
```

Hier wird also gleich zu Beginn der Programmausführung diese Startup-Routine angesprungen. Die Source-Datei zum Default-Startup findet man in `C:\C166EVAL\LIB\START167.A66`

Dieser Source-Code wird allerdings nicht wirklich gelesen, wenn das lauffähige Programm erzeugt wird. Vielmehr ist der Objectcode bereits in der Library enthalten, die vom Linker Ihrem Programm hinzugefügt wird.

Wie bereits erwähnt, müssen Sie sich für das Arbeiten mit dem Monitor nicht unbedingt um eine Startup-Datei kümmern, was für den Anfänger eine zusätzliche Erleichterung darstellt. Fortgeschrittenen Benutzern wird allerdings (auch von KEIL) empfohlen, eine von DAvE generierte STARTUP-Datei hinzuzufügen, wie wir das auch in ***Abschnitt 8*** beschrieben haben. Ein Blick in die Default-Datei `START167.A66` zeigt nämlich, dass die C-Startup-Routine unter anderem für die Zugriffe über den externen Bus 2 Waitstates definiert, obwohl RAM und FLASH des Phytec-Boards auch mit 0 Waitstates funktionieren. Eine DAvE-generierte Startup-Datei nach ***Abschnitt 8*** bringt also geringe Geschwindigkeitsvorteile.

Eine DAvE-Startup-Datei erzeugen Sie, wie bereits in ***Kapitel 8*** beschrieben. Die einzige Änderung betrifft die Optionen für `ADDRSEL1` (auf der Seite `BUSCON1`). Geben Sie auf der entsprechenden Seite ein:

```
Window Size: 1 MB
```

der Monitor erfordert das, auch wenn Sie nur 64 kB physikalisches RAM haben

```
Required Start Address: 0x000
```

das RAM muss in den Adressbereich ab Adresse 0 gemappt werden

Die generierte STARTUP-Datei fügen Sie wie beschrieben Ihrem Projekt hinzu. Auch die Optionen für den Assembler sind, wie in ***8.3*** beschrieben, einzustellen.

Hintergrundinformationen finden Sie wieder in ***[1]***.

10. Schlussbemerkungen

"Aller Anfang ist schwer", schrieb ich zu Beginn dieses Artikels. Ich hoffe aber, dass diese Beschreibung Sie zum ersten erfolgreichen C167-Programm führen konnte. Die Darstellungen wurden bewusst ausführlich und an vielen Stellen mit Hintergrundinformationen und Tipps zum weiteren Selbststudium versehen. "Übung macht den Meister" stelle ich an das Ende des Dokuments - denn es gibt vieles zu erforschen - von den Möglichkeiten der KEIL-Umgebung über DAvE bis hin zum Innenleben des C167.

Viel Erfolg dabei!

Ergänzende und weiterführende Literatur und Web-Sites zum Thema des Artikels

[1] [Generieren des Target-Monitors für das Phytec-kitCON-167-Board und die Keil-Toolkette (Infineon C167-Starterkit), Walter Waldner, 1999
verfügbar über die Homepage des Autors:
http://www.htblmo-klu.ac.at/lernen/

[2] [Umfassende Informationen über die Infineon-Mikrocontroller und Programm-Updates für DAvE finden Sie auf der Web-Seite
http://www.infineon.com/
(Die **Siemens Semiconductor Group** wurde kürzlich in **Infineon** umbenannt)

[3] [Das Internet-Angebot der Firma KEIL finden Sie unter den Adressen *http://www.keil.com/*

[4] [Interaktive Online-Tutorials, insbesondere zu DAvE und CAN finden Sie unter der Adresse
http://www.mfuniversity.com/siemens/homepage.htm

[5] [EXBO – das I/O-Board für Mikrocontroller-Experimente zum Nachbauen. Informationen und Dateien zum Laden auf Ihren PC sind über die Homepage des Autors verfügbar:
http://www.htblmo-klu.ac.at/lernen/

Professionelle Softwareentwicklung für Mikrocontroller

DAvE - Digitaler Applikationsingenieur

Inbetriebnahme des C167CR Starter Kits

Wilhelm Brezovits

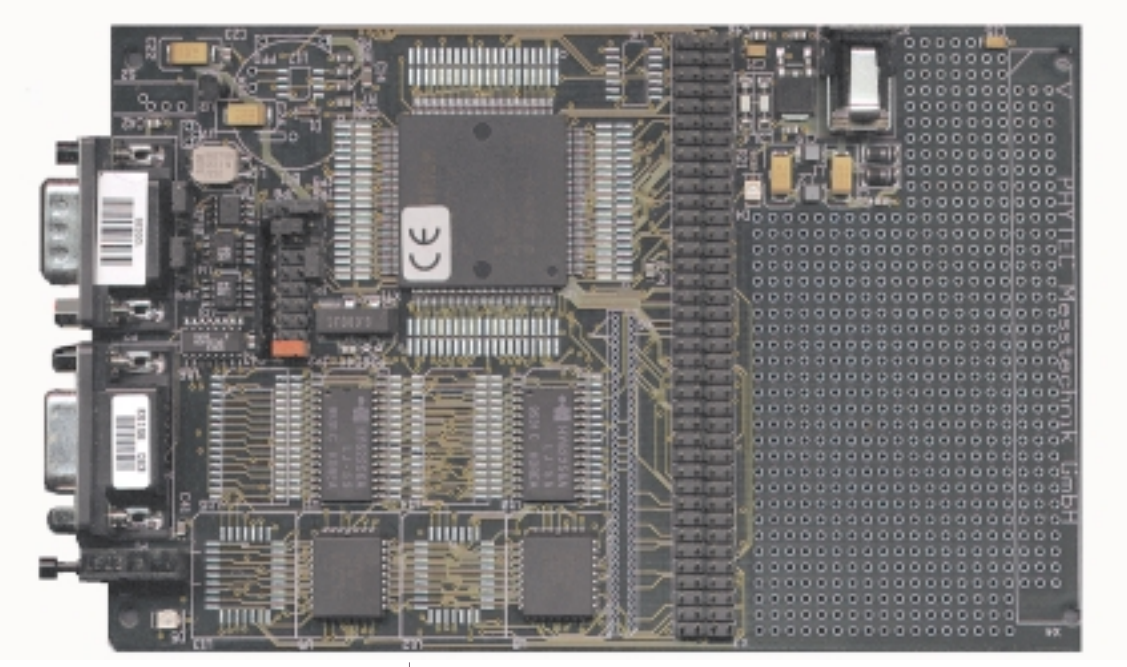
Dieser Artikel soll zu einem Erfolgserlebnis beitragen.

Als Erfolgserlebnis definieren wir eine laufende Applikation am Starterkit nach „Spannung ein".

Alle notwendigen Schritte sollen hier beschrieben werden.

Zuerst ist eine Startup-Datei zu generieren.

Die vom Compilerhersteller mitgelieferte Startup-Datei ist natürlich für die Hardware, auf welcher unsere Applikation laufen soll, anzupassen. Wir generieren diese Datei mit DAvE.



I. DAvE - Digitaler Applikationsingenieur

DAvE ist ein Arbeitskollege. Er ist eine CD-ROM. Er ist ein kostenloses aber wertvolles Support-Tool.

DAvE bietet eine einzigartige, intuitive Benutzeroberfläche, mit der per Mausklick der ausgewählte 8-,16- oder 32-Bit-Mikrocontroller einfach konfiguriert werden kann.

Durch Drücken der rechten Maustaste erhält man jederzeit Hilfe.

So landet man z. B. bei der Eingabe eines Hex-Wertes durch Drücken der rechten Maustaste im Eingabefeld einer Binäreingabe.

Weiters erreicht man durch Drücken der rechten Maustaste in einem umrandeten Feld zusätzliche Hilfestellungen. Handelt es sich z. B. um die Konfiguration eines Bits oder eines Bitfeldes, so erhält man eine Darstellung des zugehörigen Registers, oder man springt auf die richtige Seite im Manual.

DAvE generiert automatisch den richtigen C-Code und ein komplettes Dateisystem, welches mit dem File Viewer betrachtet werden kann.

Zwischen „`// USER CODE BEGIN`" und „`// USER CODE END`" besteht die Möglichkeit, eigenen, applikationsspezifischen Source-Code einzufügen. Dieser bleibt bei einer Änderung der Konfiguration natürlich erhalten!

Hinweis (Vorgehensweise bei der Softwareentwicklung)

Das von DAvE generierte Dateisystem (`START.ASM`, `*.c` und `*.h` Dateien) ist in der Compileroberfläche (Keil µVision oder Tasking EDE) als Projekt anzulegen.

In der Compileroberfläche können weitere Dateien dem Projekt hinzugefügt werden.

Applikationsspezifischer Code darf in der Compileroberfläche mit dem Texteditor in den von DAvE generierten Dateien nur zwischen „// USER CODE BEGIN" und „// USER CODE END" eingefügt werden. Ist die Konfiguration des Mikrocontrollers zu ändern, so sollte man das Projekt in der Compileroberfläche schließen, das Projekt mit DAvE öffnen, die Änderung der Konfiguration des Mikrocontrollers durchführen, das Dateisystem von DAvE neu generieren lassen, das Projekt schließen und in der Compileroberfläche wieder öffnen und dort applikationsspezifisches Programm weiter erstellen, u.s.w..

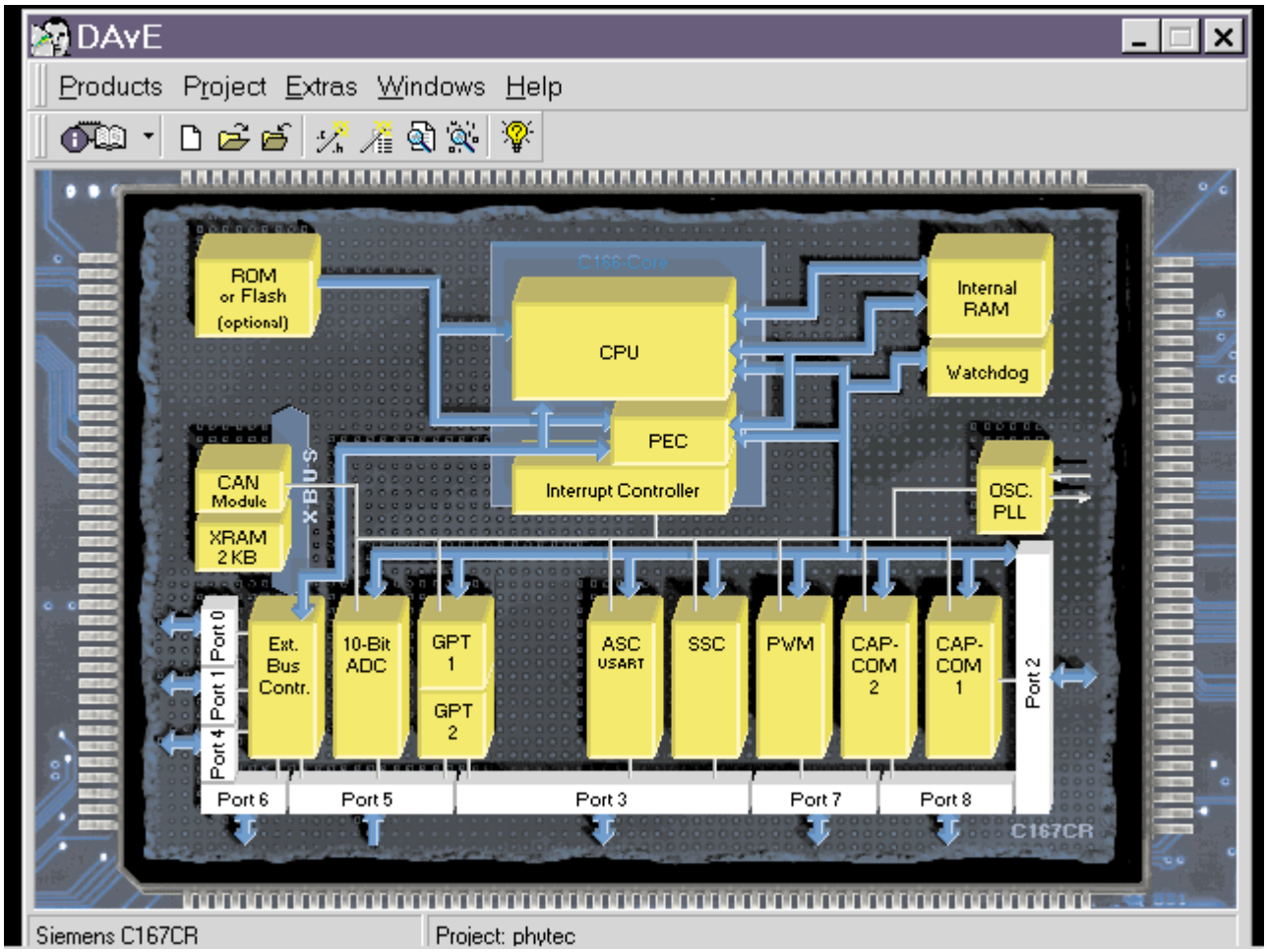
Es ist denkbar, dass Compileroberfläche und DAvE irgendwann ein gemeinsames Tool sein werden.

Hinweis (zum erfolgreichen Arbeiten mit DAvE)

Um erfolgreich mit DAvE zu arbeiten, sollte man die Architektur des Bausteins kennen (User`s Manual), ANSI C Programmierung beherrschen (speziell extern Deklarationen) und mit den mikrocontrollerspezifischen Erweiterungen des Sprachumfanges von ANSI C (zusätzliche Datentypen, Steuerworte und intrinsic/built in-Funktionen) zwecks Zugriff auf die Architektur des Bausteins vertraut sein.

Sonstiges über DAvE

- DAvE ist auch die Dokumentation der Konfiguration des Mikrocontrollers bei Programmstart.
- DAvE hilft bei der Auswahl eines Mikrocontrollers.
- Nach dem Selektieren der Anforderungen liefert DAvE eine Liste aller Mikrocontroller, welche die Anforderungen erfüllen (das günstigste Produkt steht dabei an erster Stelle).
- DAvE sollte regelmäßig über das Internet aktuell gehalten werden. Updates stehen unter *http://www.infineon.com/DAvE.html* zur Verfügung.



II. Starterkits

Starterkits bieten die Möglichkeit, auf günstige Art und Weise einen bestimmten Mikrocontroller und/oder eine Toolkette näher kennenzulernen.

Sie bestehen aus Hard- und Software.

Die Hardware ist ein einschaltfertiges Board (Spannungsstabilisierung, Mikrocontroller - alle Pins zugänglich, Programmspeicher - meist Flash-EEPROM, Datenspeicher - SRAM und Pegeltreiber für RS232). Mit Hilfe mitgelieferter Software kann einfach per Menü im Boot-Mode der Onboard - oder Onchip Programmspeicher programmiert werden. Dazu muss natürlich eine Intel-Hex-Datei vorhanden sein. Diese kann mit der mitgelieferten „Demo"-Software erstellt werden.

Das Wort „Demo" deshalb, da die Compilerpakete eingeschränkt sind (z.B. generierte Codegröße).

Ist man bereits glücklicher Besitzer einer Toolkette (z.B. Keil oder Tasking-Compiler), so bieten die Starterkits die Möglichkeit, rasch ein neues Derivat der Mikrocontrollerfamilie kennenzulernen ohne zuerst eigene Hardware entwickeln zu müssen.

III. Generierung der Startup-Datei mit Hilfe von DAvE für die Keil und Tasking Toolkette

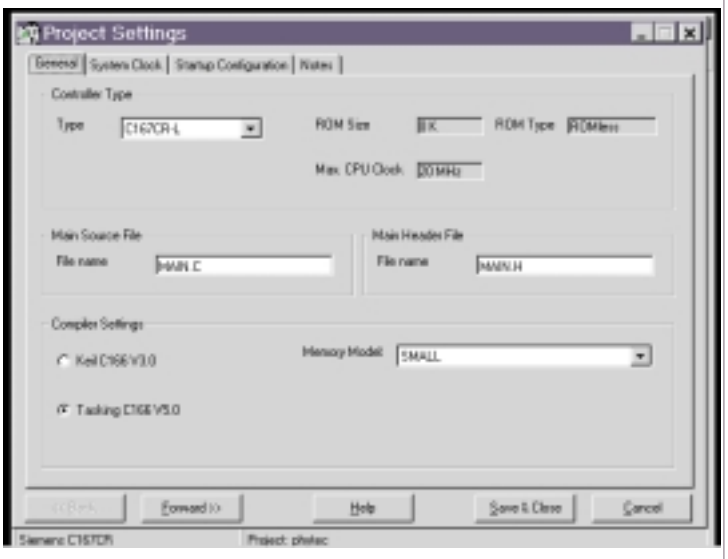
Nach Reset oder „Spannung ein" beginnt der Mikrocontroller mit dem Abarbeiten der internen Resetsequenz. Diese initialisiert alle Special Function Register auf bestimmte Werte. Danach beginnt der Mikroprozessor im Mikrocontroller mit der Abarbeitung des Programms ab Adresse 0.

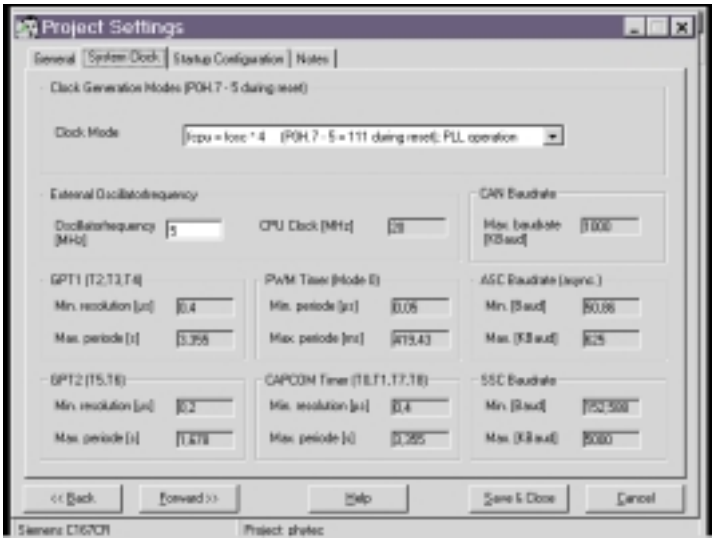
Da bei Adresse 0 die Interrupteinsprungtabelle (Vektortabelle) beginnt befindet sich hier ein Sprungbefehl zum Startup-Code. Der Startup-Code initialisiert den Mikrocontroller (Bus-Timing, Stack-Größe, ...) und schafft die Voraussetzungen für Hochsprachen (Variableninitialisierung, User-Stack, Floating-Point-Stack, ...). Der letzte Befehl des Startup-Code ist ein Sprung zum Hochsprachen-Hauptprogramm `void main (void)`.

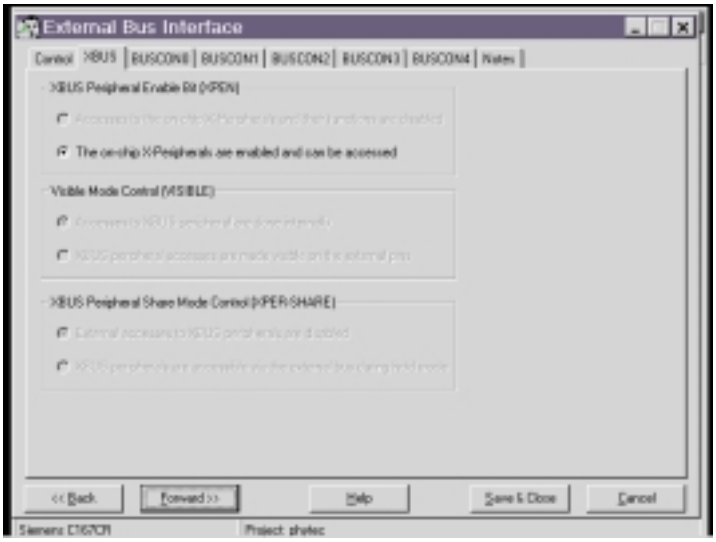
DAvE generiert eine Startup-Datei mit dem Namen `START.ASM`.

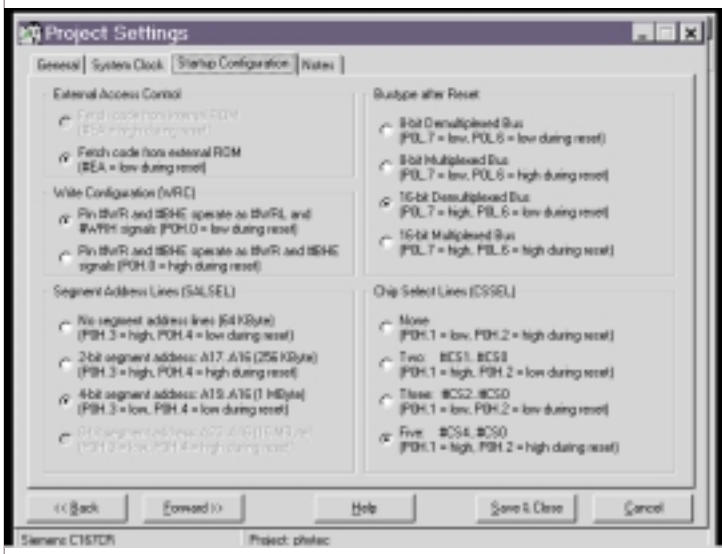
Zuerst wird mit DAvE ein Projekt (phytec) erstellt (***Project, New***). (***Screenshot oben***)

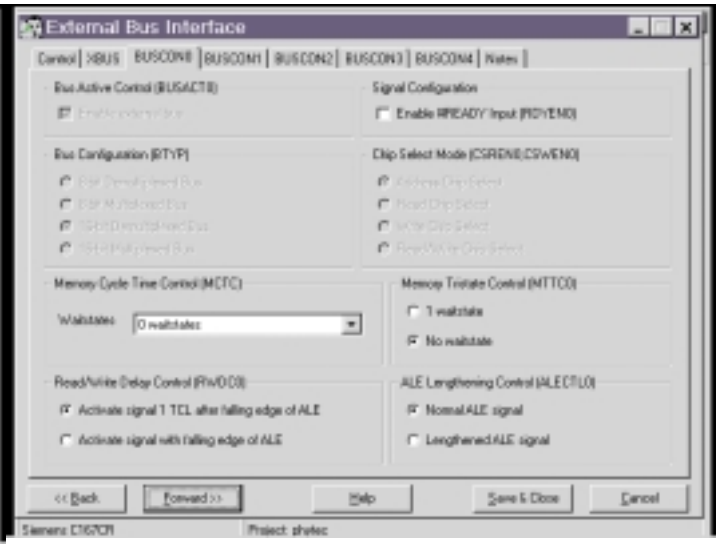
Bei den Project Settings stellen wir die verwendete Toolkette (Keil oder Tasking) ein (***Screenshot unten***). Beide Compiler sind zwar

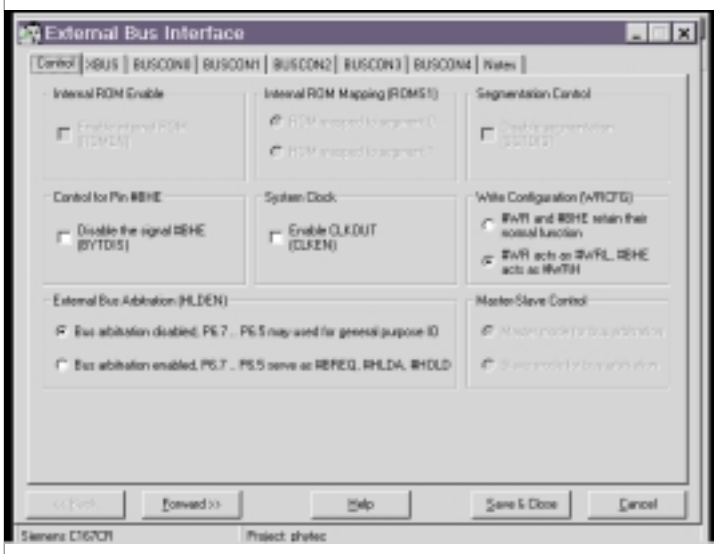


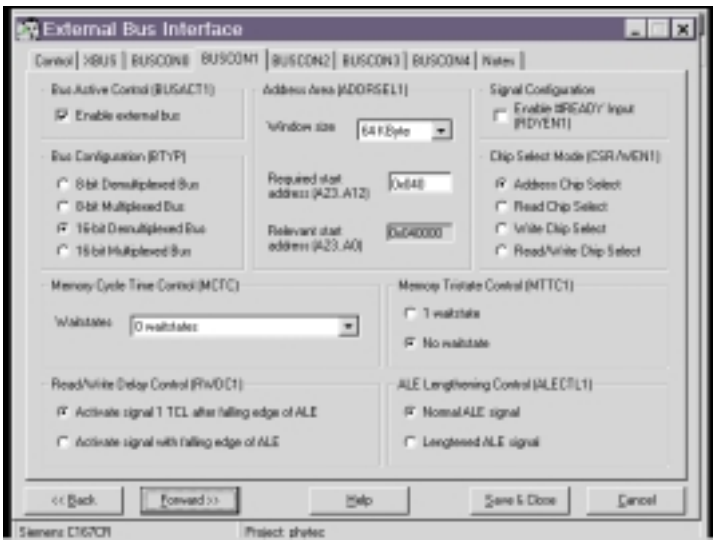












ANSI-C-Compiler unterscheiden sich aber in der Syntax der mikrocontrollerspezifischen Erweiterungen.

Durch das Einstellen der Taktfrequenz erhalten wir eine Übersicht über die zeitlichen Eigenschaften der On-Chip Peripherals.

In Übereinstimmung mit dem Phytec KitCON-167 Hardware-Manual wird die Startup(Einschalt)-Konfiguration und der External Buscontroller für BUSCON0 - CS0 - 256 KByte FLASH Bank1 U8/U9 - 00:0000h bis 03:FFFFh und BUSCON1 - ADDRSEL1 - CS1 - 64 KByte RAM Bank1 U10/U11 - 04:0000h bis 04:FFFFh konfiguriert (Chipselectkonfiguration).

Bei 55ns Speichern gilt: 0 Waitstate, RW-Delay, no Tri-state, short ALE, 16-Bit-Demultiplexed und Umschaltung von A0/BHE# zu Write LOW (WRL#) und Write HIGH (WRH#) - siehe auch Karl-Heinz Mattheis Buch Seite 94 und Kapitel 4 erhältlich als CD-ROM MC-Tools 17 – Arbeiten mit C166-Controllern unter *http://www.otmar-feger.de/*.

Mit *< Project - Generate Code >* initiieren wir die Source-Code-Generierung, die mit *< Project - File Viewer >* betrachtet werden kann.

Hinweis

DAvE generiert für die Keil und für die Tasking Toolkette eine Startupdatei mit dem Namen `START.ASM`.

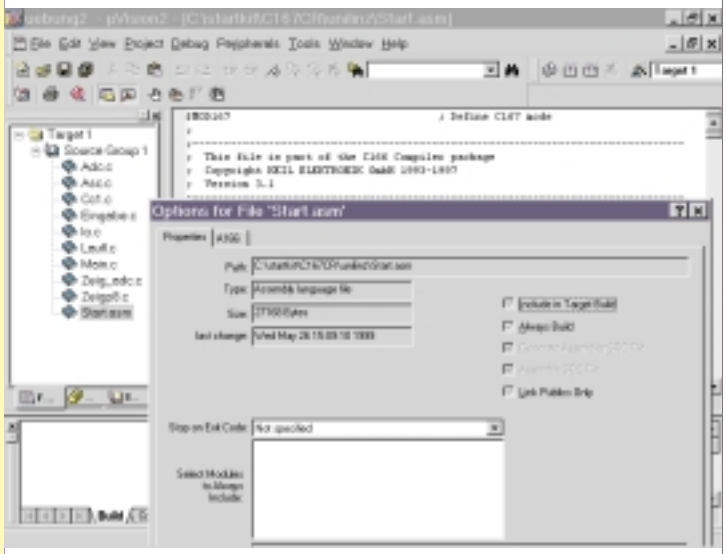
a Hinweis für die Keil-Toolkette

Die Initialisierung des externen Buscontrollers (BUSCON0 bis BUSCON4) wird von DAvE in der Startupdatei `START.ASM` gemacht und mit `*** INSERTED ***` gekennzeichnet.

Die von DAvE generierte Startupdatei `START.ASM` für unser C167CR-Starterkit wird danach folgendermaßen in die Keil Entwicklungsumgebung µVision eingebunden:

µVision1 - Keil Entwicklungsoberfläche, Einbindung der Startupdatei

Wichtig dabei ist, dass „***Include in Link/Lib***“ angekreuzt ist!



µVision2 - Keil Entwicklungsoberfläche, Einbindung der Startupdatei

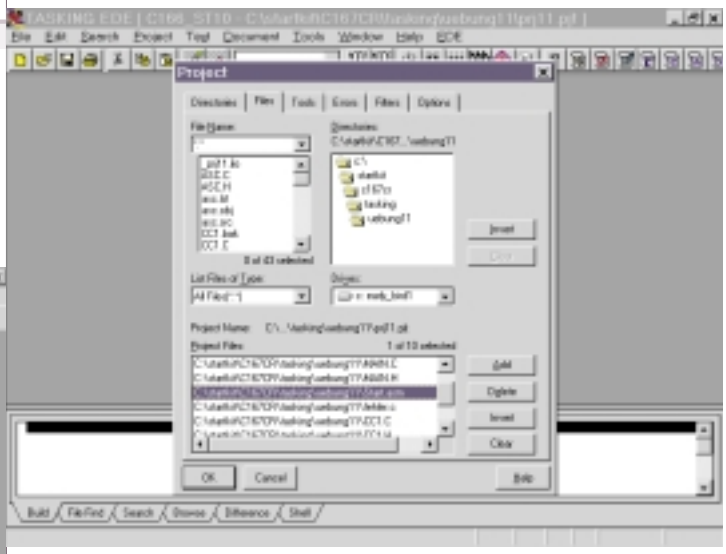
Weitere Hinweise zur Konfiguration der Compileroberfläche findet man am Unterrichtsserver der HTL-Klagenfurt unter *http://www.htblmo-klu.ac.at/lernen/*.

b Hinweis für die Tasking-Toolkette:

Die Initialisierung von BUSCON0 wird von DAvE in der Startup-Datei `START.ASM` gemacht und mit `*** INSERTED ***` gekennzeichnet.

Die Initialisierung von BUSCON1 bis BUSCON4 wir von DAvE in der Datei `main.c` innerhalb der `Project_Init()`-Funktion gemacht.

Die von DAvE generierten Dateien `START.ASM` und `MAIN.C` werden danach folgendermaßen in die Tasking-Entwicklungsumgebung EDE (Embedded Development Environment) eingebunden:



EDE - Tasking Entwicklungsoberfläche, Einbindung der Startupdatei

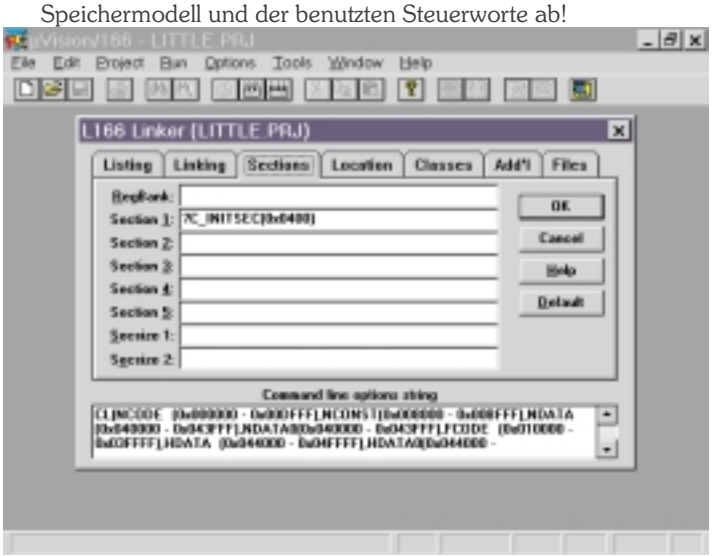
IV. Locater Steuerdatei für das C167CR Starterkit

Die Locater Anweisungen sind notwendig, damit die vom Compiler generierten Segmente (Code, Daten) vom Linker/Locater auf die richtigen physikalischen Speichermedien (ROM, RAM) gemäß der Hardware (C167CR Starterkit) aufgeteilt werden.

a bei der Keil-Toolkette, µVision 1

Die Angaben sind hier natürlich nur als Beispiel gedacht!

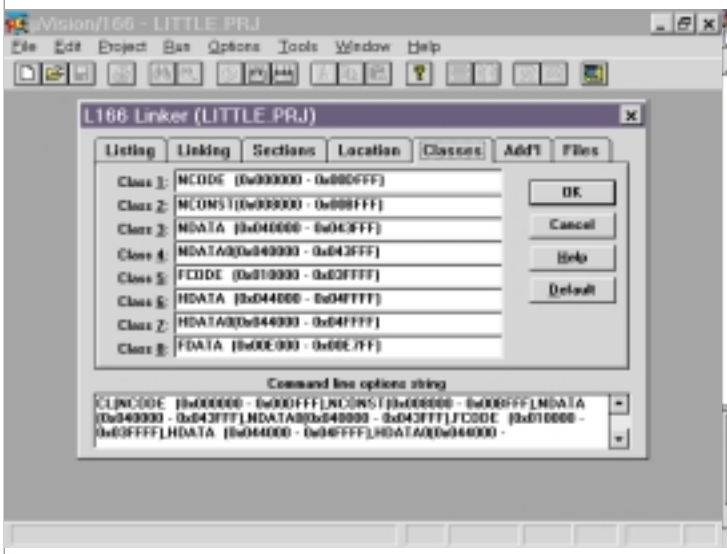
Welche Classes und Sections der Compiler tatsächlich generiert hängt natürlich vom Anwenderprogramm, vom verwendeten Speichermodell und der benutzten Steuerworte ab!



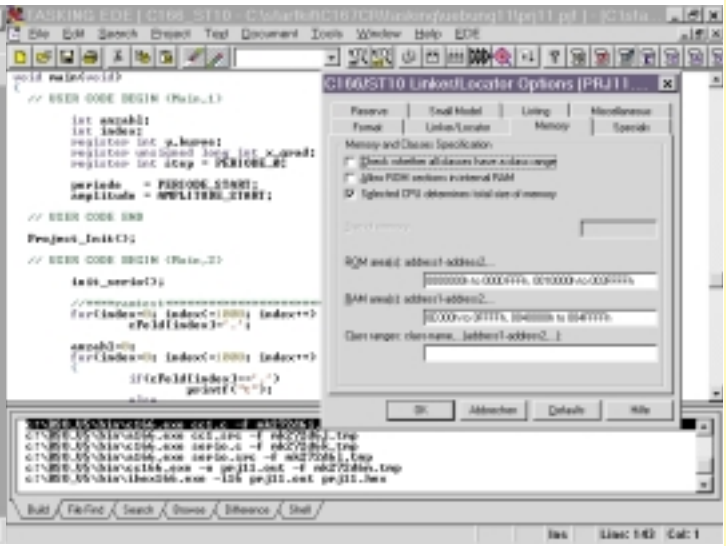
µVision1- Keil Entwicklungsoberfläche, Linker/Locater

b bei der Keil-Toolkette, µVision 2

Bei der µVision2 kann auf einfache Art und Weise angegeben werden, wo sich im Zielsystem

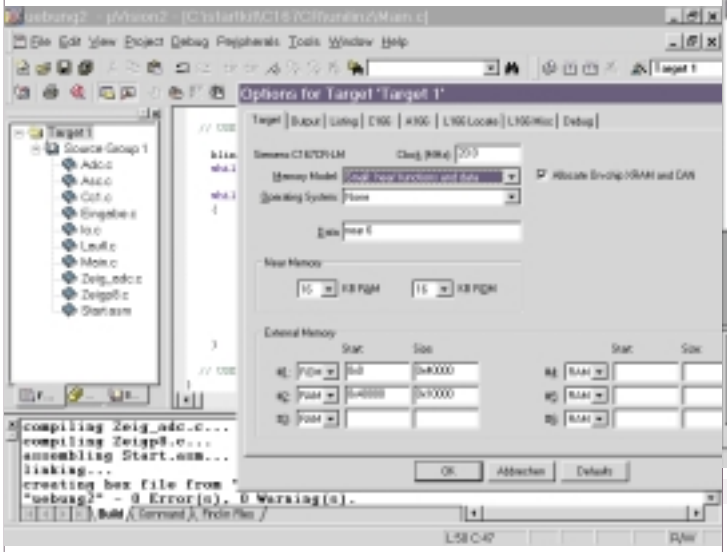


µVision1 Keil Entwicklungsoberfläche, Linker/Locater

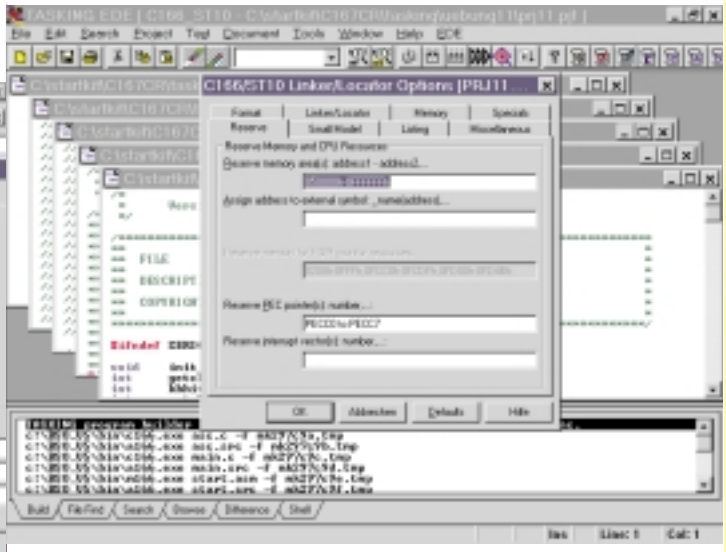


EDE - Tasking Entwicklungsoberfläche, Linker/Locater

Speichertyp „ROM“ und Speichertyp „RAM“ befindet! Diese Angaben sind zur Generierung der hex-Datei (*.h86) notwendig.



µVision2 Keil Entwicklungsoberfläche, Linker/Locater



EDE - Tasking Entwicklungsoberfläche, Linker/Locater

Ach ja, natürlich läuft das hier besprochene Programm „nur“ am Zielsystem „nach Spannung“ ein und am Simulator.

c bei der Tasking-Toolkette

Bei der EDE kann auf einfache Art und Weise angegeben werden, wo sich im Zielsystem Speichertyp „ROM“ und Speichertyp „RAM“ befindet! Diese Angaben sind zur Generierung der Hex-Datei notwendig.

V. Schlussbemerkungen

Ich hoffe, dass diese Darstellungen hilfreich sind.

Feedback zum Artikel bitte an *wilhelm.brezovits@siemens.at* .

Warum nicht mit dem Debugger?

Der Debugger muss die Möglichkeit haben, das Programm zu ändern. Definiert man zum Beispiel einen Breakpoint, so wird der Code an dieser Stelle durch einen Sprung in den Monitor (der mit dem Debugger kommuniziert) ersetzt, damit der Debugger weiss, dass diese Stelle erreicht wurde.

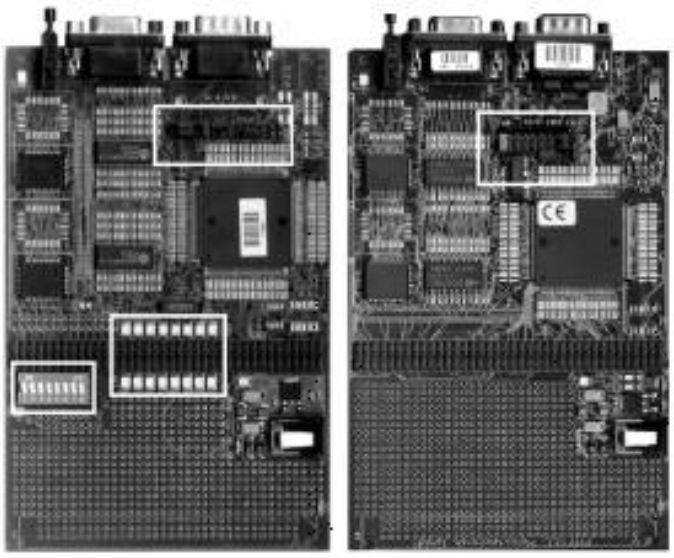
Das bedeutet allerdings: Das Programm steht im „RAM“ und nicht im Zielspeichertyp „ROM“.

Das RAM beginnt im Debuggerbetrieb auf der Adresse 00:0000 H (statt 04:0000 H) – also andere START.ASM und andere Linker/Locater-Steuerung!

Monitor für Mikrocontroller

Generieren des Target-Monitors für das Phytec-kitCON-167-Board und die KEIL-Toolkette (Infineon C167CR-Starterkit)

Walter Waldner



1. Einleitung

Die KEIL-Toolkette für die Infineon 16-Bit Mikrocontroller-Familie erlaubt die Software-Entwicklung mit dem Assembler bzw. dem ANSI-C-Compiler und das komfortable Laden, Simulieren und Debuggen der Anwenderprogramme (**dScope**). Eine ausführliche Beschreibung der KEIL-Entwicklungsumgebung (**µVision**) für das Phytec-kitCON-167-Board, das dem Infineon C167CR-Starterkit beiliegt, finden Sie in ***[1]***.

1.1 Beachten Sie die Starterkit-Version

Vom Infineon C167CR-Starterkit gibt es inzwischen zwei Versionen, die sich geringfügig unterscheiden. Die Ausführungen dieses Artikels gelten mit kleinen Änderungen für beide Starterkit-Ausgaben.

So erkennen Sie Ihr Starterkit:

Starterkit 1997

Phytec-Hardware Manual
Version 1.0 6/1997
Starterkit-CD-ROM: Edition 2.1

Starterkit 1998

Phytec-Hardware Manual
Version 2.0 7/1998
Starterkit-CD-ROM: Edition 3.2

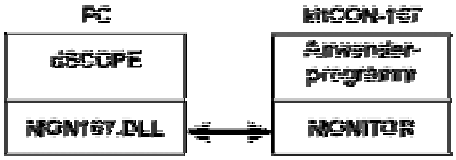
Die Unterschiede zwischen den beiden Starterkit-Ausgaben betreffen hauptsächlich das beigefügte **Phytec-kitCON-167-Board**. Das neuere Board, im obigen Bild links zu sehen, (wir bezeichnen es in der Folge als kitCON-167-1998) enthält zusätzlich 16 Leuchtdioden, die mit dem Port 2 des Mikrocontrollers verbunden sind. Außerdem wurde die Jumperbelegung geändert. So ist etwa der Bootstrap-Modus nicht wie am älteren kitCON-167-1997 über einen Jumper, sondern über Schalter 1 des neuen Schalterblocks DIP-Switch S3 einzustellen.

In diesem Artikel wird vor allem das KEIL-Entwicklungstool dScope (ein MS-Windows-Programm für den PC) beschrieben. Die Ausführungen beziehen sich stets auf den Mikrocontroller C167CR bzw. das entsprechende Infineon C167CR-Starterkit. Falls es Unterschiede zwischen den beiden Starterkit-Versionen gibt, sind diese explizit angeführt.

dScope vereint zwei Funktionen unter einer einheitlichen Oberfläche:

- ***C167-Hardware-Simulator***
  dScope bildet die Funktionen des Mikrocontrollers durch Software nach und erlaubt so die Simulation von C167-Anwenderprogrammen
- ***Debugger (Target-Simulator)***
  Das Anwenderprogramm läuft auf der Ziel-Hardware (in diesem Fall dem Phytec kitCON-167-Board). dScope kommuniziert mit der Ziel-Hardware und ermöglicht verschiedene Debug-Funktionen. KEIL nennt den Debugger auch **tScope**. In diesem Artikel werden wir für den Debugger aber stets die Bezeichnung dScope verwenden.
  Der Debugger (tScope) und der Hardware-Simulator (dScope) bieten dieselbe Benutzeroberfläche.

Für das Arbeiten mit dem dScope-Debugger muss in den Speicher (RAM bzw. FLASH Memory) des Phytec kitCON-167-Boards ein Monitor-Programm geladen werden, das über eine spezielle DLL und über die serielle Schnittstelle mit dem dScope-Programm am PC kommuniziert.



Die dScope-Oberfläche und das Monitorprogramm ermöglichen unter anderem folgende Funktionen:

- Laden von Anwendungsprogrammen in das SRAM
- Starten / Unterbrechen von Programmen
- Schrittweise Programmausführung auf Source-Code-Ebene
- Setzen / Löschen von Breakpoints
- Betrachten der CPU-internen Register und SFRs
- Darstellung der Onchip-Peripherals des C167
- Anzeige von Unterprogramm- und Interrupt-Service-Routinen-Aufrufen
- Anzeige / Verändern von Speicherinhalten und Variablen
- Disassemblieren von Code-Sequenzen
- Performance-Analyse

Für das Zusammenspiel von dScope und dem Monitorprogramm gibt es eine Reihe von Konfigurationsmöglichkeiten, die in diesem Artikel beschrieben werden. Die Optionen sind im wesentlichen:

- MONITOR ins SRAM oder FLASH laden
- Kommunikation über die integrierte serielle Schnittstelle ASC0 des C167 oder über eine software-simulierte serielle Schnittstelle

Auch können die Baudraten für die Kommunikation unterschiedlich eingestellt werden. Die Source-Dateien für das Erstellen indi-

REKIRSCH - 1

viduell konfigurierter Monitor-Dateien sind Teil der Demo-Version der Keil-Toolkette, die sich auf der Starterkit-CD-ROM befindet.

Hinweis: Im Debug-Modus muss sich das Anwender-Programm immer im SRAM befinden, da beispielsweise beim Setzen von Breakpoints Anweisungen in das Programm eingefügt werden.

2. Generieren des Monitor-Programms

Wir nehmen an, dass die KEIL-Toolkette in das Verzeichnis

```
C:\C166EVAL
```

installiert wurde. Im Unterverzeichnis

```
C:\C166EVAL\MON166
```

befinden sich die Source-Codes zum Erzeugen der Dateien BOOT und MONITOR. BOOT wird vom internen Bootstrap-Loader des C167 geladen und gestartet und wird benötigt, um das Monitor-Programm in das SRAM des kitCON-Boards zu bringen. Zum besseren Verständnis der folgenden Ausführungen sollten Sie zunächst die Datei README.TXT im obigen Verzeichnis lesen.

Für das KitCON-Board sind die Source-Dateien

```
CONFPHY7.INC
BOOTPHY7.A66
INSTPHY7.A66
```

relevant. In der Datei CONFPHY7.INC werden Parameter für die Konfiguration der SYSCON- und der BUSCON-Register definiert. Diese Datei wird beim Assemblieren von BOOTPHY7.A66 und INSTPHY7.A66 gelesen.

Obwohl die README.TXT-Datei im MON166-Verzeichnis behauptet, dass die Dateien CONFPHY7.INC, BOOTPHY7.A66 und INSTPHY7.A66 für das kitCON-167-Board adaptiert wurden, sind dort einige Parameter falsch gesetzt.

2.1 Anpassen der Source-Dateien für das kitCON167-Board

In der Datei CONFPHY7.INC sind folgende Zeilen zu ändern:

```
_MCTC0    EQU    0  ; Memory wait states is 0 (MCTC0 = 0FH).
 RWDC0    EQU    0  ; 0 = Delay Time     0.5 States (Reset Value)
_MTTC0    EQU    1  ; 1 = No Delay Time  0   States
_XRAMEN   EQU    1  ; 1 = on-chip XRAM is accessed
```

In der Datei BOOTPHY7.A66 sind die folgenden Veränderungen durchzuführen:

Die Zeile

```
BFLDL SYSCON,#080H,#SYS_L
```

ist zu ersetzen durch

```
$IF (WRCFG_ENABLE == 1)
     BFLDL SYSCON,#084H,#SYS L
$ELSE
     BFLDL  SYSCON,#004H,#SYS L
$ENDIF
```

Starter-Kit 1998: Auf der diesem Starterkit beiliegenden CD-ROM (Edition 3.2) sind diese Änderungen in der Datei CONFPHY7.INC bereits vorgenommen worden.

Außerdem können die Zeilen

```
MOV  ADDRSEL2,#1008H  ; 1MB RAM BANK2 (10:0000H - 1F:FFFFH)
MOV  BUSCON2,BUSCON0
```

gelöscht werden (bzw. durch ein Semikolon in der ersten Spalte als Kommentar markiert werden), da das kitCON-167-Board nur das Flash-Memory (wird durch das Chip-select-Signal CS0 [BUSCON0] aktiviert) und den RAM-Speicher (wird durch CS1 [BUSCON1, ADDRSEL1] aktiviert) als externe Komponenten enthält. Es spricht allerdings nichts gegen das Verbleiben der Zeilen in der Source-Datei, außer Sie benötigen CS2 in Ihre Applikation zum Ansteuern externer Komponenten. CS2 ist auf dem kitCON-Board selbst nicht in Verwendung.

In der Datei INSTPHY7.A66 sind folgende Einträge zu verändern:

Die Zeile

```
BFLDL SYSCON,#080H,#SYS_L
```

ist durch

```
$IF (WRCFG ENABLE == 1)
        BFLDL   SYSCON,#084H,#SYS_L
$ELSE
        BFLDL   SYSCON,#004H,#SYS L
$ENDIF
```

zu ersetzen.

Für die Zeilen

```
MOV  ADDRSEL2,#1008H
MOV  BUSCON2,BUSCON0
```

gelten die oben gemachten Aussagen.

Schließlich sind die Definitionen zur simulierten seriellen Schnittstelle falsch. Die Zeilen

```
T_LINE          BIT     P2.0  ; Transmit Data Line TxD
T_OUT           BIT     DP2.0 ; Port direction register for TxD
R_LINE          BIT     P2.1  ; Receive Data Line RxD
R_IN            BIT     DP2.1 ; Port direction register for RxD
```

müssen ersetzt werden. Hier sind Unterschiede für die beiden kitCON-Varianten zu beachten.

Wenn Sie das ältere kitCON167-Board besitzen (Ausgabe 1997), ist einzugeben:

```
T_LINE          BIT     P3.0  ; Transmit Data Line TxD
T OUT           BIT     DP3.0 ; Port direction register for TxD
R_LINE          BIT     P3.2  ; Receive Data Line RxD
R_IN            BIT     DP3.2 ; Port direction register for RxD
```

Für das neuere kitCON167-Board (Ausgabe 1998 mit den LEDs an Port 2) ist einzugeben:

```
T LINE          BIT     P3.9  ; Transmit Data Line TxD
T OUT           BIT     DP3.9 ; Port direction register for TxD
R LINE          BIT     P3.8  ; Receive Data Line RxD
R_IN            BIT     DP3.8 ; Port direction register for RxD
```

2.2 Varianten für die Verwendung des Monitor-Programms

Für das Arbeiten mit dScope/Monitor kann der Anwender zwischen folgenden Varianten wählen:

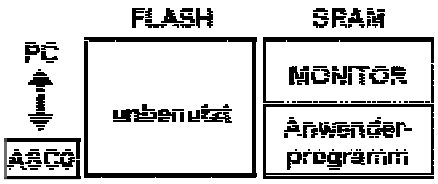
1 Monitor wird in das SRAM geladen
Die Kommunikation zwischen dScope und dem Monitor läuft über die serielle Schnittstelle des Phytec-Boards (DB9-Stecker P1)
Der Monitor wird über den C167-Bootstrap-Mechanismus und mit Hilfe des Programms BOOT geladen

2 Der Monitor wird mittels FLASHT.EXE in das Flash-Memory des Phytec-Boards programmiert. Die Kommunikation läuft über die serielle Schnittstelle des Boards (DB9-Stecker P1)

3 Der Monitor wird mittels FLASHT.EXE in das Flash-Memory des Phytec-Boards programmiert. Die Monitor-dScope-Kommunikation läuft über die simulierte serielle Schnittstelle (Debug-Interface DB9-Buchse P2)

Für die Variante 1 müssen die Absolute-Object-Dateien BOOT und MONITOR erzeugt werden, für die beiden anderen Varianten benötigt man die Intel-Hex-Datei MONITOR.H86, die dann mit dem Programmiertool FLASHT.EXE in das Flash-Memory programmiert wird. Im folgenden wird beschrieben, wie Sie diese Dateien erzeugen können. In jedem Fall sind zunächst die vorher beschriebenen Änderungen in den Dateien CONFPHY7.INC, BOOTPHY7.A66 und INSTPHY7.A66 erforderlich.

2.2.1 Variante 1



In dieser Variante wird nur das SRAM des kitCON-167-Boards und die Bootstrap-Sequenz des Mikrocontrollers verwendet. Das Bootstrap-Programm befindet sich im BOOT-ROM des C167 und wird aktiviert, wenn der BOOT-Mode gewählt wurde. Am kitCON-167-1997 muss dazu der Jumper JP2 (1+2) mit dem [roten] Stecker geschlossen werden und die RESET-Taste am Phytec-Board gedrückt werden.

Am kitCON-167-1998 muss der Schalter 1 auf SW3 auf ON geschalten werden und die RESET-Taste gedrückt werden. Der Ablauf des Boot-Vorgangs ist wie folgt:

Schritt	Aktion
1	Die C167-Bootstrap-Sequenz wartet auf ein 0-Byte vom PC und sendet ID-Byte zum PC-Programm (in unserem Fall ist das dScope)
2	Die C167-Bootstrap-Sequenz lädt ein 32 Byte großes Preload-Programm ins RAM (das sind die ersten 32 Bytes der Datei `BOOT`)
3	Das Preload-Programm wird aktiviert und lädt den Rest des Programms `BOOT` ins RAM
4	`BOOT` wird aktiviert und lädt den `MONITOR` ins RAM
5	Der `MONITOR` wird aktiviert und kommuniziert mit dScope. dScope kann verschiedene Kommandos an den `MONITOR` senden (z.B. Laden des Anwenderprogramms ins RAM) bzw. Daten vom `MONITOR` empfangen und anzeigen

Die Kommunikation zwischen dScope und dem Monitor läuft in Variante 1 über die integrierte serielle Schnittstelle `ASC0` des C167-Controllers. Nachfolgend wird beschrieben, wie Sie die Dateien `BOOT` und `MONITOR` generieren können.

Öffnen Sie unter Windows das DOS-Fenster und wechseln Sie in das Verzeichnis

```
C:\C166EVAL\MON166
```

Nun geben Sie den Befehl

```
INSTALL PHY7 0 FEA FEC BOOTSTRAP
```

ein. Die Batch-Datei `INSTALL` erzeugt die Dateien `BOOT` und `MONITOR` (absolute Objekt-Dateien) und kopiert sie auch gleich in das richtige Verzeichnis `C:\C166EVAL\BIN`

Beachten Sie bitte, dass der C167 in den Bootstrap-Modus versetzt werden muss. Dazu wird auf dem Board - Ausgabe 1997 - der rote Jumper (JP2 1+2) gesetzt und die gedrückt. Auf dem Phytec-Board – Ausgabe 1998 – ist der Schalter 1 auf SW3 zu schließen und RESET zu drücken.

Außerdem muss in der µVision-Oberfläche unter "***Options - L166 Linker -Reserve 1***" eingetragen sein:

```
O8h-OBh, OACh-OAFh
```

Das Monitor-Programm verwendet den NMI (non maskable interrupt) als Breakpoint-Trap und den Empfangs-Interrupt der seriellen Schnittstelle `ASC0`. Daher müssen die oben angeführten Adressen in der Interrupt-Vektortabelle vor dem Überschreiben geschützt werden.

Adressen:	verwendet von:
`08h-0Bh`	NMI
`0ACh-0AFh`	ASC0

ACHTUNG: Der Monitor verwendet **Interrupt-Level 15**, **Group-Level 0** für den `ASC0`-Receive-Interrupt (siehe `README.TXT` im MON166-Verzeichnis). Diese Kombination darf im Anwenderprogramm **NICHT** verwendet werden.

2.2.2 Variante 2

In diesem Fall wird der Monitor `MONITOR.H86` in das Onboard-Flash-EEPROM programmiert und muss nach Spannungsunterbrechungen des Boards nicht neu geladen werden. Das Anwenderprogramm wird vom Monitor in das SRAM geladen.

Auch in dieser Variante kommunizieren dSCOPE und der Monitor über die integrierte `ASC0`-Schnittstelle des C167-Mikrocontrollers. Zunächst beschreiben wir das Erzeugen der Datei `MONITOR.H86`, die mit dem Programmiertool `FLASHT.EXE` in das Flash-Memory programmiert werden kann.

Öffnen Sie unter Windows das DOS-Fenster und wechseln Sie in das Verzeichnis

```
C:\C166EVAL\MON166
```

Nun geben Sie den Befehl

```
INSTALL PHY7 0 FFE 2000
```

ein. Anschließend müssen Sie noch

```
..\BIN\OH166 MONITOR H167 OFFSET (-0x200000)
```

eingeben. `OH166.EXE` konvertiert die "*absolute object*"-Datei `MONITOR` in eine intel-Hex-86-Datei `MONITOR.H86`.

Nun können wir das dadurch erzeugte File `MONITOR.H86` in das Flash-Memory laden.

Dazu verwenden wir das Programm `FLASHT.EXE`. Auf der Starterkit-CD-ROM finden wir es im Verzeichnis

```
X:\CDROM\STARTKIT\SK_167\FLASH
```

Zum Flash-Programmieren muss das Phytec-Board und der PC wird mit dem seriellen Kabel verbunden und der C167 in den **Bootstrap-Modus** gebracht werden. Für das **kitCON-Board 1997** ist der rote Stecker auf Jumper JP2 1+2 zu setzen und die RESET-Taste zu drücken. Für das **neuere Board 1998** ist der Schalter 1 von SW3 zu schließen und RESET zu drücken. Das Flash-Programmiertool `FLASHT.EXE` kann über die Batch-Dateien `FLASHT_1.BAT` (für COM1) bzw. `FLASHT_2.BAT` (für COM2) oder auch direkt durch

```
flasht 1 br(9600) bzw.
flasht 2 br(9600)
```

gestartet werden. Die Parameter 1 bzw. 2 geben die Schnittstelle am PC an (1=COM1, 2=COM2), die Zahl nach `br` die Baudrate der Kommunikation.

Nach dem Laden der Programmiersoftware wird die Option 7 (Löschen des Flash-Memory) gewählt. Anschließend muss F2 gedrückt und der Pfad der Intel-Hex-Datei angegeben werden. In unserem Fall geben wir ein:

```
C:\C166EVAL\MON166\MONITOR.H86
```

Ist der Ladevorgang abgeschlossen, wird FLASHT mit F1 beendet.

kitCON167-Board 1997

Entfernen Sie nun den roten Stecker von Jumper JP2 (1+2) und drücken Sie RESET.

kitCON167-Board 1998

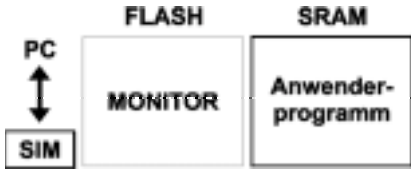
Öffnen Sie Schalter 1 auf SW3 und drücken Sie RESET.

Wie unter Variante 1 beschrieben, sind in der KEIL-Umgebung µVision unter den Linker-Optionen die Adressen

```
O8h-OBh, OACh-OAFh
```

zu reservieren und die Verwendung von Interrupt-Level 15, Group-Level 0 im Anwenderprogramm verboten.

2.2.3 Variante 3



Variante 3 ist eine interessante Möglichkeit mit dem Monitor zu arbeiten, die vor allem für fortgeschrittene C167-Anwender zu empfehlen ist. In diesem Fall wird das Monitor-Programm so generiert, dass es über eine software-simulierte serielle Schnittstelle mit dem PC kommuniziert.

Die vorher beschriebenen Varianten benutzen die C167-interne asynchrone serielle Schnittstelle ASC0 zur Kommunikation zwischen Monitor und PC. Damit ist es aber nicht möglich, Anwenderprogramme zu testen, die selbst die asynchrone serielle Schnittstelle ASC0 des C167 benötigen.

Unter "simulierte serielle Schnittstelle" versteht man, dass das Monitorprogramm über zwei beliebige I/O-Pins des C167-Mikrocontrollers das asynchrone Protokoll der ASC0-Schnittstelle durch Software nachbildet. Ein Pin wird als Sendeleitung (TxD), der andere als Empfangsleitung (RxD) verwendet. Nun waren die Entwickler des Phytec-Boards so freundlich, die Pins 0 bzw. 9 (TxD) und 2 bzw. 8 (RxD) des Ports 3 mit dem MAX232-Baustein zu verbinden (Pin 0 und 2 gelten für das ältere kitCON167-Board-1997, Pin 9 und 8 gelten für das neuere kitCON167-Board-1998). Der MAX232 kann für zwei Kanäle CMOS-Pegel in die von der RS232C-Schnittstelle erforderlichen +10/-10 Volt-Pegel bzw. umgekehrt wandeln. Da nur ein Kanal des MAX232 für die C167-interne serielle Schnittstelle verwendet wird, ist der zweite Kanal für die software-simulierte serielle Kommunikation frei.

Die Buchse P2 des Phytec-Boards kann wahlweise als CAN-Interface oder als Debug-Schnittstelle für die simulierte serielle Kommunikation verwendet werden. Die Auswahl geschieht beim kitCON 1997 über insgesamt drei Jumper und über vier Jumper beim kitCON 1998. Damit man die simulierte serielle Schnittstelle verwenden kann, müssen folgende Jumper-Einstellungen vorgenommen werden:

Einstellungen für kitCON 1998	
JP2	2+3
JP8	2+3
JP9	2+3
JP10	2+3

Einstellungen für kitCON 1997	
JP8	schließen
JP9	2+3
JP10	2+3

Dadurch verbinden wir die MAX232-Leitungen mit der P2-Buchse. Die Pin-Belegungen der Jumper können Sie im Phytec-kitCON167-Hardware-Manual ab der Seite 23 nachlesen. (**Hinweis**: Die Beschreibung im kitCON167-1998-Hardware-Manual Version 2.0 7/1998 ist falsch. Es wird empfohlen, den Schaltplan am Ende des Manuals (Sheet 1 of 4) zu betrachten)

Nun ergibt sich nur noch das Problem, dass P2 im Gegensatz zu P1 ein Stecker (männlich) ist und somit das Kabel aus dem Starterkit nicht passt. Besorgen Sie sich entweder einen sogenannten "Gender-Changer" (Buchse-Buchse / weiblich-weiblich) oder löten Sie sich ein entsprechendes Kabel, was sehr einfach ist, da nur drei Leitungen erforderlich sind. Dazu besorgen Sie sich zwei DB9-Buchsen und verbinden

- Pin 2 mit Pin 2 (TxD)
- Pin 3 mit Pin 3 (RxD)
- Pin 5 mit Pin 5 (Ground)

Das Erzeugen der Datei MONITOR.H86 für die Kommunikation über die simulierte serielle Schnittstelle geschieht durch

```
INSTALL PHY7 2 FFE 2000
```

Damit wird die absolute Objekt-Datei MONITOR generiert. Anschließend müssen Sie wie bei Variante 2

```
..\BIN\OH166 MONITOR H167 OFFSET (-0x200000)
```

eingeben. OH166 erzeugt die Intel-HEX-86-Datei MONITOR.H86. Zum Laden der Datei MONITOR.H86 in das Flash-Memory ist wie unter Variante 2 beschrieben vorzugehen. Beachten Sie, dass der C167 wiederum in den **Bootstrap-Modus** geschalten werden muss (kitCON167-1997: Jumper JP2 1+2 schließen und RESET drücken, kitCON167-1998: Schalter 1 auf SW3 schließen und RESET drücken). Für die FLASH-Programmierung wird der PC über das Starterkit-Kabel mit der seriellen Schnittstelle ASC0 des C167 verbunden (**Buchse P1**!), da das Flash-Tool FLASHT.EXE den Bootstrap-Mechanismus und damit die integrierte serielle Schnittstelle ASC0 des C167 erfordert.

Nach dem Programmieren des Flash-EEPROMs wird auf dem 1997er kitCON167-Board der rote Stecker (JP2 1+2) entfernt, auf der 1998er-Ausgabe des kitCON-Boards wird Schalter 1 auf SW3 geöffnet. In jedem Fall ist anschließend die RESET-Taste zu drücken. Der PC wird nun mit dem Phytec-Board über die Debug-Buchse P2 verbunden (Gender-Changer oder selbstgelötetes Kabel verwenden). Die interne asynchrone serielle Schnittstelle ASC0 des C167 (über Stecker P1) steht dem Anwenderprogramm voll zur Verfügung und kann wahlweise mit einer freien Schnittstelle des PCs oder anderer Zielhardware verbunden werden.

Die Variante 3 erfordert unter "*Options - L166 Linker - Reserve 1*" in der µVision-Umgebung nur den Eintrag

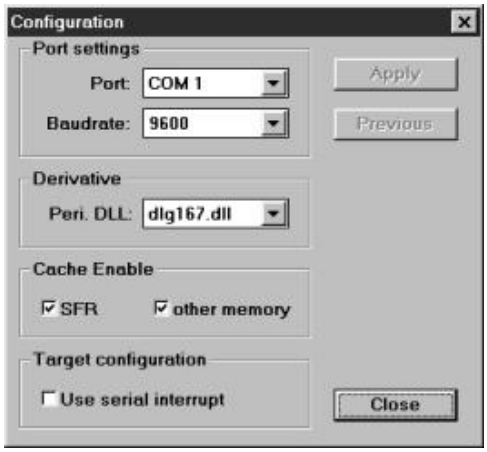
```
O8h-OBh
```

da in diesem Fall der ASC0-Receive-Interrupt vom Monitor nicht benutzt wird, sondern dem Anwenderprogramm zur Verfügung steht. Interrupt-Level 15 / Group-Level 0 ist frei zur Verwendung im Anwenderprogramm.

Im Programm dScope ist unter

"*Peripherials - Config. - Target Configuration*"

die Option "*use serial interrupt*" zu deaktivieren!



3. Einstellen der Baudrate

Wenn Sie mit der **Variante 1** arbeiten, erkennt der interne Bootstrap-Loader-Code des C167 die in dScope eingestellte Baudrate automatisch anhand eines Nullbytes, das die Kommunikation zwischen Bootstrap-Loader und dScope einleitet (siehe Kapitel "Bootstrap Loader" im C167-User Manual).

Für die **Variante 2** kann die Geschwindigkeit der serielle Kommunikation zwischen dem Target-Monitor (Mikrocontroller-Board) und der dScope-Oberfläche (am PC) in der Datei `INSTPHY7.A66` eingestellt werden. Suchen Sie in der Datei `INSTPHY7.A66` den Abschnitt "`Initialization of Serial Interface 0`". Dort ist eine der `MOV`-Anweisungen

```
;    MOV  S0BG ,#0040H   ; SET BAUDRATE TO 9600 BAUD @ 40MHz
;    MOV  S0BG ,#0020H   ; SET BAUDRATE TO 19200 BAUD @ 40MHz
;    MOV  S0BG ,#000FH   ; SET BAUDRATE TO 38400 BAUD @ 40MHz
     MOV  S0BG ,#000AH   ; SET BAUDRATE TO 57600 BAUD @ 40MHz
```

zu aktivieren. Das führende Semikolon deaktiviert eine Zeile als Kommentar, der beim Assemblieren überlesen wird. Um etwa 57600 Baud (die maximale Geschwindigkeit) einzustellen, löschen Sie das Seminkolon in der Zeile

```
MOV  S0BG ,#000AH
```

Alle anderen Optionen müssen durch ein Semikolon am Beginn der Zeile deaktiviert sein.

Für die **Variante 3** (simulierte serielle Schnittstelle) suchen Sie in der Datei `INSTPHY7.A66` die Stelle, die mit "`Initialization of simulated Serial Interface 2`" überschrieben ist. Dort kann die Zeile

```
BAUDRATE        EQU     9600
```

auf

```
BAUDRATE        EQU     38400
```

abgeändert werden. 38400 Baud ist die maximale Geschwindigkeit für die simulierte serielle Schnittstelle. Sollten Kommunikationsprobleme auftreten, empfehle ich eine geringere Baudrate (z.B. 19200) zu wählen.

Noch eine Bemerkung: Ältere C167-Controller hatten keine PLL-Einheit zum Generieren des Systemtakts. Diese Bausteine erforderten einen externen 40MHz-Takt, der zur Erzeugung des Systemtakts mit exaktem duty-cycle von 50% intern durch zwei geteilt wurde. Aus diesem Grund ist in `INSTPHY7.A66` stets 40 MHz die Berechnungsgrundlage. Die Zeile `XTAL EQU 40000000` darf **NICHT** geändert werden, da zur Berechnung der Zeitparameter für die serielle Kommunikation XTAL ohnehin durch zwei dividiert wird.

4. STARTUP-CODE

Wenn Sie mit dem Monitor (Varianten 1 bis 3) arbeiten, initialisiert dieses Programm die C167-Hardware bereits so, dass das Flash-Memory und das SRAM richtig angesteuert werden (Adressraum und Timing, Register `SYSCON`, `BUSCON0`, `BUSCON1`, `ADDRSEL1`). Ihr KEIL-Projekt erfordert keine spezielle Startup-Datei, da Ihr Programm in eine richtig konfigurierte C167-Umgebung geladen wird. Erst wenn Sie Ihr Anwendungsprogramm in das FLASH-Memory laden möchten, ist eine STARTUP-Datei erforderlich, die mit dem komfortablen Tool DAvE generiert werden kann.

5. EINSTELLUNGEN in der KEIL-Entwicklungsumgebung

Hier werden die wichtigsten Einstellungen für die Keil-Oberfläche beschrieben, die für das Arbeiten mit den verschiedenen Monitor-Varianten erforderlich sind.

In µVision Version 1.xx (KEIL Software-Entwicklungsoberfläche) sind unter "***Options - L166 Linker - Location - Reserve 1***" die im Artikel beschriebenen Adressräume zu reservieren (für alle Varianten 08h-0Bh, bzw. 0ACh-0AFh für die Varianten 1 und 2)

Unter ***Options - L166 Linker - Classes*** sind die Adressbereiche einzustellen, die der Linker (Locater) bei der Adresszuordnung für Anwenderdaten und -code verwenden darf. Alle drei hier beschriebenen Varianten laden das Anwenderprogramm in das SRAM. Der Compiler generiert, abhängig vom gewählten Speichermodell (in unserem Beispiel SMALL) und abhängig von den Steuerworten, die bei der Definition von Variablen, Konstanten und Funktionen verwendet werden, Sections und Classes vom Typ Code (Speichertyp ROM) oder Daten (Speichertyp ROM oder RAM) mit bestimmten Namen, denen der Locater physikalische Adressen zuweist. Wir können folgende Bereiche festlegen:

```
NCODE (0x0000-0x9FFF)
ICODE (0x0000-0x9FFF)
NDATA (0x4000-0x7FFF)
NDATA0 (0x4000-0x7FFF)
NCONST (0x0000-0x3FFF)
```

Die Classes bedeuten:

`NCODE`	Anwendercode (Speichertyp ROM)
`ICODE`	Initialisierungscode (Speichertyp ROM)
`NDATA`	nicht initialisierte Variable (Speichertyp RAM)
`NDATA0`	initialisierte Variable (Speichertyp RAM)
`NCONST`	Konstante (Speichertyp ROM)

Hinweis: Ein Blick in die Datei `*.M66` (Protokoll-Datei des Linker/Locaters) zur Kontrolle der vom Locater vergebenen physikalischen Adressen für die vom Compiler generierten Sections und Classes (Speichertyp ROM /RAM) ist ratsam.

Nachdem ein Anwenderprogramm erstellt wurde ("***Project - Make: Build Project***"), kann der dScope-Debugger aufgerufen werden. Wählen Sie im Hauptmenü den Punkt

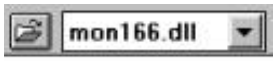
Run - dScope Debugger

dScope kann als Simulator und als Debugger arbeiten (in diesem Fall bezeichnet KEIL das Programm auch als tSCOPE). Durch die Wahl der entsprechenden DLL wird der entsprechende Arbeitsmodus eingestellt. Wir beschränken uns hier auf die Verwendung als Debugger (tSCOPE).

Das kitCON-Board und der PC sind über das Starterkit-Kabel zu verbinden. Achten Sie darauf, dass nur für Variante 1 der Bootstrap-Modus gewählt sein muss (kitCON167-1997: den roten Jumper JP2 1+2 schließen, kitCON167-1998: Schalter 1 auf SW3 schließen). Für alle anderen Varianten muss der Jumper

bzw. Schalter geöffnet sein. Der geschlossene Jumper bzw. Schalter aktiviert nach dem Drücken der RESET-Taste die C167-interne Bootstrap-Sequenz. Der MONITOR wird nur in Variante 1 mit dem Bootstrap-Mechanismus in das RAM geladen.

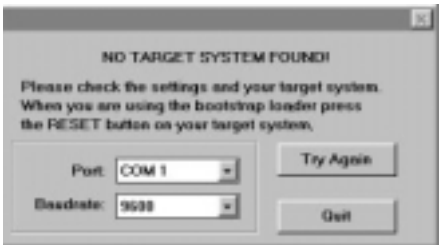
Wählen Sie nun im Auswahlfenster links oben die `MON166.DLL` aus. Durch die Auswahl der DLL



`MON166.DLL` setzen Sie dScope in den Debug-Modus (die DLL `80167.DLL` würde den Hardware-Simulator-Modus einstellen). Lassen Sie sich von der Bezeichnung `MON166.DLL` nicht verwirren. Die spezifische Auswahl des C167-Controllers für den Debugger erfolgt weiter unten.

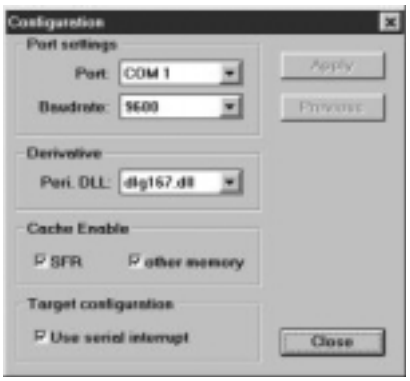
Arbeiten Sie mit Variante 1, so sollte jetzt ein Fenster erscheinen, das das Laden des MONITOR-Programms in das SRAM des kitCON-Boards anzeigt. Die Varianten 2 und 3 erfordern kein Laden des Monitors, da er sich ja bereits im Flash-Memory befindet.

Sollten Sie die Meldung



erhalten, kommuniziert der Monitor nicht mit dem dSCOPE-Programm. Überprüfen Sie, ob das Kabel an der eingestellten PC-Schnittstelle angeschlossen ist und ob die in `INSTPHY7.A66` gewählte Baudrate mit der in dScope einzustellenden Geschwindigkeit übereinstimmt.

Die entsprechende Dialogbox erhalten Sie unter "***Peripherals - Conf.***" im dScope-Hauptmenü. Für die Varianten 1 und 2 kann "***Use serial interrupt***" aktiviert sein, für Variante 3 muss diese Option deaktiviert sein. Stellen Sie außerdem in jedem Fall unter "***Derivative - Peri. DLL***" die Option `dlg167.dll` ein, damit im Debug-Modus die Komponenten des C167-Mikrocontrollers korrekt angezeigt werden. Diese sind geringfügig anders als beim C166 (`dlg166.dll`).



Noch eine Bemerkung zur Option "***Use serial interrupt***": Wenn Sie für die Varianten 1 und 2 diese Option aktivieren, kann das Anwenderprogramm durch ESC (Command-Fenster) oder STOP (Debug-Fenster) angehalten werden. In diesem Fall kann das Anwenderprogramm jedoch die ASC0-Schnittstelle des C167 nicht für Text-Ein- und Ausgabe verwenden (z.B. `printf`, `scanf`). Einige Demoprogramme (im Verzeichnis `C:\C166EVAL\EXAMPLES`) verwenden diese C-Ein-/Ausgabe-Befehle.

Nun kann das Anwenderprogramm geladen werden. Klicken Sie dazu auf das Symbol links von der Auswahlliste, in der Sie `MON166.DLL` gewählt haben. Alternativ kann das Anwenderprogramm über "***File - Load Object File***" geladen werden. Anschließend sollten Sie zumindest das Debug-Fenster öffnen. Über das Debug-Fenster kann das Programm gestartet werden (Menüpunkt "***Go***") oder im Einzelschritt-Verfahren (Menüpunkte "***Step Into***", "***Step Over***") abgearbeitet werden. Alternativ können Sie das Command-Fenster öffnen und dort den Befehl g,main eingeben. Laufende Programme können im Debug-Fenster über "***Stop***" (oder alternativ dazu im Command-Fenster durch Drücken der ESC-Taste) angehalten werden. Außerdem können Breakpoints gesetzt werden. Nach jeder Unterbrechung des Anwenderprogramms bzw. Ausführung eines Befehls im Single-Step-Modus erfolgt ein Rücksprung in den Monitor. Der Monitor liest die aktuellen Werte von internen Registern des C167 aus und überträgt diese an die dScope-Oberfläche, wo sie dargestellt werden. Welche Register , Speicherbereiche, Variablen usw. angezeigt werden sollen, können Sie unter dem Menüpunkt "***Peripherals***" einstellen.

6. Insider-Informationen

Adressbereiche und Speicherbelegungen der Monitor-Programme

Zum Verständnis der nachfolgenden Informationen sollten Sie unbedingt vorher das ***Kapitel 8*** des C167-User-Manuals (External Bus Controller) studieren.

Das kitCON-C167-Board des Starterkits enthält 256 kB Flash-Memory und 64 kB SRAM. Das Flash-Memory wird über `CS0` (chip select 0), das SRAM über `CS1` (chip select 1) aktiviert. Durch Konfiguration von `ADDRSEL1` kann festgelegt werden, für welchen Adressbereich der C167-Controller das CS1-Signal auf LOW (aktiv) setzt und damit das SRAM anspricht. Ein Blick in `BOOTPHY7.INC` und `INSTPHY7.INC` zeigt, dass `ADDRSEL1` mit der Hex-Konstanten `0008` geladen wird. Dadurch wird festgelegt, dass das erste Megabyte (`000000` bis `0FFFFF`) des Adressraums dem SRAM zugeordnet wird. Wird also eine Adresse im Bereich `000000` bis `0FFFFF` generiert, so ist damit stets eine Speicherzelle im RAM gemeint. Nun ist aber das SRAM physikalisch nur 64 kB groß. Das bedeutet, dass durch diese `ADDRSEL1`-Konfiguration das SRAM 16-fach gespiegelt in das erste Megabyte eingeblendet wird, da das SRAM nur die untersten 16 Bit der Adresse interpretiert. Die Adressen `FEAC5`, `EEAC5`, `DEAC5`, ..., `0EAC5` adressieren also alle dieselbe physikalische Speicherzelle.

Die Zuordnung von 1MB Adressraum für das SRAM wird vom Monitor deshalb gemacht, weil es das kitCON-167-Board von Phytec auch mit 1 MB SRAM gibt. Außerdem kann so der Adressbereich `F000` bis `FFFF` (Systembereich) auch für das SRAM genutzt werden (`0F000` ... adressiert das interne RAM bzw. die SFRs, `xF000` ... [x>>0] adressiert das SRAM).

Das Flash-Memory wird über `CS0` aktiviert. `CS0` wird immer dann LOW (aktiv) gesetzt, wenn eine Adresse nicht in einem Adressraum liegt, der durch `ADDRSEL1` bis `ADDRSEL4` festgelegt ist.

6.1 Memory-Mapping für die beschriebenen Variante 1

Für die Variante 1 wurde durch den Befehl

```
INSTALL PHY7 0 FEA FEC BOOTSTRAP
```

der Monitor so generiert, dass er für Daten den Adressbereich `FEA00` bis `FEBFF` verwendet. Der Code selbst wird in den Adressbereich ab `FEC00` abgelegt und ist ca. 6 kB groß. Auf Grund der vorherigen Ausführungen beginnt der Code des Monitors am Starterkit-Board somit auf der physikalischen SRAM-Adresse `EC00`

und der Datenbereich bei EA00. Ihr Anwenderprogramm darf somit in den SRAM-Bereich ab EA00 weder Daten noch Code ablegen, da ansonsten Programmteile bzw. Daten des Monitor-Programms überschrieben werden.

Um das zu verhindern, sollte (insbesondere bei größeren Anwendungsprogrammen) die folgenden Anweisungen unter "*Options – L166 Linker – Location*" eingetragen werden:

```
Reverve EA00h – EFFFh
```

Hier das vollständige **Address-Mapping** für die hier beschriebene Variante 1 (Monitor im SRAM):

000000H-00EFFFH	freies SRAM
00F000H-00FFFFH	internes RAM und SFR's
010000H-0FE9FFH	freies SRAM (Notiz 1)
0FEA00H-0FFFFFH	RAM für MONITOR
100000H-1FFFFFH	freies SRAM (Notiz 2)
200000H-23FFFFH	Flash Memory (Notiz 3)
240000H-....	Flash ROM (optional)

Notizen

1. Dieser Adressbereich entspricht nur dann physikalischem SRAM-Speicher, wenn das kitCON-Board mit 1 MB SRAM ausgestattet ist (siehe Handbuch). Beim Starterkit-Board werden diese Adressen auf die physikalischen Adressen 0000 bis E9FF des 64-kB-SRAMs gemappt (ADDRSEL1 = 0008), da vom 64 kB-SRAM nur die unteren 16 Bit des Adressbusses ausgewertet werden.
2. In der Originaldatei CONFPHY7.INC wird ADDRSEL2 mit der Hexkonstanten 1008 geladen. Damit wird CS2 (chip select 2) aktiviert, wenn eine Adresse im Bereich des zweiten Megabytes liegt (100000 bis 1FFFFF). CS2 wird vom Starterkit-Board nicht weiter verwendet.
3. Das Flash-Memory wird durch CS0 aktiviert. CS0 wird generiert, wenn eine Adresse außerhalb der konfigurierten ADDRSEL-Bereiche angesprochen wird.

6.2 Memory-Mapping für die beschriebenen Varianten 2 und 3

Für die Varianten 2 und 3 wurde der Monitor durch

```
INSTALL PHY7 0 FFE 2000 bzw.
INSTALL PHY7 2 FFE 2000
```

generiert. Der Monitor belegt für Daten den Adressbereich FFE00 bis FFFFF (das oberste Ende des (gespiegelten) SRAM-Bereichs). Der Code beginnt an der Adresse 200000 (FLASH-Adressbereich) und ist 6 kB groß. Eine Reserve-Anweisung (wie unter ***6.1***) ist hier nicht erforderlich, da es nicht zu einem Adresskonflikt zwischen Monitordaten und Anwenderdaten im internen RAM (ausgenommen sind wiederum Anwenderdaten im SRAM) kommen kann.

Das vollständige Adress-Mapping für die Varianten 2 und 3 (Monitor im FLASH-EEPROM):

000000H-00BFFFH	freies SRAM
00C000H-00EFFFH	SRAM,XRAM,CAN
00F000H-00FFFFH	internes RAM, SFR
010000H-0FFDFFH	freies SRAM (Notiz 1)
0FFE00H-0FFFFFH	Daten für Monitor
100000H-1FFFFFH	freies SRAM (Notiz 2)
200000H-23FFFFH	Flash Memory (Notiz 3)
240000H-....	Flash ROM (optional)

Notizen

1. wie vorhin (Variante 1)
2. wie vorhin (Variante 1)
3. Für die Varianten 2 und 3 liegt der Code des Monitor-Programms im Speicherbereich ab 200000. Auch dieser Speicherbereich ist nicht wirklich (physikalisch) mit Flash-Memory hinterlegt. Wie bereits beim SRAM beschrieben, erscheint auch das Flash-Memory im 16-MB-Adressraum des C167 mehrfach gespiegelt. Die Adresse 200000 ist physikalisch die Speicherzelle 0 im 256-kB-Flash-Modul des Starterkit-Boards. Da der Flash-Programmer FLASHT.EXE physikalische Adressen ab Adresse 0 erfordert, war im Aufruf

   ```
   ..\BIN\OH166 MONITOR H167 OFFSET (-0x200000)
   ```

   die Offset-Angabe erforderlich.

7. Schlussbemerkungen

Herrn Ing. Wilhelm Brezovits, Siemens Wien und Herrn Dipl.-Ing. Hans Schneebauer, Keil München danke ich für die Unterstützung und die Informationen, ohne die dieser Artikel nicht zustande gekommen wäre.

Übrigens:

Die **Siemens Semiconductor Group** wurde kürzlich in **Infineon** umbenannt.

Ergänzende und weiterführende Literatur und Web-Sites zum Thema des Artikels

[1] Erfolgreich Starten mit dem Infineon C167CR-Starterkit und dem Software-Entwicklungssystem von KEIL, Walter Waldner 1998. Verfügbar auf: *http://www.htblmo-klu.ac.at/lernen/*

[2] Umfassende Informationen über die Infineon-Mikrocontroller finden Sie auf der Web-Seite *http://www.infineon.com/products/ics/34/34.htm*

[3] Das Internet-Angebot der Firma KEIL finden Sie unter den Adressen *http://www.keil.com/*

[4] Die Homepage der Firma Phytec: *http://www.phytec.de/*

[5] EXBO – das I/O-Board für Mikrocontroller-Experimente zum Nachbauen.
Informationen und Dateien zum Laden auf Ihren PC finden Sie über die Homepage des Autors *http://www.htblmo-klu.ac.at/lernen/*

Kleiner Monitor ganz groß

MINIMON

Systemmonitor für die Infineon 16-Bit Mikrocontroller C16x

Christian Perschl, Peter Kliegelhöfer

Mikrocontroller

Überall sind sie in Massen zu finden: in Automobilen, Haushaltsgeräten, Computerteilen oder Videorekordern: die **Mikrocontroller**. Sie stellen die „Eierlegenden Wollmilchsäue" der großen Familie von Integrierten Schaltkreisen dar: Ein Alleskönner soll sämtliche „intelligente" Aufgaben von der Messwertaufnahme, Abfrage von Sensoren bis zur Signalerzeugung übernehmen. Deshalb sind im Mikrocontroller neben der CPU auch eine Reihe von Peripherieeinheiten (Timer, Standard-Schnittstellen, A/D-Wandler, digitale Ein-/Ausgänge) enthalten.

In allen Bereichen der Anwendungen von Mikrocontrollern (Automotive, Computer Peripheral, Consumer Electronics, Steuerungs- und Regelungssysteme) ist ein immer stärkerer Trend zu **Single-Chip**-Applikationen erkennbar: Nicht zuletzt aus Kostengründen soll dabei auf alle externe Hardware, speziell auf Speicherbausteine verzichtet werden und für die Applikation lediglich interne Speicherresourcen des Controllers wie Flash oder OTP (One Time Programmable) verwendet werden. Ein weiteres Argument für die Verwendung von internen Flash-Speichern ist die Wartbarkeit: Neuere Programmversionen sollen jederzeit und einfach einzuspielen sein, ohne dass nur eine Hardwarekomponente ausgetauscht oder geändert werden muss.

Mikrocontroller der 16 Bit-Klasse sind vielfach für einen Single-Chip-Betrieb ausgelegt: So existieren schon recht lange Derivate mit internem Program-Flash, den Durchbruch schafften diese Bausteine jedoch erst im Laufe der letzten Jahre.

Was ist Minimon ?

Mikrocontroller leben, mehr als alle anderen elektronischen Bausteine, von den verfügbaren Tools. Eine Reihe von Compilern, Programmierutilities, Monitoren und andere Softwareentwicklungstools unterstützt den Entwickler von Mikrocontrolleranwendungen beim Entwurf und Testen des Produktes.

Dieser relativ neue Trend der Single-Chip-Applikationen schafft aber auch für Tools neue Anforderungen: Speziell Tools, die direkt mit der Zielhardware in Kontakt stehen, sollen die vorhandene Hardware möglichst ressourcensparend verwenden und flexibel in bezug auf Lokatierung und Erweiterungen sein. Keinesfalls sollen bestimmte Bereiche vom Monitor blockiert sein. Diese Anforderungen haben zum Universal-Tool „Minimon" geführt:

Minimon ist ein ressourcensparender Monitor, welcher mittels Bootvorgang in das interne RAM des Mikrocontrollers geladen wird. Er ist nicht auf externe Ressourcen angewiesen und kann jederzeit in einen anderen Speicherbereich verschoben werden. Nicht zuletzt deshalb ist Minimon für Single-Chip-Applikationen prädestiniert.

Trotz der „Winzigkeit" des Monitors (ca. 400 Bytes) ist der Funktionsumfang beachtlich:

- Beschreiben, Auslesen und Vergleichen von beliebigen Speicherbereichen im Chip
- Setzen und Auswerten von Peripherieregistern
- Laden und Starten von Programmen / Subroutinen
- Programmieren des OnChip Flash-/OTP-Speichers

Ein Frontend sorgt für einen komfortablen Umgang mit diesen Funktionen. Die Voraussetzungen für die Benutzung von Minimon sind: ein System mit Windows 95/98/NT®, Zielhardware mit einem beliebigen C16x 16Bit-Mikrocontroller und eine serielle Verbindung zwischen beiden (z.B. C16x Starterkit, Prototypenboard).

Monitorkonzept

Das Programm Minimon umschließt zwei Komponenten: den Monitor-Kernel, welcher auf der C16x-Zielhardware läuft, und das Frontend unter Windows, mit welchem alle Einstellungen und Aktionen bequem vorgenommen werden können. Kommuniziert wird über ein einfaches Protokoll, welches gut dokumentiert und frei verfügbar ist.

Der Monitor-Kernel ist dafür ausgelegt, im internen RAM des Controllers Platz zu finden. Er ist lediglich ein paar hundert Bytes groß und kann nach dem Booten jederzeit in einen beliebigen anderen Bereich (z.B. externes RAM) verlegt werden. Der Kernel kennt nur ein paar rudimentäre Kommandos wie Speicherblock lesen/schreiben, Prozeduraufruf und Softwarereset. Mit diesen wenigen Grundbefehlen können alle komplexeren Operationen durch das Frontend realisiert werden. Dies bedeutet, dass so viel wie möglich Intelligenz in das Frontend-Programm verlagert wurde und damit der Monitor-Kernel schlank bleibt.

Der Verbindungsaufbau zwischen Kernel und Frontend erfolgt im allgemeinen durch einen Bootvorgang. Es ist allerdings auch möglich, den Kernel dauerhaft im Targetspeicher abzulegen.

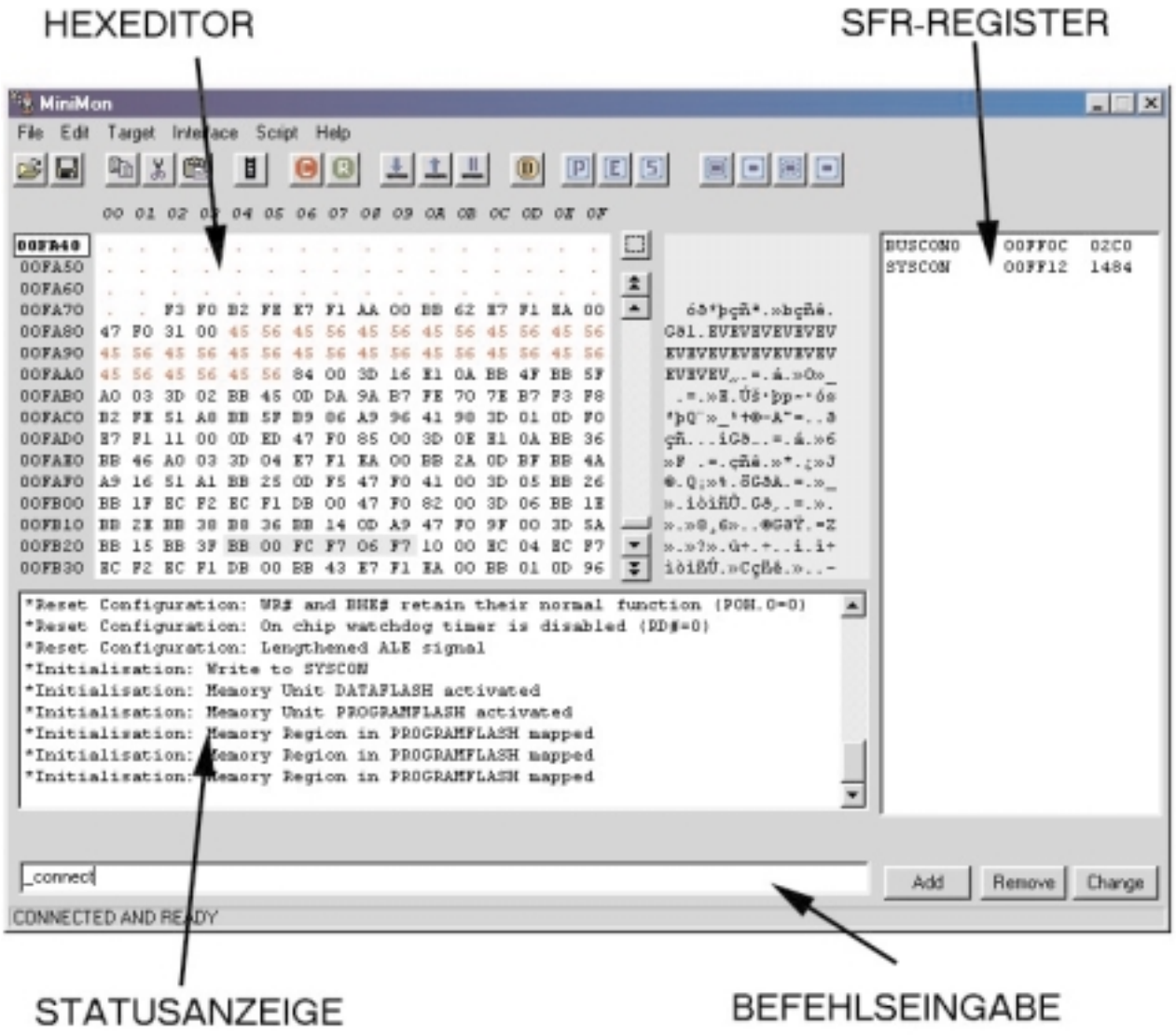
Frontend-Funktionen

Das Frontend realisiert eine Reihe von komfortablen Funktionen:

Ein zentraler Bestandteil des Frontends ist der **Hexeditor**. Hier können beliebige Speicherbereiche angezeigt, editiert, Intel-Hex-Dateien geladen und gespeichert werden. Beliebige, auch unzusammenhängende Adressbereiche können selektiert und mit bestimmten Werten gefüllt werden. Der Hexeditor kann auch unabhängig von eventuell vorhandener Targethardware verwendet werden.

Mit dem Hexeditor kooperieren die **Transferfunktionen**. Die im Hexeditor selektierten Speicherbereiche können in den Speicher

 Christian Perschl e9327470@stud1.tuwien.ac.at



der Zielhardware geschrieben oder von dort gelesen werden. Auch eine Vergleichsfunktion steht zur Verfügung, z.B. um den Inhalt des Speichers mit einer Datei zu vergleichen.

Handelt es sich nicht um RAM-, sondern um Flash- oder OTP-Speicherbereiche, so können diese programmiert werden. Die **Flashprogrammierung** stellt ebenfalls eine der Hauptanwendungen von Minimon dar. Der Monitor unterstützt von Grund auf alle internen Flash- und OTP-Typen der Familie. Minimon kann aber jederzeit um eigene Treiber erweitert werden, da sowohl Kommunikationsprotokoll als auch Treiber-Interface offen und dokumentiert sind.

Eine weitere wichtige Funktion von Minimon ist das Laden und **Starten von Applikationen**. Das Programm wird im Intel-Hex Format in den Hexeditor geladen und mit der Transferfunktion „Download“ in den Speicher verfrachtet oder in das Flash programmiert. Danach kann das Programm per Minimon „angesprungen“ werden oder ein Software-Reset durchgeführt werden. Eine interessante dritte Möglichkeit ist die Verwendung des Subroutinen-Interfaces (verwendet von den Flashtreibern). Hier können beliebige Programme als Unterprogramme des Monitors aufgerufen werden. Dies ermöglicht eine geordnete Rückkehr in den Monitor nach Terminierung des Programms. Ideales Einsatzgebiet: kleine Testroutinen für Peripherie oder Hardware.

Ist eine Anwendung einmal gestartet, so kann die (serielle) Kommunikation mit Minimon erfolgen, und zwar mit der **Terminalfunktion**: Empfangene Daten von der Applikation werden in einem Terminalfenster entweder als Hexadezimalwerte oder als Text angezeigt. Auch das Senden von Hex-Zeichen oder Text zur Applikation ist mit Minimon einfach möglich.

Das **„Systemmonitoring“** stellt ebenfalls eine der Hauptanwendungsbereiche von Minimon dar. Gemeint ist damit das Initialisieren, Auslesen und Setzen von Peripherieeinheiten. So kann z.B. ganz ohne Programmierung ein Port-Pin gesetzt, ein Timer initialisiert oder ein Wert vom A/D-Wandler eingelesen werden. Auch zur Hardware-Evaluierung läßt sich damit Minimon sehr gut einsetzen. Häufig benötigte SFR-Register können im Minimon-Hauptfenster angezeigt und werden auch laufend aktualisiert.

So nebenbei bietet Minimon auch die Möglichkeit, im Hexeditor selektierte Bereiche zu **disassemblieren**, und so Speicherinhalte besser interpretieren zu können.

Flexibilität

Befehle können bei Minimon entweder über das Menü, über die Symbolleiste oder in der Kommandozeile eingegeben werden. Oft hat man immer wiederkehrende Befehlsfolgen. Da es mühselig und fehleranfällig ist, immer wieder die gleichen Befehle einzugeben, bietet Minimon die Möglichkeit der Ausführung von **Scripts** an: In eine einfache Textdatei werden einfach alle Befehle mit den Parametern hineingeschrieben. Zusätzlich können Scripts auch (begrenzte oder unbegrenzte) Schleifen enthalten.

SIEMENS-3

SIEMENS-4

Der Monitorkernel ist so konzipiert, dass er nur minimal Speicher benötigt. Das **flexible Speicherkonzept** ermöglicht, dass der Monitorkernel nach dem Starten in jeden beliebigen anderen Bereich verschoben werden kann. Auch die Flash-Treiber sind frei lokatierbar, um hier von vornherein keine Einschränkungen zu machen und eventuell wo anders benötigte Bereiche frei zu halten.

Ein weiteres Feature von Minimon ist das **Initialisierungskonzept**. Beim Start des Monitors wird im Normalfall keinerlei Initialisierung vorgenommen. Um dennoch Register beim Verbindungsstart zu setzen oder bestimmte Befehle aufzurufen, kann deren Initialisierung frei konfiguriert werden. Zum Initialisierungskonzept gehört auch die Konfiguration aller in der Zielhardware vorhandenen Speicherbereiche. Diese können dann aktiviert/deaktiviert oder gemappt werden.

Alle **Einstellungen** werden von Minimon nach dem Beenden gespeichert, und nach dem erneuten Starten ist alles beim Alten. Um zusätzlich das Arbeiten an mehreren Projekten bzw. verschiedenen Zielsystemen zu ermöglichen, können Einstellungen in eigene Dateien gespeichert (Benutzerprofile) oder von dort geladen werden. Zudem werden alle Ein- und Ausgaben im Terminal/Statusfenster protokolliert.

Alle **Quellcodes** der Mikrocontrollerprogramme – Monitorkernel und Flash/OTP-Treiber – liegen bei. Außerdem sind Protokoll und Treiber-Schnittstelle dokumentiert. Dies erleichtert die Erstellung und Anpassung neuer Treiber.

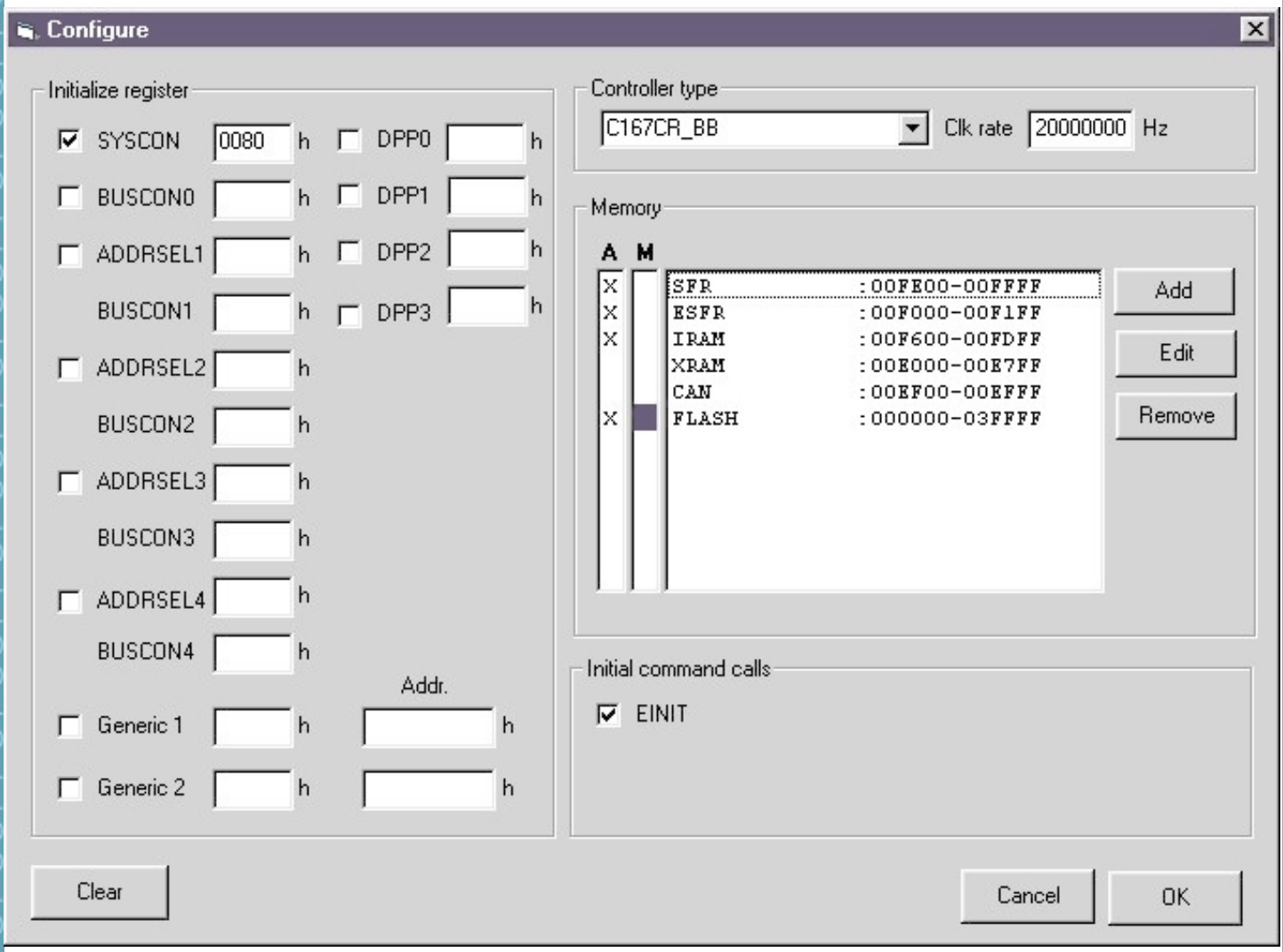
Neue Controllertypen können ebenfalls einfach in Minimon integriert werden. Alle Mikrocontroller-spezifischen Eigenschaften sind in externen Textdateien gespeichert.

Anwendung

Wer mit MiniMon arbeitet, wird schnell feststellen, dass dieses Tool hinsichtlich Funktionalität die Antwort auf verschiedene Anforderungen ist, die sich aus dem täglichen Umgang mit Mikrocontrollern in der Anwendungstechnik ergeben. Mitnichten wird man also auf überflüssige Features stoßen, ebensowenig auf eine überladene Windows-Oberfläche. Beim strukturellen Aufbau wurde auf höchstmögliche Transparenz Wert gelegt, die neuen Usern den schnellen Einstieg in das Tool ermöglicht und erfahrenen Anwendern die Einbindung eigener Funktionen (z.B. Testprogramme oder Memory-Treiber) erleichtert.

Dank der Selbstinstallationsroutine (Setup) werden bei den ersten Schritten mit MiniMon Fehler weitestgehend ausgeschlossen; schließlich ist nichts ärgerlicher, als ein fehlerhaft installiertes Programm. Der Zielpfad kann natürlich frei gewählt werden. Die aktuelle MiniMon Version 2.1.16 beansprucht inklusive Systemdateien ca. 6 MB auf der Festplatte.

Danach wird die Hardware (z.B. C167 Starterkit) an eine der seriellen Schnittstellen des PC angeschlossen, und schon kann's losgehen. Bei der Schnittstellenwahl sollte darauf geachtet werden, dass diese exklusiv für Minimon zur Verfügung steht.



Christian Perschl e9327470@stud1.tuwien.ac.at

Nach dem Start von Minimon empfiehlt sich, einmalig die Hardwarekonfiguration vorzunehmen: Auswahl des Controller-Typs, eventuelle Initialisierungsregister setzen, Speicherbereiche festlegen. Standardmäßig ist der Typ C167CR und dessen interne Speicherbereiche konfiguriert, sodass man sich in vielen Fällen die Hardwarekonfiguration sparen kann. ***Bild 3*** zeigt das Hardware-Konfigurationsmenü:

Sodann stellen wir die Verbindung zwischen Frontend und Target her (`CONNECT`). Der Verbindungsaufbau zwischen Host (PC) und Target (z.B. Phytec C16x Starterkit) geschieht mit Hilfe des für die C16x Familie typischen internen Bootstraploaders (BSL).

Das Booten dauert nur 1-2 Sekunden, es wird der Chip-Identifier, die Resetkonfiguration angezeigt und alle Register-Initialisierungen vorgenommen. Zur Initialisierung von Special Function Registern (SFRs) des jeweils selektierten Controller-Derivats stützt sich MiniMon auf standardisierte SFR-Beschreibungsdateien (ASCII-Format), wie sie von Infineon auch an die offiziellen Toolpartner herausgegeben werden.

Als Alternative zum Booten kann der Minimon-Kernel auch in einem externen Speicher dauerhaft programmiert werden. Es erfolgt dann lediglich ein „RECONNECT“ zum Verbindungsaufbau.

MiniMon unterstützt zur Zeit Übertragungsraten von bis zu 28800 Baud, was für Laboranwendungen völlig ausreichend ist. An höheren Baudraten wird jedoch bereits gearbeitet.

Sobald die Verbindung besteht, kann mit der eigentlichen Arbeit begonnen werden. Im folgenden werden einige einfache Anwendungsbeispiele vorgestellt:

Upload

Es soll der Bereich von 0000-0FFF (die ersten 4 kByte) aus dem Targetspeicher geladen und in die Datei `TEST.HEX` gespeichert werden.

Der gewünschte Speicherbereich wird im Hexeditor markiert und mit dem Befehl `UPLOAD` in das Frontend geladen. Dieser nun geladene Bereich kann dann entweder in eine Datei gespeichert oder disassembliert werden. Ein Script für diese Funktion könnte folgendermaßen aussehen:

```
 connect                  ; Verbindung herstellen
 clearselections          ; alle Selektionen entfernen
 addselection 0000,0FFF   ; Bereich 0000h-0FFFh markieren
_upload                   ; Diesen Bereich aus dem
                          ; Target-Speicher laden
_save test.hex            ; als Intel Hex Datei test.hex speichern
```

Wollen wir den Bereich disassemblieren, dann rufen wir anstatt dem Befehl Speichern den Befehl Disassemblieren (`_disassemble`) auf.

Download und Start

Oder wir wollen ein Programm in den Targetspeicher laden und starten. Zuerst wird die Datei geladen. Der geladene Bereich ist jetzt auch selektiert. Mit `DOWNLOAD` wird das Programm in das Ziel geladen (es muss sich um RAM handeln). Sodann kann das Programm angesprungen oder ein Software-Reset durchgeführt werden.

Der Script

```
 connect                ; Verbindung herstellen
_load c:\test\test.hex  ; Datei laden und im Hexeditor selektieren
 download               ; Selektion in den Target-Speicher laden
_srst                   ; Software-Reset durchführen
```

Handelt es sich um Flash-Speicher, so muss anstelle des Downloads der Programmierbefehl (`_iprogram`) aufgerufen werden. Dieser löscht bei Bedarf auch alle zu beschreibenden Sektoren.

Hardwaretest

In diesem Beispiel wollen wir an einem Ein-/Ausgabeport des Mikrocontrollers abwechselnd Einsen und Nullen anlegen:

```
 _mov  DP8,FF ; Port 8 auf Ausgang stellen

 DO           ; Schleife
   mov P8,FF  ; Ausgänge auf 1 stellen
  _delay 5    ; 5 Sekunden auf 1 lassen
   mov P8,00  ; Ausgänge auf 0 stellen
  _pause      ; Auf Tastendruck warten
 LOOP UNTIL 100 ; 100 Durchläufe

  mov  DP8,00 ; Port 8 auf Eingang stellen
 _einit       ; den Befehl EINIT aufrufen
```

Subroutinen-Interface

Für Profis bietet sich außerdem ein ganz besonderes Feature: das Monitor-Interface, mit dem eigene Subroutinen (wie auch Flashtreiber) geladen und aufgerufen werden können.

Testprogramme werden als Subroutinen geschrieben, mit Minimon geladen und aufgerufen (Befehl `_CALL`). Über Register werden der Routine Parameter übergeben und zurückgegeben. Nach Beendigung der Routine (mit dem `RET`-Befehl) wird in den Monitor zurückgekehrt. Somit enthält die Subroutine wirklich nur alle für diesen Test spezifischen Befehle, das Booten, die Kommunikation und Darstellung der Ergebnisse wird von Minimon übernommen.

Bei der Anwendung von Minimon sind der Phantasie keine Grenzen gesetzt, und gerade durch die Skriptfähigkeit können viele Test-, Lade-, Programmier- oder Initialisierungsaufgaben einfach und durchschaubar durchgeführt werden. Minimon versteht sich daher durchaus als Gerüst für eigene Aufgaben und Speziallösungen.

Fazit

Minimon ist ein Tool, das es eigentlich schon lange hätte geben müssen. Es bietet für jede C16x-Platform – speziell aber Single-Chip-Applikationen - ein breites Spektrum an Einsatzmöglichkeiten. Gerade die Vielfältigkeit und hohe Flexibilität macht es aber zu einem Profi-Tool, Anfänger verlieren sicher schnell den Überblick. Die umfangreiche Online-Hilfe und Referenz kann hier jedoch Abhilfe verschaffen.

Minimon ist jedoch kein Ersatz für herkömmliche „Monitor“-Programme. Diese haben als Hauptaufgabenbereich Software-Debugging, wogegen Minimon eher als Hardware-Tool zu klassifizieren ist. Das Programm lebt deswegen auch davon, dass auch die neuesten Typen rasch unterstützt werden und kontinuierlich auf dem laufenden gehalten wird. Deshalb wird Minimon auch ständig weiterentwickelt und gewartet. Es ist auch bereits vielfach bei C16x-Profis auf der ganzen Welt im Einsatz.

Minimon hat noch einen Vorteil: Es ist Freeware und kostet daher nix – außer ein paar Augenblicke, um es aus dem WWW zu laden. Für jeden C16x-Anwender ist daher Minimon ein Muss.

Minimon kann kostenlos heruntergeladen werden unter *http://stud4.tuwien.ac.at/~e9327470/minimon/minimon.htm*

e9327470@stud1.tuwien.ac.at Christian Perschl

CC166 und MK166

zentrale Tools der Tasking-Toolkette

Gottfried Prandstetter

CC166 ist das zentrale Steuerprogramm, welches die einzelnen Tools der Kette bedient und somit wesentlich zur automatischen Übersetzung beiträgt. Sowohl Benutzer von EDE als auch Entwickler, welche lieber mit einem eigenen Editor und Batch-Dateien arbeiten, werden von CC166 unterstützt. Nachfolgend soll für Benutzer der zweiten Kategorie anhand eines kleinen Beispiels gezeigt werden, wie sich eine komfortable Übersetzung realisieren läßt.

Zentrale Steuerdatei für die Toolkette ist die Datei `makefile`. In dieser Textdatei lassen sich alle Anweisungen aller Übersetzungsstufen unterbringen. Man kann also mit nur einer Steuerdatei ein ganzes Projekt übersetzen.

Diese Übersetzung erfolgt dann intelligent, d. h. nur jene Module, welche tatsächlich verändert oder noch nicht bearbeitet wurden, werden übersetzt.

Gestartet wird dieser Übersetzungsvorgang durch den Aufruf von MK166. Daraufhin sucht und analysiert MK166 `makefile`, entscheidet welche Teile des Projekts zu übersetzen sind und überlässt CC166 die Ausführung der Übersetzungsarbeit. Dazu übergibt er an CC166 eine Reihe von Parametern. Da die Menge an Parametern schnell die zulässige Länge einer DOS-Eingabezeile überschreiten, übergibt er diese Parameter bei Bedarf in einem File. CC166 weiß mit welchen Tools es den Auftrag zu übersetzen hat, ruft diese auf, generiert dazu seinerseits die nötigen Tool-Parameter bzw. reicht manche Parameter durch.

Obwohl man also ein Projekt auch mit CC166 allein weitgehend automatisieren kann, kommt wahrer Luxus erst durch die Zusammenarbeit von MK166 und CC166 auf.

Das `makefile` - Beispiel soll ein kleines Programm für SAB167 als Zielprozessor übersetzen, d. h. einige der Anweisungen dienen dazu von 166 auf 167 umzuschalten (`EXTEND`). Kommentarzeilen beginnen immer mit einem # als Umschaltzeichen, Befehlszeilen werden nur bis zum Auftreten eines # abgearbeitet. Weiters werden die Makros `CFLAGS` und `LDFLAGS` benutzt. Die Statments in `CFLAGS` werden an den Compiler bzw. Assembler durchgereicht, `LDFLAGS` wird für den Linker/Locator verwendet. Zum Auflösen dieser Makros sucht und verwendet MK166 die Datei `MK166.MK` im Verzeichnis `ETC`. In dieser Datei sind noch weitere Makros vordefiniert, für eine normale Übersetzung reichen die beiden genannten.

Enthält das `makefile` Syntax-Fehler, so wird nicht einfach abgebrochen sondern MK166 versucht kleinere Verstöße wie überflüssige oder fehlerhafte Parameter zu „übersehen“. Auch Zeilen, welche keinerlei Sinn ergeben, also Kommentare ohne vorangestelltes # werden verdaut. Trotzdem gibt es z.B. Probleme, wenn hinter dem Backslash, welcher hier benutzt wird um Zeilen zusammenzuhängen, noch Kommentar eingefügt wird.

Was bei der Übersetzung geschieht, kann man bei aktiven –v Switches gut verfolgen. Die dadurch erzeugte Informationsflut kann aber auch störend wirken und sollte, wenn alles zufriedenstellend läuft, ausgeschaltet werden. Eventuelle Fehlermeldungen kann man dann besser erkennen.

Auf der nächsten Seite nun das komplette `makefile`. Es ist praxiserprobt und sollte auch in Ihren Projekten arbeiten. Stellen Sie die File-Namen und Parameter entsprechend Ihren Anforderungen ein. Selbstverständlich muss für jedes Projekt ein eigenes Directory angelegt werden und ein Suchpfad auf das BIN-Verzeichnis existieren.

Was geschieht nun wenn man dieses Makefile übersetzen läßt? Das Kommando dazu lautet schlicht: `MK166` Wir gehen davon aus, dass alle Files neu übersetzt werden müssen. MK166 generiert folgende Aufrufe: Dreimal wird CC166 aufgerufen, um *.obj – Files zu erzeugen

```
cc166 -c -v -tmp -s PRINT -S -O2 -x EXTEND t.c
cc166 -c -v -tmp -s PRINT -S -O2 -x EXTEND serio.c
cc166 DEF(MODEL,SMALL) DEF(C167) DEF(EVA,0) DEF(MUXBUS,1)
start.asm –c-v
```

Ein Parameter-File wird erzeugt:

```
mk01674a.tmp:
LISTSYMBOLS
'me rom  (0000h TO 07fffh)  me rom (18000h TO 27fffh)
 me ram  (0e000h TO 0e7ffh) me ram (30000h TO 3ffffh)
 cl(CFAR (30000h TO 30fffh))'
'me ir(0f000h)  re me(0f200h TO 0f5ffh) re pp(PECC0)'
```

```
# Abhängigkeiten der Files werden definiert:
# t.hex wird erzeugt, abhängig von t.obj serio.obj start.obj
all             : t.hex
t.hex           : t.obj serio.obj start.obj
# Anzeige d. Tool-Kommandos und Listfiles erzeugen:
VERBOSE = -v -tmp -s PRINT
#            -tmp löscht tmp.-Dateien nicht
#                 -s mergt C-Code in *.src und *.lst
#                    PRINT  hebt default NOPRINT auf,erzeugt *.lst-Files
# Switches f. C-Compiler:
CFLAGS X        = -S -O2
#                 -S Static allocation of automatics
#                     O1 = default, O2 = size O3 = speed optimize
# Das Makro CFLAGS wird zusammengesetzt aus ....
CFLAGS    = $(VERBOSE) $(CFLAGS X) -x EXTEND  #-t
#                                              -t : Modul-Daten anzeigen
# Switches f. Linker:Speicherbereiche deklarieren:
LDFLAGS_X = "'me rom  (00000h TO 07fffh) \
             me rom  (18000h TO 27fffh) \
             me ram  (0e000h TO 0e7ffh) \
             me ram  (30000h TO 3ffffh) \
             cl(CFAR (30000h TO 30fffh))'"
# Zuordnung physischer-logischer Speicher über Variablenklasse CFAR
# Internes RAM, internes XRAM, PEC-Pointer reservieren:
SET EXT LOC = "'me ir(0f000h)\
               re me(0f200h TO 0f5ffh) \
               re pp(PECC0)'"
# Makro LDFLAGS zusammensetzen:
LDFLAGS     = LISTSYMBOLS $(LDFLAGS X) $(SET EXT LOC) $(SET MODEL)-v -x
# LISTSYMBOLS: Tabelle erzeugen
# switch -x setzt richtige Library über CC166!!!
# switch -v zeigt Tool-Kommandozeile
# switch -ihex serzeugt Hex-File
# start.obj soll aus start.asm erzeugt werden:
start.obj : start.asm
# Sonderfall CSTART.ASM !!!
# An M166 sollen DEFINES zur bedingten Übersetung übergeben werden.
# Hier wird CC166 direkt mit jenen Parametern aufgerufen, welche an
# M166 übergeben werden sollen.
# Achtung! Die folgende Befehlszeile muss eingerückt werden !!!
   cc166 DEF(MODEL,SMALL) DEF(C167) DEF(EVA,0) DEF(MUXBUS,1) start.asm -c -v
# -c : nicht linken, nur übersetzen mit M166 und A166
# -v : zeige Aufrufzeilen detailliert
# Radikal alle Output-Files löschen mit: MK166 clean
clean:
        del *.obj
        del *.abs
        del *.src
        del *.map
        del *.lno
        del *.h86
        del *.hex
        del *.lst
```

```
-v -x
t.obj serio.obj start.obj
```

CC166 wird mit dem Parameter-File gefüttert, um das Projekt zu linken und ein HEX-File zu erzeugen:

```
cc166 -cf -ihex -o t.hex -f mk01674a.tmp
```

Hier ist das Projekt fertig übersetzt und das Ergebnis `t.hex` liegt vor. Viermal wurde CC166 gerufen, um die Übersetzungstools zu starten. CC166 generiert seinerseits die Parameter für die Tools, welche er zur Übersetzung braucht.

In der ersten Zeile hat CC166 folgendes getan:

```
 c166 t.c -o t.src -e -s -S -O2 -x
 a166 t.src TO t.obj NOPR EXTEND PR
```

In der letzten Zeile (4. Aufruf) hat er das Projekt gelinkt und das HEX-File erzeugt:

```
l166 LOC TO cc09531c.tmp
PTOG t.obj serio.obj start.obj ext\c166s.lib ext\f166
s.lib PR(t) LISTSYMBOLS
ME rom  (0000h TO 07fffh)  me rom  (18000h TO 27fffh)
me ram  (0e000h TO 0e7ffh) me ram  (30000h TO 3ffffh)
cl(CFAR (30000h TO 30fffh))
ME ir(0f000h)  re me(0f200h TO 0f5ffh) re pp(PECC0)
ihex166 cc09531c.tmp t.hex
```

Die Möglichkeiten von Make-Files sind ähnlich flexibel wie die von Batch-Dateien. Hier sollte nur ein brauchbares, einigermaßen übersichtliches Minimal-Gerüst beschrieben werden. Erweiterungsmöglichkeiten entnehmen Sie bitte den Dokumentationen der jeweiligen Programme.

Eine Hand wäscht die andere

Uniprog: 16-Bit-Controller programmiert 8-Bit-Controller

Christian Perschl

Einleitung

Für viele einfache Steuerungs- oder Regelungsaufgaben reichen die immer noch sehr verbreiteten 8-Bit-Controller der 8051/C500-Familie. Wenn schon einfach, dann wäre eine Single-Chip-Applikation angebracht, d.h. alle Programm- und Datenressourcen im Chip intern, keine externen Speicherbausteine, die das Design oft erheblich verkomplizieren.

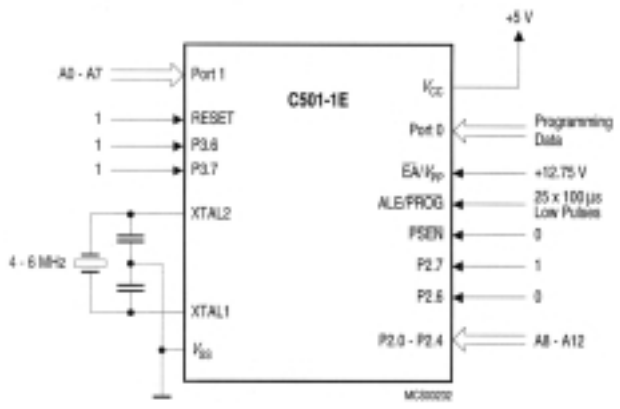
Der C501G-1EP ist ein 80C32 mit internem OTP-Speicher. OTP bedeutet One Time Programmable = genau einmal programmierbar. Für Evaluierungszwecke ist diese Speicherart zwar nicht so geeignet, denn einmal falsch programmiert, läßt sich der interne Speicher im Gegensatz zu einem EPROM- oder FLASH-Baustein nicht mehr löschen oder neu programmieren.

Ist die Designphase abgeschlossen, so bietet jedoch der interne OTP-Speicher eine interessante Variante sowohl für kleinste Stückzahlen als auch größere Produktionen.

Aber wie den internen OTP-Speicher programmieren ?

Programmierung des OTP

Die Programmierung des C501G-1EP geschieht extern durch Anlegen von Daten, Adressen und Steuerleitungen. Zusätzlich muss ein Taktsignal für die interne Programmierhardware angelegt werden.

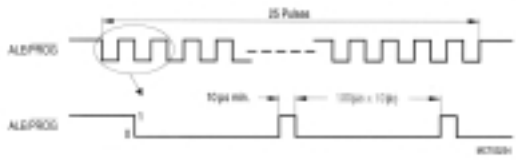


Die zu programmierende Adresse im 8 KByte großen internen OTP wird an Port 1 bzw. den unteren 5 Bits des Port 2 angelegt. Die Daten liegen am Port 0 an. Der Reset-Eingang bleibt während der Programmierung auf logisch 1. Dann werden noch ein paar Steuersignale an den Pins P3.6, P3.7, P2.6, P2.7 und PSEN angelegt.

Ein externer Takt an XTAL1/XTAL2 ist ebenfalls erforderlich.

Die eigentliche Programmierung erfolgt durch Anlegen der Programmierspannung von 12.75 V und der Erzeugung von Programmierimpulsen am Pin ALE/PROG:

Es gibt auch die Möglichkeit, die Programmierung zu überprüfen: Um den Speicher auszulesen, muss am Pin 2.6 logisch 1 anliegen. Zusätzlich muss der ALE/PROG-Eingang und der Vpp-Pin auf 1 liegen. Nach Anlegen der Adresse können am Port 0 der Inhalt byteweise ausgelesen werden.



Kompliziert? Na ja, zumindest ein bisschen Aufwand. Ich möchte im folgenden eine Komplettlösung vorstellen, die jeder nachbauen kann und kaum Aufwand darstellt. Was wird benötigt? Ein C167-Starterkit, ein bisschen Lötfertigkeit und ein paar Standardbauteile.

Uniprog

Uniprog ist ein universelles Programmiertool zur Programmierung des internen OTP (One-Time-Programmable) des C501G-1EP. Dabei wird ein Standard-C167-Starterkit zur Ausführung des Programmieralgorithmus verwendet. Ein Zusatzmodul, welches auf die Starterkit-Verbindungsleiste gesteckt wird, nimmt dann in einem ZIF (Zero Insert Force) – Sockel den zu programmierenden Mikrocontroller auf.

Das auf dem Host laufende Frontend-Programm von Uniprog ermöglicht ein komfortables Laden und Editieren von Intel-Hexdateien und eine ebenso einfache Programmierung. Außerdem kann durch das vorgestellte Konzept Uniprog einfach für weitere Flash/OTP-Module angepasst werden.

Installation

Folgende Komponenten werden benötigt:

- PC mit Windows 95/98/NT
- C167 Starterkit mit seriellem Kabel und Netzteil
- Aufsteckmodul C501G-1EP
- Softwarepaket Uniprog

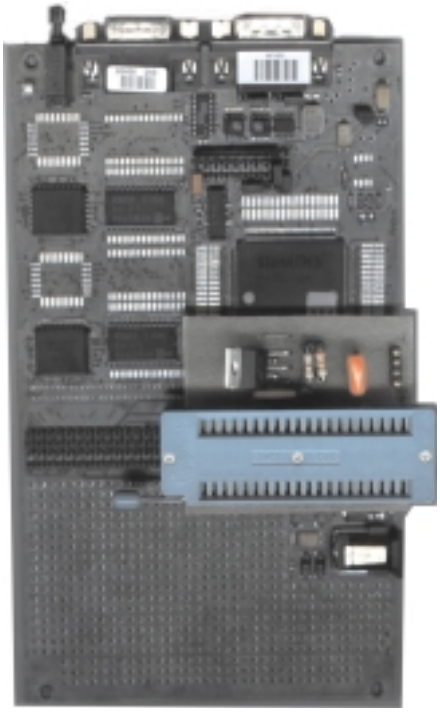
Installation des Softwarepakets

Als erstes muss das Softwarepaket Uniprog installiert werden. Dabei wird zuerst das Programm `SETUP.EXE` aufgerufen.

Bei der nachfolgenden Installationsroutine muss lediglich der Ziel-Pfad angegeben werden.

Inbetriebnahme des Starter Kits

- Der Starterkit muss mit allen Standardeinstellungen in Betrieb genommen werden (alle Jumper wie beim Auslieferungszustand.).
- Sodann muss das Starterkitboard mit dem seriellen Kabel mit dem PC verbunden und an das Netzteil angeschlossen werden.
- Als nächstes muss der Bootstrap-Loader (roter Jumper) aktiviert werden (wenn nicht bereits gesetzt)
- Jetzt kann der Starterkit programmiert werden. Die dem Softwarepaket beiliegende Datei `UNIPROG.H86` (ist im Uniprog-Verzeichnis im Unterverzeichnis `MCPROG` zu finden) muss nun in das externe Flash des Starterkits programmiert werden. Dazu kann z.B. das Phytec-Flashtool oder Minimon verwendet werden.
- Dann muss der Bootstrap Loader wieder deaktiviert werden (roter Jumper aus). Dies ist erforderlich, damit der C167 auf dem Starterkit das soeben programmierte UNIPROG.H86 nach dem Einschalten ausführt.
- Als letztes wird das Uniprog-Modul auf die Starterkit-Verbindungsleiste gesteckt. Es empfiehlt sich, ein einmal (richtig) gestecktes Modul nicht wieder herunterzunehmen, da es dadurch leicht beschädigt werden kann. Das Modul muss wie in der folgenden Abbildung eingesteckt werden:



Bedienung

Einstecken des C501G-1EP

Der Zeitpunkt zum Einstecken des zu programmierenden Mikrocontrollers ist an und für sich kein Problem, er kann also entweder vor dem Einschalten oder während des Betriebes – natürlich nicht während des Programmierens – gewechselt werden, da Uniprog sich standardmäßig im Lesemodus befindet und daher die meisten I/O-Leitungen auf Eingang geschaltet sind.

Wichtig ist dabei lediglich, dass der C501 mit Pin 1 zur Mitte des Starterkits hin (beim Hebel des ZIF-Sockels) platziert wird:

Anlegen der Programmierspannung

Da der C501G-1EP eine Programmierspannung von 12,75 V benötigt, muss eine Spannung von ca. 15 V wie beschriftet am Adapter angelegt werden. Ein Spannungsregler erzeugt genau die erforderlichen 12,75 V beim Programmieren bzw. 5 V beim Lesen.

Frontend

Uniprog enthält ein einfach zu bedienendes Frontend, welches es ermöglicht, den C501-OTP-Speicher zu programmieren, den OTP-Speicher auszulesen oder Inhalte zu vergleichen.

Das Frontend besteht aus folgenden Komponenten:

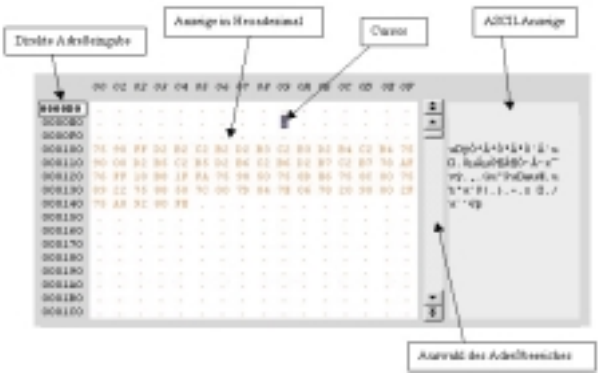
- Menü und Toolbar: Hier können alle Befehle aufgerufen werden
- Speicheranzeige: Hier wird der aktuelle Inhalt des Frontend-Buffers angezeigt oder geändert.
- Statusfenster: Hier werden Statusmeldungen angezeigt

Kommunikation herstellen

Um eine Kommunikationsverbindung herzustellen, muss der Menübefehl ***Target - Connect*** aufgerufen werden. Alternativ kann auf die Symbolschaltfläche ***Connect*** (rot) geklickt werden. Im Statusfenster unten sollte dann eine entsprechende Meldung angezeigt werden.

Die Speicheranzeige

Der aktuelle Inhalt des Frontendbuffers wird in der Speicheranzeige angezeigt. Dabei werden immer 256 Bytes angezeigt:



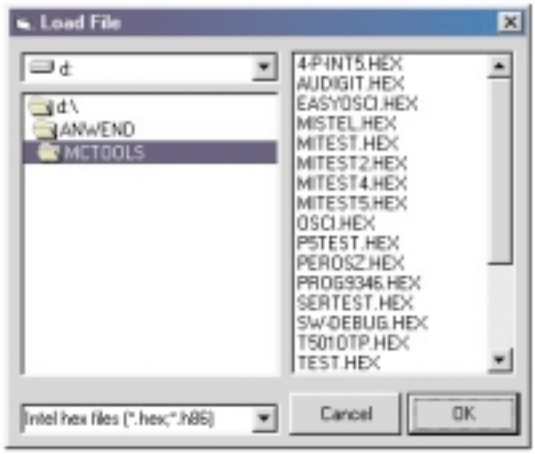
- Welcher Adressbereich gerade angezeigt wird, kann mit dem Rollbalken rechts bestimmt werden.
- Es kann auch direkt eine Adresse angegeben werden, und zwar rechts oben im Feld “Direkte Adresseingabe”.

- Mit dem Cursor kann eine bestimmte "Zelle" angefahren werden, entweder mit den Pfeiltasten oder mit der Maus. Das Datenbyte an der Cursorposition kann durch Eingabe eines Hexwertes geändert werden.
- Nicht verwendete Zellen werden mit einem Punkt dargestellt. Verwendete Daten können noch unterschieden werden, ob sie zur Zeit nur im Frontend-Buffer stehen (rot), oder ob sie tatsächlich dem Inhalt des OTP-Speichers entsprechen.
- Zusätzlich wird der aktuelle Adressbereich auch mit ASCII-Zeichen dargestellt (rechts).

Ein Bereich kann mit der Maus selektiert werden: Dazu wird der zu selektierende Bereich mit der Maus gezogen.

Datei laden

Um eine Intel-Hex-Datei in den Frontendbuffer zu laden, muss der Menü-Befehl ***File Load*** angeklickt werden. Danach erscheint ein Fenster zum Auswählen der Datei:



Nachdem eine Datei ausgewählt wurde, kann diese mit OK geladen werden. Sie wird dann auch gleich in der Speicheranzeige selektiert und dargestellt.

Programmieren des OTP Speichers

Der OTP-Speicher des aktuell im Sockel steckenden Mikrocontrollers kann byteweise programmiert werden. Es wird immer die aktuelle Selektion programmiert !

Die Aktion wird durch den Befehl ***Target-Program Selection*** im Menü oder das entsprechende Symbol durchgeführt.

Auslesen des OTP Speichers

Der OTP-Speicher des aktuell im Sockel steckenden Mikrocontrollers kann auch byteweise ausgelesen werden. Wie beim Programmieren wird die aktuelle Selektion ausgelesen! Daher muss vor dem Auslesen der gewünschte Bereich selektiert werden.

Die Aktion wird durch den Befehl ***Target-Read Selection*** im Menü oder das entsprechende Symbol durchgeführt.

Vergleichen

Eine einfache Möglichkeit, den Inhalt des OTP-Speichers mit dem Inhalt des Frontend-Buffers zu vergleichen, bietet der Menübefehl ***Target - Compare Selection***: Es muss vor Ausführung des Befehls wiederum der zu vergleichende Bereich selektiert werden (Zur Erinnerung: Dateien werden nach dem Laden selektiert, d.h. durch Aufrufen von ***File-Load*** und ***Target-Compare Selection*** wird der Inhalt des OTP-Speichers mit der gewünschten Datei verifiziert).

Ergebnis des Vergleiches: übereinstimmende Datenbytes werden schwarz dargestellt, nicht übereinstimmende rot.

Informationen zur Herstellung des Moduls

Beschaltung

- Die meisten Pins des Moduls (Adressen, Daten, Steuerleitungen) sind mit I/O-Leitungen des C167CR auf dem Starterkit verbunden. Diese werden je nach Bedarf entweder als Eingänge oder als Ausgänge geschalten.
- Die Belegung der I/O Pins wurde so gewählt, dass sich ein minimaler "Verdrahtungsaufwand" ergibt, also keinesfalls nach den Ports des C167CR (keine "logische" Zuordnung).
- Die Adressleitungen sind immer auf Ausgang geschaltet.
- Ebenso sind die Modusverbindungen immer auf Ausgang geschaltet (Modus: Programmieren/Lesen).
- Die Datenleitungen werden je nach Modus auf Eingang oder Ausgang geschaltet.
- Der Takteingang wird mit einem Digitalsignal von der PLL-Einheit des C167CR versorgt (ca. 5 MHz).

Programmierspannungssteuerung

Da der C501G-1EP eine Programmierspannung von 12,75 V benötigt, muss eine Spannung von ca. 15 V wie beschriftet am Adapter angelegt werden. Der Standard-Spannungsregler L200 (erhältlich bei Conrad Electronics) erzeugt genau die erforderlichen 12,75 V beim Programmieren bzw. 5 V beim Lesen. Zwischen diesen beiden Spannungen umgeschaltet wird mittels I/O-Leitung und Ansteuerung eines Transistors.

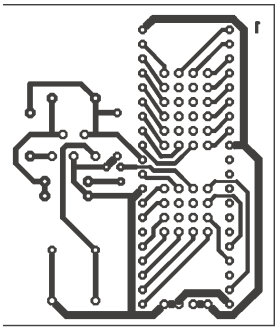
Die Spannungen ergeben sich durch den Spannungsteiler R2/R1 und errechnen sich nach folgender Formel:

$U = 2{,}75*(1+R2/R1)$

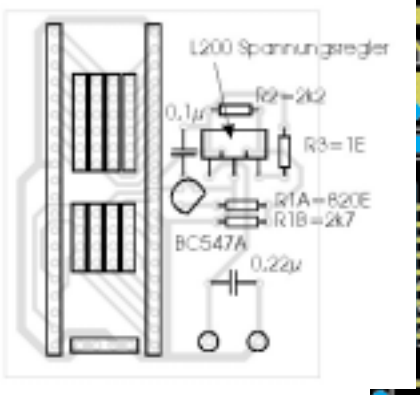
Bei ausgeschalteter Programmierspannung ergibt lediglich R2/R1B den Spannungsteiler (U=5V). Bei eingeschalteter Programmierspannung (U=12,75V) wird R1A parallel zu R1B geschaltet.

Bauanleitung

Zur Selbstbauanleitung werden hier Layout und Bestückungsplan angegeben:



Layout 1:1, Lötseite



Fazit

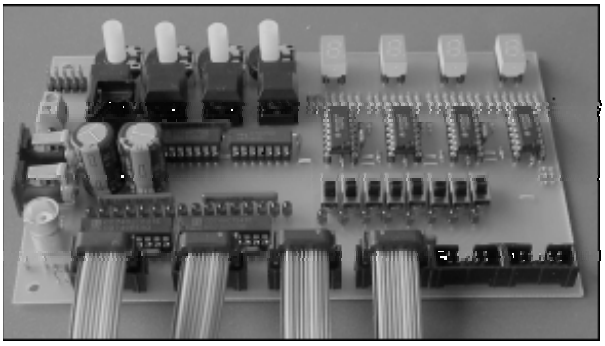
Es gibt natürlich professionelle Programmiergeräte für den C501-OTP, nur sind diese für den Hobbyelektroniker, Schulen oder einen kleinen Betrieb mit hohen Investitionskosten verbunden. Die hier vorgestellte Lösung "Uniprog" ist eine recht günstige Alternative (Starterkit: ca. 2000,- Modul ca. 300,-) zum Selbstbau. Die erforderliche Software ist frei verfügbar.

e9327470@stud1.tuwien.ac.at Christian Perschl

EXBO

Experimentierboard für Mikrocontroller-Übungen

Walter Waldner



EXBO wurde als Zusatzplatine zum Phytec kitCON-167CR-Board, das im Infineon Starterkit SK-167CR enthalten ist, entwickelt. Die Grundidee war, einfache Ein- und Ausgabekomponenten zu entwerfen, um mit dem Infineon 16-Bit-Mikrocontroller C167CR Experimente zum Kennenlernen dieses Systems durchführen zu können. EXBO wird mit den auf dem kitCON-Board über Steckleisten zugänglichen Ports des Mikrocontrollers C167CR mit Hilfe von Flachbandkabeln verbunden. Auch andere kitCON-Boards von Phytec (welche in den von Infineon vertriebenen Starterkits enthalten sind) können mit EXBO zusammengeschaltet werden.

Bei der Entwicklung von EXBO wurde in erster Linie an den Einsatz im Unterricht gedacht. Der Selbstbau dieser Platine sollte kaum Schwierigkeiten bereiten und auch die Gesamtkosten sind "schülerfreundlich". Inklusive Netzgerät muss man, je nach Einkaufsquelle und Menge, mit einem Gesamtbetrag von etwa ATS 1000,- rechnen. Die einseitige Platine kann selbst gefertigt werden, oder aber von der Firma MTM in Wien zum Stückpreis von ATS 300,- bezogen werden. Alle Unterlagen zum Selbstbau des EXBO-Boards können vom Internet geladen werden. Unter der Adresse *http://www.htblmo-klu.ac.at/lernen/exbomain.htm* finden Sie PDF-Dateien für den Platinenfilm, den Schaltplan, den Bestückungsplan und die Stückliste. Auch Bestellnummern für diverse Elektronik-Versandfirmen sind in der Stückliste bereits angeführt.

Die Komponenten von EXBO

Das EXBO-Board enthält folgende Komponenten:

16 Leuchtdioden (16 Bit – Ausgabeeinheit)

4 Siebensegment-Anzeigen mit BCD-Decodern (16 Bit – Ausgabeeinheit)

8 Schiebeschalter (1 x um) mit LED-Kontrolle (8 Bit – Eingabeeinheit)

4 entprellte Digi-Taster (4 Bit – Eingabeeinheit)

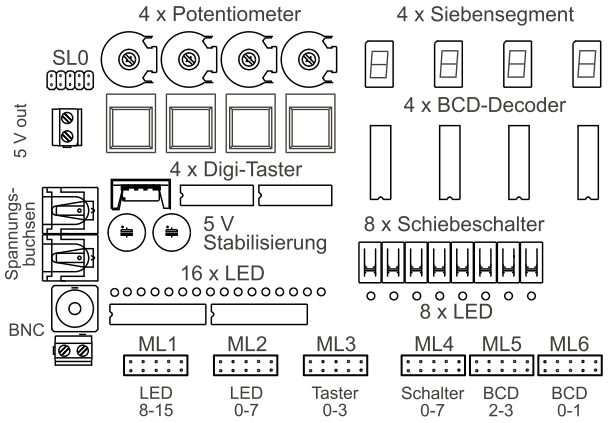
4 Potentiometer (4 x Analog-Eingabe: 0 – 5 Volt)

5 Volt – Spannungsstabilisierung mit Buchse zum Anschalten weiterer Experimentierschaltungen

1 BNC-Buchse zum Anschluss von Messgeräten

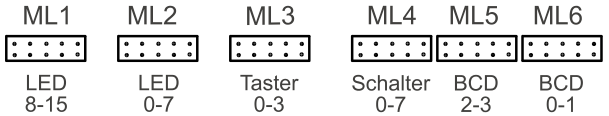
2 Spannungsbuchsen für 8-12 Volt Gleichspannung (von einem Steckernetzteil / ca. 1 A)

Das folgende Bild zeigt die ungefähre Lage der Komponenten von EXBO:

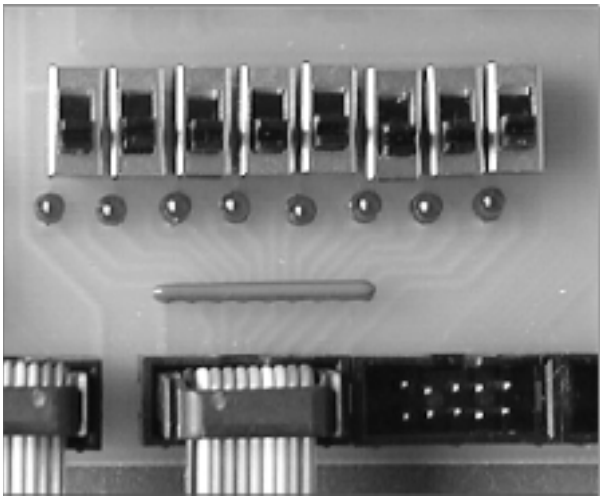


Alle EXBO-Einheiten sind mit Messerleisten (ML1 bis ML6) verbunden, die sich am unteren Rand der Platine befinden. Die Stiftleiste (SL0) der Analogausgänge (Potis) befindet sich im linken oberen Eck. Von diesen Messer- bzw. Steckerleisten kann mittels Flachbandkabeln eine Verbindung zum Phytec kitCON-167-Board oder auch anderen Mikrocontroller-Boards hergestellt werden. Die Belegung der Leisten wurde so gewählt, dass sie der Belegung der kitCON-Steckerleiste entspricht, über die man auf die Ports des Infineon-Mikrocontrollers C167CR zugreifen kann. Da Messerleisten bzw. Federleisten für Flachbandkabel mit 8 Pins kaum erhältlich sind, wurden 10-Pin-Versionen vorgesehen, wobei 2 der Pins unbelegt sind.

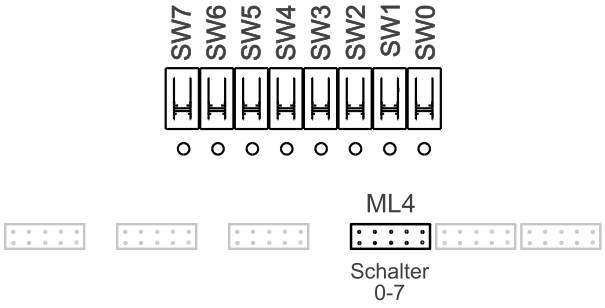
Für die folgenden Erklärungen seien die Messerleisten von links nach rechts wie folgt bezeichnet:



8-Bit-Eingabeeinheit: Schiebeschalter (Messerleiste ML4)



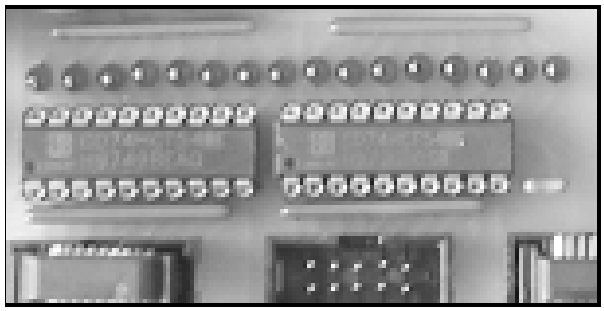
Diese Baugruppe umfasst 8 Schiebeschalter (je 1 x um), die entweder 0 oder 5 Volt an die Pins der Steckerleiste liefern. Mit dieser Einheit können beliebige 8-Bit-Kombinationen eingestellt werden. Zusätzlich zeigen Leuchtdioden den gewählten Ausgangspegel an, um so eine optische Rückmeldung über die eingestellten Bit-Kombinationen zu geben. Die Schaltereinheit kann mit einem Port des Mikrocontrollers verbunden werden, das auf INPUT konfiguriert ist. Der linke Schalter SW7 entspricht Bit 7, der rechte Schalter SW0 Bit 0 auf der zugehörigen Messerleiste / Stiftleiste ML4.



Belegung der Messerleiste ML4

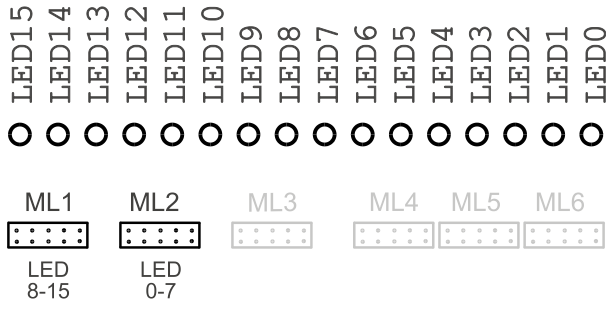
frei	SW7	SW5	SW3	SW1
frei	SW6	SW4	SW2	SW0

16-Bit-Anzeigeeinheit: Leuchtdioden (Messerleisten ML1 und ML2)



Die 16-Bit-Anzeige ist aus 16 Leuchtdioden aufgebaut, die über zwei invertierende Treiber-Bausteine vom Typ 74HC540 angesteuert werden. Die Platine sollte mit Low-Current-LEDs (1-2 mA) aufgebaut werden, um den Stromverbrauch niedrig zu halten. Die Eingänge der beiden Treiber-ICs sind auf die beiden linken Messerleisten ML1 und ML2 geführt. Die LEDs leuchten, wenn die entsprechenden Pins auf HIGH-Pegel gelegt werden. Da es sich bei den ICs 74HC540 um CMOS-Bausteine handelt, sind die Eingänge mit 1 MOhm-Pulldown-Widerstände auf LOW gelegt, um ein "Floaten" bei Nichtbelegung der Eingangsleitungen zu verhindern. Die Messerleisten ML1 und ML2 können mit Ports des Mikrocontrollers verbunden werden, die auf OUTPUT konfiguriert sind.

Die LEDs entsprechen von links nach rechts den Bits 15 bis 0. Die linken 8 LEDs (HIGH-BYTE) sind mit der Messerleiste ML1 (LED 8 ist Bit 0 auf dieser Messerleiste), die rechten 8 LEDs (LOW-BYTE) mit der Messerleiste ML2 verbunden (LED 0 ist Bit 0 auf dieser Messerleiste).



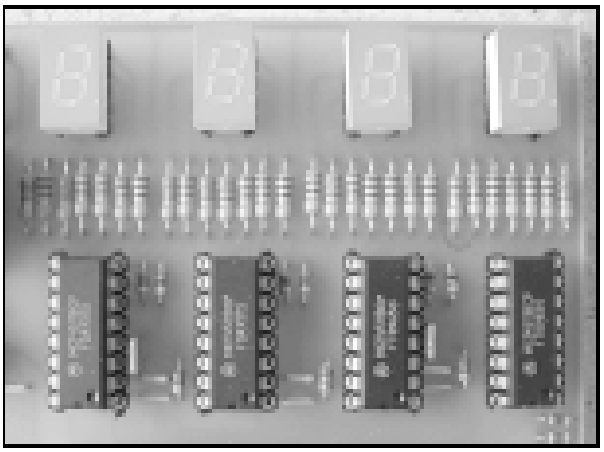
Belegung der Messerleiste ML1 (HIGH-BYTE)

frei	LED15	LED13	LED11	LED9
frei	LED14	LED12	LED10	LED8

Belegung der Messerleiste ML2 (LOW-BYTE)

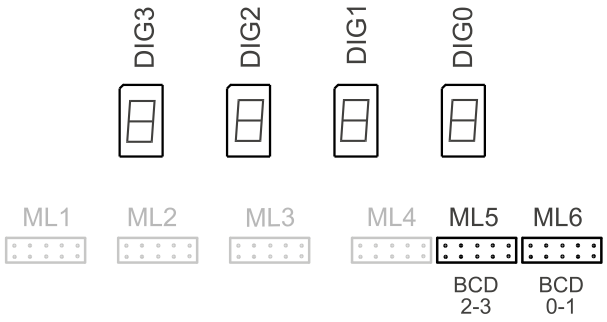
frei	LED7	LED5	LED3	LED1
frei	LED6	LED4	LED2	LED0

4-fach Siebensegment-Anzeige (Messerleisten ML 5 und ML6)



Die vier Siebensegment-Anzeigen vom Typ HDSP-7503 o.ä. (gemeinsame Kathode) werden über vier BCD-Decoder vom Typ MC14511 (4511) angesteuert. Diese Einheit kann zur Anzeige einer vierstelligen Dezimalzahl oder zwei zweistelliger Dezimalzahlen (jeweils in BCD-Darstellung) verwendet werden. Die Eingangsleitungen der BCD-Decoder werden, wie bei der 16-Bit-Anzeigeeinheit (LEDS), bei Nichtbeschaltung mit 1 MOhm Widerständen auf LOW gezogen und zeigen somit bei Nichtbeschaltung vier Nullen.

Die beiden linken Anzeigen (DIG3 und DIG2) sind mit der Messerleiste ML5, die beiden rechten Anzeigen (DIG1 und DIG0) mit der Messerleiste ML6 verbunden. ML5 und ML6 können mit Mikrocontroller-Ports verbunden werden, die auf OUTPUT programmiert sind.



Die Eingänge der BCD-Decoder seien (wie in Datenblättern üblich) mit A (Bit 0), B (Bit 1) , C (Bit 2) und D (Bit 3) bezeichnet.

Die vier Anzeigen mit BCD-Decoder sind von rechts nach links mit DIG0 (Einerstelle), DIG1 (Zehnerstelle), DIG2 (Hunderterstelle) und DIG3 (Tausenderstelle) bezeichnet (DIG steht für digit).

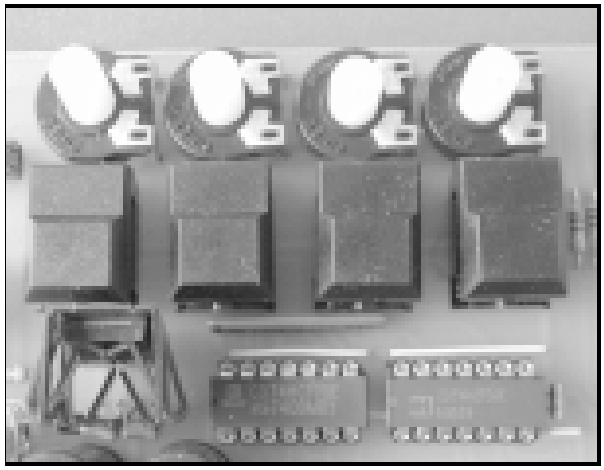
Belegung der Messerleiste ML5 (Tausender- und Hunderterstelle)

frei	DIG3.D	DIG3.B	DIG2.D	DIG2.B
frei	DIG3.C	DIG3.A	DIG2.C	DIG2.A

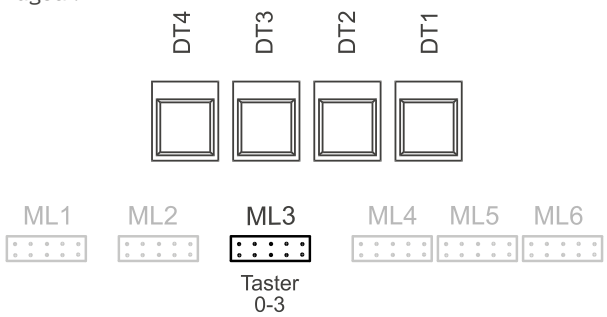
Belegung der Messerleiste ML6 (Zehner- und Einerstelle)

frei	DIG1.D	DIG1.B	DIG0.D	DIG0.B
frei	DIG1.C	DIG1.A	DIG0.C	DIG0.A

4-Bit-Eingabeeinheit (Digi-Taster) (Messerleiste ML3)



EXBO enthält vier Digi-Taster, die zum Erzeugen von sauberen 0-1- bzw. 1-0-Flanken verwendet werden können. Dies wird durch eine klassische Entprellungsschaltung mit R-S-Flipflops (aufgebaut mit zwei 74HC00-NANDs) realisiert. Für jeden Taster sind beide Flankensignale (0-1 und 1-0) auf der Messerleiste ML3 verfügbar.

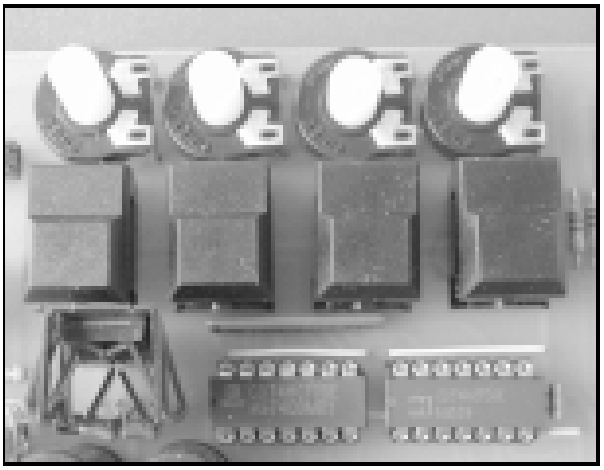


Belegung der Messerleiste ML3

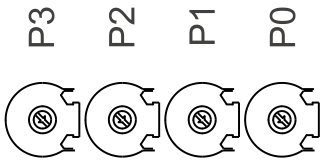
frei	#DT3	#DT2	#DT1	#DT0
frei	DT3	DT2	DT1	DT0

Die vier Digitaster werden von links nach rechts mit DT3, DT2, DT1 und DT0 bezeichnet. DTn bezeichnet das normale Ausgangssignal (0-1-Flanke bei Tastenbetätigung), #DTn das invertierte Ausgangssignal (1-0-Flanke).

4-fach Analog-Eingang (Potentiometer) (Stiftleiste SL0 - links oben)



Für Experimente mit dem Analog-Digitalwandler sind die 4 Potis des EXBO-Boards gedacht. Sie realisieren eine einfache Spannungsteilung der 5V-Versorgungsspannung und liefern somit beliebige Spannungswerte zwischen 0 und 5 Volt an die Pins der Steckerleiste SL0 (links oben auf der Platine). P0 bezeichne das rechte, P3 das linke Poti:

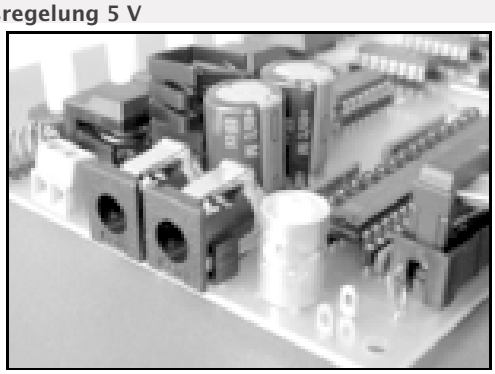


Belegung der Stiftleiste SL0

frei	frei	frei	P3	P1
frei	frei	frei	P2	P0

Zusätzlich auf der EXBO-Platine enthalten

Spannungsregelung 5 V

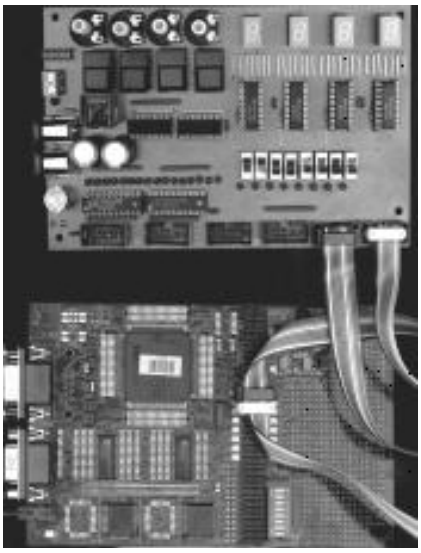


Das EXBO-Board arbeitet mit 5 V Gleichspannung. Diese Spannung wird aus der von einem Netzadapter gelieferten Eingangsspannung mittels 7805-Spannungsregler-IC erzeugt. Das Netzteil sollte etwa 8-12 Volt Gleichspannung liefern. Die EXBO-Platine enthält zwei Spannungsversorgungsbuchsen, die untereinander verbunden sind. An eine der Buchsen wird der Netzadapter angeschlossen, von der anderen Buchse aus kann das kitCON-Board mit der Spannung des Adapters versorgt werden. Die Buchsen entsprechen denen der kitCON-Platine. Der Netzadapter sollte etwa 1 A Strom liefern können. Das EXBO-Board ist mit einer Diode vor Verpolung geschützt. Am Netzadapter-Stecker muss Masse außen liegen !

BNC-Buchse

Auch für eine BNC-Buchse wurde auf der EXBO-Platine ein entsprechender Platz vorgesehen. Diese Buchse kann mit verschiedenen Pins des kitCON verbunden werden und erleichtert so das Anschließen von Messgeräten (Oszilloskop, Frequenzmesser, Logik-Analysator u.a.), die meist BNC-Eingänge aufweisen.

Verbindung zum kitCON-167-Board von Phytec und anderen Mikrocontroller-Boards



Wie bereits erwähnt, wurde EXBO vor allem für das Mikrocontroller-Starterkit SK-167CR von Infineon entwickelt. Die Messer- bzw. Steckerleisten wurden so belegt, dass eine direkte Flachbandkabel-Verbindung zwischen EXBO und kitCON möglich ist.

HINWEIS: Legen Sie beim Arbeiten das kitCON-Board unter das EXBO-Board, wie dies auf dem Bild zu sehen ist. Dabei wird die kitCON-Platine so ausgerichtet, dass sich die Buchse für die serielle Schnittstelle links befindet. Das Board steht also quasi „auf dem Kopf“ (bezogen auf die im kitCON-Handbuch abgedruckte Belegung der Steckerleiste). Dadurch erreicht man, dass sich auf der kitCON-Steckerleiste die Pins 0 der Ports, wie bei den EXBO-Messerleisten, rechts befinden.

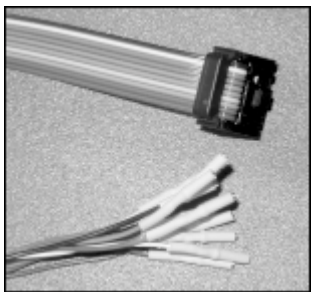
Bit7	Bit5	Bit3	Bit1
Bit6	Bit4	Bit2	Bit0

Beim HIGH-BYTE der 16-Bit-Ports ist die Bitnummer jeweils um 8 größer als hier angegeben. Messerleisten von EXBO können nun mit Flachbandkabel mit den jeweiligen Doppelzeilen der kitCON-Stiftleiste verbunden werden. Wir verwenden 10polige Flachbandkabel, da 8polige Messer- und Federleisten erfahrungsgemäß schwer aufzutreiben sind. Auf dem kitCON stehen die zwei unbelegten Pins der 10-poligen Buchsen (links) über die 4-spaltige Steckerleiste hinaus.

Wenn für die Experimente nur einzelne Pins eines Ports verwendet werden, empfiehlt sich der Bau des nachfolgend abgebildeten Kabels:

An das eine Ende ist eine 10-poligen Federleiste angeschlossen. Das andere Ende des Flachbandkabels wurde adernweise aufgetrennt. An die Leitungen wurden Crimpkontakte (Metallbuchsen) gelötet und mit Schrumpfschlauch überzogen.

Danksagungen



An dieser Stelle möchte ich Herrn Hermann Schönbauer von der Lehrwerkstätte der Firma Siemens in Bregenz für seine Hilfe beim Entwurf und Layout von EXBO danken.

Dank auch an Herrn Ing. Wilhelm Brezovits, Siemens Wien für die vielen Anregungen und wertvollen Unterstützungen und Herrn Ing. Gerhard Muttenthaler für die Bereitschaft, die EXBO-Platine (zur Selbstbestückung) über seine Firma MTM zu vertreiben.

Siemens Semiconductor = Infineon

Die Siemens Semiconductor-Gruppe, die unter anderem die Mikrocontroller und Starterkits produziert, wurde kürzlich in „Infineon“ umbenannt. Die Homepage ist

http://www.infineon.com

RESSOURCEN im INTERNET

Die Homepage des Autors finden Sie unter

http://www.htblmo-klu.ac.at/lernen

Von hier führen Links zu Starterkit-Dokumenten und zur EXBO-Seite.

Eine fertige EXBO-Platine kann von der Firma MTM Systeme, Ing. Gerhard Muttenthaler, Hirschstettnerstraße 19-21, 1220 Wien (Tel. 2032814) bezogen werden. Im Internet ist diese Firma unter

http://www.mtm.at/

erreichbar.

Die verschiedenen Download-Dokumente zu EXBO (Schaltbild, Platinenfilm, Stückliste, Bestückungsplan) sind im PDF-Format gespeichert. Zum Betrachten und Drucken dieser Dateien benötigen Sie den Adobe Acrobat Reader, der unter

http://www.adobe.com/

im Internet verfügbar ist.

EXBO wurde mit EAGLE entwickelt. Eine Demoversion, mit der Sie auch die EXBO-Dateien verarbeiten können, finden Sie unter

http://www.cadsoft.de/

LCD-Modul

Ansteuerung eines LCD-Moduls mit dem kitCON-167 (Infineon C167CR-Starterkit)

Walter Waldner

AUFGABENSTELLUNG

Die Ansteuerung einer **LCD-Punktmatrix-Anzeige** gehört zu den Standard-Anwendungen für Mikrocontroller. Der vorliegende Artikel beschreibt, wie gängige LCD-Module mit dem weitverbreiteten **HD44780**-LCD-Controller vom **Phytec kitCON-167-Board** (Inhalt des **Infineon Starterkit** für den **C167CR**-Mikrocontroller) angesteuert werden können. Die Anzeige-Einheit wird über den C167-Datenbus als **Memory-Mapped-I/O**-Komponente angeschlossen. Die Software wurde in der Sprache C für die KEIL-Toolkette entwickelt. Der Programmcode ist sehr klein, sodass sich das Ansteuerprogramm auch mit der Demoversion des KEIL-Systems (liegt dem Starterkit ebenfalls bei) problemlos übersetzen läßt.

Die folgende Beschreibung bezieht sich auf ein mir vorliegendes LCD-Punktmatrix-Modul mit dem Hitachi-Controller HD44780 (4 Zeilen zu je 20 Zeichen) mit Hintergrundbeleuchtung. Vergleiche mit einigen anderen Punktmatrix-Anzeigen, die vom selben Controller angesteuert werden, haben gezeigt, dass die Ausführungen auch für diese Module (1- und 2-zeilig, 16/20/40 Zeichen) gelten können. Überprüfen Sie bitte aber in jedem Fall anhand des Datenblatts Ihres Moduls, ob die vorgestellte Schaltung und insbesondere die Pinbelegungen übernommen werden können.

Für die folgenden Ausführungen setze ich voraus, dass Sie das Datenblatt des LCD-Moduls (siehe Anhang) und das Kapitel "External Bus Controller" des C167-User Manuals gelesen haben.

ANSCHLÜSSE DES LCD-MODULS

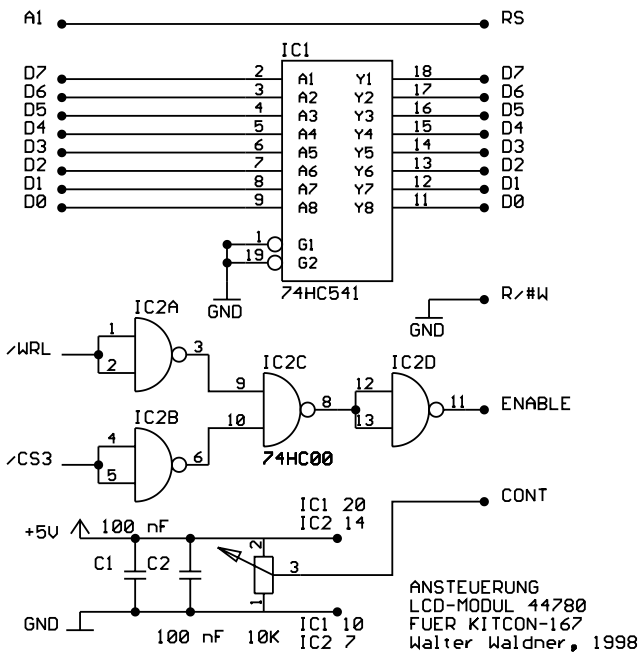
Die LCD-Module mit dem Hitachi HD44780-Controller haben folgende Anschlüsse (Pin-Belegungen gelten für den mir vorliegenden Typ mit 4 Zeilen / 20 Zeichen):

Pin	Symbol	Funktion
1	VSS	Versorgungsspannung: GND = 0V
2	VDD	Versorgungsspannung: 5 Volt
3	CONT	Spannungseingang für Kontrastwahl
4	RS	Register Select HIGH: Datenregister LOW: Befehlsregister
5	R/#W	Read / Write - Signal HIGH = READ LOW = WRITE
6	ENABLE	ENABLE SIGNAL
7	D0	Datenbus-Leitungen (D0=LSB)
8	D1	Datenbus-Leitung (D1)
9	D2	Datenbus-Leitung (D2)
10	D3	Datenbus-Leitung (D3)
11	D4	Datenbus-Leitung (D4)
12	D3	Datenbus-Leitung (D5)
13	D3	Datenbus-Leitung (D6)
14	D3	Datenbus-Leitung (D7)

Manche Module verfügen auch über eine Hintergrund-Beleuchtung. Der mir vorliegende Typ kann mit bis zu 150mA Strom für die Hintergrundbeleuchtung betrieben werden (beim Anschluss an 5V entsprechenden dimensionierten Vorwiderstand verwenden!).

ANSCHALTUNG DER LCD-ANZEIGE AN DAS PHYTEC KITCON-167-BOARD

Die LCD-Anzeigeeinheit wird mit dem Phytec-kitCON-167-Board über eine kleine Zusatzplatine verbunden. Die Schaltung und die erforderlichen Signale sind nachfolgend beschrieben.



Auf das LCD-Modul kann schreibend und lesend zugegriffen werden. Gelesen werden kann unter anderem das Busy-Flag, das anzeigt, ob die letzte interne Operation abgeschlossen ist und die Inhalte des Daten-RAMs (DDRAM = display data RAM) und des Character-Generator-RAMs (CGRAM).

Der Einfachheit halber werden wir auf das LCD-Modul nur **schreibend** zugreifen. Lesende Zugriffe (insbesondere die Abfrage des Busy-Flags) sind für eine volle Funktion nicht unbedingt erforderlich, da die LCD-Operationen dokumentierte Ausführungszeiten haben. Durch entsprechende Warteschleifen (die Wartezeiten dürfen dabei beliebig über den Minimalangaben liegen) kann sichergestellt werden, dass die Befehle in akzeptabler Geschwindigkeit übergeben werden. Der R/#W-Eingang des LCD-Moduls wird also fest auf GND-Pegel (LOW) gelegt (WRITE ist damit ständig aktiviert).

Das LCD-Modul wird gesteuert, indem Instruktionen und Daten übertragen werden. Zwei Register des HD44780 stehen dafür zur Verfügung:

Befehlsregister: in dieses Befehlsregister werden 8-Bit-Instruktionen geschrieben. Die Instruktionen haben unterschiedliche Ausführungszeiten (siehe Datenblatt)

Datenregister: in dieses Datenregister werden 8-Bit-Daten übertragen. Ist das MSB=0 (most significant bit), stellen die Daten den 7-bit-ASCII-Code des Zeichens dar, das angezeigt wer-

den soll. Manche LCD-Module haben zusätzliche Sonderzeichen (z.B. Katakana - japanische Schriftzeichen). Außerdem können benutzerspezifische Zeichen definiert werden (siehe HD44780-Datenblatt).

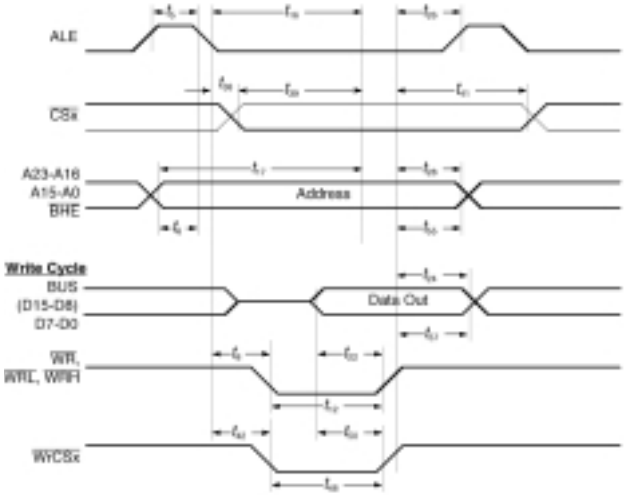
Zugriffe auf das Befehls- bzw. Datenregister werden durch den Pegel der Leitung RS (register select) unterschieden. RS kann somit als Adressleitung aufgefasst werden. In der vorliegenden Schaltung wird die C167-Adressleitung A1 auf den "Register Select"-Eingang (RS) des LCD-Moduls geführt. Es gilt:

RS = 0: Controll-Register wird adressiert

RS = 1: Datenregister wird adressiert

Die Übernahme der Daten vom Datenbus erfolgt durch die **fallende Flanke** des **ENABLE**-Signals. Leider stellt der C167 kein Bussignal zur Verfügung, das direkt als ENABLE verwendet werden kann. Wir können aber ein passendes ENABLE-Signal aus den C167-Signale #WRL und #CS3 erzeugen (siehe nachfolgende Bustiming-Diagramme). Dazu bilden wir die UND-Verknüpfung des invertierten #WRL- und des invertierten #CS3-Signals. ENABLE ist somit HIGH, wenn #CS3 und #WRL LOW sind und fällt auf LOW-Pegel, wenn #WRL wieder HIGH wird. In der hier abgebildeten Schaltung wird dazu der 4-fach-NAND-IC 74HC00 verwendet.

Die folgenden Diagramme zeigen das Timing eines Schreibzyklus für den C167 (demultiplexed 16-Bit-Datenbus) und den HD44780:



Timing für den "demultiplex" Bus - aus dem C167-Datenblatt

Der HD44780 kann über einen 4- oder 8-Bit-Datenbus angesteuert werden. Wir wählen die 8-Bit-Variante. Die C167-Datenleitungen **D0 bis D7** (Port 0L) werden über den Treiber-IC **74HC541** an die Dateneingänge des LCD-Moduls geführt. Eine direkte Verbindung der C167-Datenpins mit dem LCD-Datenbus ist beim Phytec-Board nicht möglich. Folgende Punkte sind zu berücksichtigen:

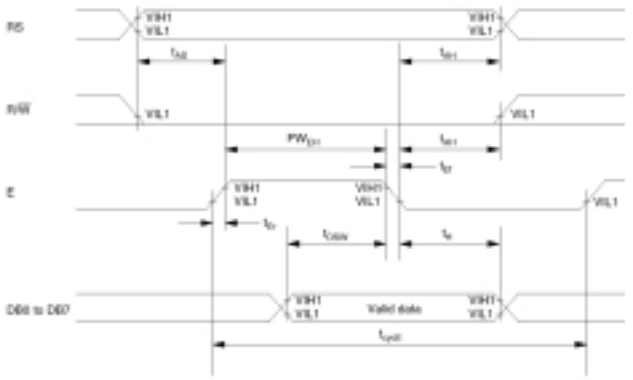
- Für die C167-Datenbusleitungen (Port 0) gelten (wie für alle Pins) laut Datenblatt bestimmte Grenzwerte für Sink-/Source-Ströme (siehe ***[1]*** und ***[2]***)
- Der Port 0 des C167 wird beim RESET-Vorgang zur Einstellung bestimmter Buskonfigurationen gelesen. Durch externe Pull-Down-Widerstände (siehe Schaltplan im kitCON-167-Handbuch) werden bestimmte Pins während des Resets auf LOW-Pegel gezogen. Die LOW-Pegel werden nur dann korrekt interpretiert, wenn bestimmte Grenzwerte für die Ströme nicht überschritten werden (siehe ***[2]***)

Beim Anlegen eines LOW-Signals an eine der Datenleitungen des HD44780 fließen ca. 125 µA über den internen Pull-Up-MOS-Transistor (siehe Abbildung der Eingangsschaltung im HD44780-Datenblatt), was zu viel ist. Insbesondere ist zu berücksichtigen, dass ja auch die Onboard-SRAMs und FLASH-EEPROMs den Datenbus belasten und insbesondere die Pull-Down-Widerstände des Phytec-Boards so gewählt sind, dass das zusätzliche Anschließen des LCD-Controllers die erforderlichen RESET-Pegel für die korrekte Initialisierung des Bus-Interface auf PORT0 verändert.

Aus den eben erwähnten Gründen schalten wir zwischen die unteren 8 Bits des C167-Datenbus (Port 0L) und den Datenleitungen des LCD-Controllers einen Treiber-IC vom Typ 74HC541 (Datenblatt siehe ***[3]***). Die Eingänge des 74HC541 liefern bzw. ziehen bei LOW- und HIGH-Pegel des Eingangssignals nur 0.1 µA (typisch, 1.0 µA maximal).

Beachten Sie bitte außerdem folgende Punkte:

- Als ICs müssen unbedingt CMOS-Bausteine der HC-Serie verwendet werden
- Die 100nF-Kondensatoren sind möglichst nahe an die Versorgungspins der ICs 1 und 2 zu bringen (diese buffern die Spitzenbelastungen der Versorgungsspannung bei Umschaltvorgängen in den CMOS-Ausgangsschaltungen)



Timing für den HD44780-Schreibzyklus - HD44780-Datenblatt

Das **Trim-Poti** (Wert ca. 10 kOhm) dient zur Kontrast-Einstellung der Anzeige. Der Schleiferausgang des Poti wird mit dem **CONTRAST**-Eingang (Pin 3) des Moduls verbunden.

Die Versorgungsspannung (GND und 5V) wird von der kitCON-Steckerleiste auf die LCD-Zusatzplatine geleitet. ACHTUNG: Sollte Ihr LCD-Modul über eine Hintergrundbeleuchtung verfügen, so darf die Spannungsversorgung dafür **NICHT** vom kitCON-Board geholt werden, da der Spannungsregler den dafür notwendigen Strom nicht liefern kann. Die Hintergrundbeleuchtung erfordert ca. 150 mA Strom (Vorwiderstand oder Konstantstromschaltung verwenden).

Das C-Source-Programm

```
/*********************************************
 Ansteuerung eines LCD-Modul mit Baustein HD44780
 ueber C167-Datenbus (Memory-Mapped-IO)
 (c) 1999 by Walter Waldner

 Anschluesse:
 D0..D7: Datenleitungen
 A1    : register select (0=instruction, 1=data)
 WRL   : write enable (low byte)
 CS3   : chip select
*********************************************/


#include <reg167.h>
#include <absacc.h>
#include <string.h>

void write string(char * s, unsigned char line);

#define LCD CNTL MVAR (unsigned char,0x800000)
#define LCD DATA MVAR (unsigned char,0x800002)

void wait(unsigned int w);


void main(void)
{

  unsigned char x;

  /* BUSCON3 / ADDRSEL3 */
  BUSCON3 = 0x0430;
  /* Adressbereich = 80:0000 .. 80:0FFF */
  ADDRSEL3 = 0x8000;

  /* Initialisierung Timer T3 - 1.6 us */
  T3CON = 0x0002;

  /* Software-Initialisierung */
  x = 0x38;
  wait(10000); /* 16 ms */
  LCD CNTL = x;
  wait(3000); /* 4.8 ms */
  LCD CNTL = x;
  wait(3000); /* 4.8 ms */
  LCD CNTL = x;
  wait(3000); /* 4.8 ms */

  /* init display */
  LCD CNTL = x;
  wait(3000); /* 4.8 ms */

  /* set display off */
  LCD CNTL = 0x08;
  wait(3000); /* 4.8 ms */

  /* clear display */
  LCD_CNTL = 0x01;
  wait(3000); /* 4.8 ms */

  /* display on */
  LCD CNTL = 0x0C;
  wait(3000); /* 4.8 ms */

  write string("LCD-MODUL 44780 ",1);
  write string("POWERED BY INFINEON",2);
  write_string("C167 MICROCONTROLLER",3);
  write string("Walter Waldner 1999",4);
  while (1);
}

void wait(unsigned int w)
{
  /* Aufloesung T3: 1.6 microseconds */
  T3 = 0;
  T3R = 1;
  while (T3 < w);
  T3R = 0;
}

void write string(char * s, unsigned char line)
{
  unsigned int i;

  /* DDRAM-Adresse setzen */

  if (line == 1)
    LCD CNTL = 0x80;
  if (line == 2)
    LCD CNTL = 0xC0;
  if (line == 3)
    LCD CNTL = 0x94;
  if (line == 4)
    LCD CNTL = 0xD4;
  wait(3000); /* 4.8 ms */

  for (i=0; i<strlen(s); i++)
  {
    LCD DATA = s[i];
    wait(30); /* 48 us */
  }
}
```

Wie bereits eingangs erwähnt, wird das LCD-Modul mittels der oben beschriebenen Schaltung als /O-Komponente an den C167 angeschlossen. **Memory-Mapped-I/O** bedeutet, dass ein Ein-/Ausgabe-Baustein quasi wie Speicher mit den Controller verbunden wird und die Register über Variable angesprochen werden können. Dem I/O-Baustein wird ein Adressbereich im Adressraum zugeordnet. Wenn eine Adresse innerhalb dieses Adressraums generiert wird, muss die I/O-Komponente selektiert sein ("**Chip select**"). Das Erzeugen eines Chip-Select-Signals erledigen oft externe Adressdekoder-Schaltungen. Der Infineon C167-Mikrocontroller stellt bereits 5 Chip-Select-Leitungen zur Verfügung, sodass auf externe Zusatzlogik verzichtet werden kann. Das Generieren der Chip-Select-Signale wird über die On-chip-Einheit "**External Bus-Controller**" des C167 gesteuert.

Das Phytec-kitCON-167-Board verwendet Chip-Select-1 (CS1) für die Ansteuerung des SRAMs und Chip-Select-0 (CS0) für die Ansteuerung des FLASH-EEPROMs. Die übrigen Chip-Select-Leitungen (CS2, CS3, CS4) sind verfügbar. Für unser Projekt verwenden wir **CS3**.

Für die Anbindung des LCD-Moduls über den Datenbus des C167 müssen die beiden Register **BUSCON3** und **ADDRSEL3** entsprechend konfiguriert werden.

BUSCON3 enthält die Einstellungen für die Bussignale und das Timing, **ADDRSEL3** spezifiziert den Adressbereich, für den CS3 aktiviert (LOW) werden soll.

In unserem Programm weisen wir dem LCD-Modul den Adressbereich **800000** bis **803FFF** zu (4 kB ist die kleinstmögliche Größe eines Adressfensters). Dazu muss in das ADDRSEL3-Register der Wert 0x8000 geschrieben werden. Die Wahl des Adressbereichs ist ziemlich willkürlich. Nicht verwendet werden sollten Adressbereiche, die von den SRAM- bzw. FLASH-Speichern belegt sind (siehe ***[4]*** und ***[5]***). Der hier gewählte Bereich liegt fernab von anderwärtig belegten Adressen.

Das LCD-Modul erfordert, wie bereits erwähnt, lediglich zwei Adressen (zur Unterscheidung zwischen Zugriffen auf das Daten- bzw. Befehlsregister). Gemäß unserer Schaltung ist die C167-Adressleitung A1 mit RS (register select) verbunden. Wir definieren im Programm zwei **absolute Variable** (Variable mit genau festgelegter Adresse) im Adressbereich 800000 bis 803FFF. Über

diese Variablen können wir Daten bzw. Befehle an den LCD-Controller senden.

Für Zugriffe auf das HD44780-Befehlsregister muss RS=0 (A1=0) sein, für Zugriffe auf das HD44780-Datenregister muss RS=1 (A1=1) sein. Wir legen daher folgende Variablen fest:

```
#define LCD_CNTL MVAR (unsigned char,0x800000)
#define LCD_DATA MVAR (unsigned char,0x800002)
```

LCD_CNTL ist eine absolute `char`-Variable (8 bit) an der Adresse 0x800000 (A1=0), LCD_DATA eine absolute `char`-Variable an der Adresse 0x800002 (A1=1). Eine Zuweisung auf die Variable `LCD_CNTL` bewirkt ein Schreiben einer Instruktion in das HD44780-Befehlsregister, eine Zuweisung an die Variable `LCD_DATA` schreibt Daten in das Datenregister des HD44780.

Beachten Sie: für die beiden absoluten Variablen könnten auch andere Adressen im Adressbereich 800000 bis 803FFF gewählt werden. Voraussetzung ist nur, dass die Adressen stets gerade sind (da wir #WRL für die Generierung des ENABLE-Signals verwenden) und dass die Adressleitung A1 den entsprechenden Wert hat.

Bustiming

Über das BUSCON3-Register wird das Timing der Bussignale für Lese-/Schreiboperationen im durch ADDRSEL3 spezifizierten Adressbereich bestimmt.

Für unser Beispiel stellen wir ein (siehe ***Kapitel 8*** im C167-User Manual):

```
BUSCON3 = 0x0430;
```

Für die Ansteuerung des LCD-Moduls funktionieren auch verschiedene andere BUSCON3-Werte. Beachtet werden muss, dass ein "demultiplexed"-Bus (getrennter Adress- und Datenbus) verwendet wird. Außerdem sind Waitstates erforderlich. Das erklärt sich daraus, dass wir das ENABLE-Signal aus dem #WRL- und dem #CS3-Signal erzeugen. Laut HD44780-Datenblatt sollte ein ENABLE-Signal mindestens 500 ns HIGH sein. Wir bewegen uns auf der sicheren Seite und programmieren die maximale Anzahl von Waitstates (15) und erfüllen so die Mindestforderung, da durch die Waitstates auch die Signale #WRL und #CS3 entsprechend verlängert werden.

Programmablauf

Zunächst werden die Register BUSCON3 und ADDRSEL3 konfiguriert. Anschließend wird der Timer T3 initialisiert. Der Timer T3 wird (im Unterprogramm `wait`) für die Realisierung von Warteschleifen verwendet. Nach der Übergabe von Befehlen bzw. Daten sind gemäß HD44780-Datenblatt bestimmte Mindest-Wartezeiten einzuhalten, wenn der Abschluss einer internen Operation nicht über das Busy-Flag abgefragt wird. **T3CON** wird mit dem Wert **0x0002** beschrieben. Damit wird festgelegt, dass das Timer-Register T3 in Abständen von 1.6 μs inkrementiert wird, wenn der Timer über T3R=1 gestartet wurde.

Als nächstes folgt die Software-Initialisierung des LCD-Moduls. Dazu wird dreimal der Wert 0x0038 in das Befehlsregister geschrieben, wobei nach jedem Schreiben bestimmte Wartezeiten einzuhalten sind.

Danach ist das LCD-Modul bereit, verschiedene Befehle entgegenzunehmen. In unserem Programm werden folgende Befehle gesandt (siehe Datenblatt):

- Initialisierung (Busbreite, Anzeigeart)
- Display ausschalten
- Display löschen
- Display einschalten

Die entsprechenden Operationen werden durch Einschreiben bestimmter Werte in das Befehlsregister (über die absolute Variable LCD_CNTL) programmiert.

Nun können die Datenbytes übertragen werden. Das Beispielprogramm enthält dazu das Unterprogramm

```
void write_string(char * s, unsigned char line)
```

Der Parameter s ist ein Pointer auf den String, der angezeigt werden soll. line beschreibt, in welche Zeile des Displays der Text geschrieben werden soll. Für den Zeilenwechsel muss die Startadresse des Display Data RAMs gesetzt werden, was durch die vier `if`-Anweisungen geschieht. Nach dem Schreiben eines Zeichens wird der interne Adresszähler automatisch inkrementiert. Das Setzen der DDRAM-Adressen ist somit nur von Zeile zu Zeile erforderlich.

Bemerkung zum Timing

Die Ausführung der HD44780-Operationen dauert unterschiedlich lange. Ein nachfolgender Befehl darf erst übertragen werden, wenn der vorangegangene abgeschlossen ist. Da wir auf die Abfrage des Busy-Flags verzichten (dazu wäre anstelle des 74HC541 ein bidirektionaler Transceiver-IC erforderlich), muss nach jedem Befehl eine Warteschleife ablaufen. Die Ausführungszeiten der HD44780-Operationen hängen im wesentlichen von der Taktfrequenz ab, mit der dieser Controller betrieben wird. Diese Taktfrequenz ist nicht bei allen Modulen einheitlich. Im HD44780-Datenblatt können Sie die Ausführungszeiten der einzelnen Befehle bei gegebener Taktfrequenz nachlesen. Der Einfachheit halber wurde im obigen Beispielprogramm einheitlich eine Wartezeit von ca. 4.8 ms nach jedem Befehl und 48 μs nach dem Schreiben eines Wertes in das DDRAM programmiert. Diese Wartezeiten sind zum Teil stark überhöht (können beliebig lang gewählt werden) und sollten für die gängigen 44780-basierenden LCD-Anzeigen passen. Anhand des 44780-Datenblattes und entsprechend der Taktfrequenz des LCD-Controllers können Sie die Parameter in den wait-Aufrufen für Ihr Modul auf Wunsch herabsetzen.

Ergänzende und weiterführende Literatur und Web-Sites zum Thema des Artikels

[1] Datenblatt zum Hitachi LCD-Controller HD44780
http://semiconductor.hitachi.com/search/tree/

[2] Arbeiten mit C166-Controllern, Karl-Heinz Mattheis u. Steffen Storandt, Traunreut, Feger Hard- und Software-Verlag, 1995. ISBN: 3-928434-26-8

[3] Datenblatt zum 74HC541
http://www-eu.semiconductors.philips.com/logic/
http://www.fairchildsemi.com/catalog/Digital_Buffers.html

[4] Erfolgreich Starten mit dem Infineon C167CR-Starterkit und dem Software-Entwicklungssystem von KEIL, Walter Waldner 1999. Verfügbar auf *http://www.htblmo-klu.ac.at/lernen/*

[5] Generieren des Target-Monitors für das Phytec-kitCON-167-Board und die Keil-Toolkette (Infineon C167-Starterkit), Walter Waldner, 1999.
Verfügbar auf *http://www.htblmo-klu.ac.at/lernen/*

[6] Umfassende Informationen über die Infineon-Mikrocontroller finden Sie auf der Web-Seite
http://www.infineon.com/microcontrollers/

[7] Das Internet-Angebot der Firma KEIL finden Sie unter den Adressen *http://www.keil.com/*
http://www.keil.com/~market/

Mikrocontroller HTL-hl Board

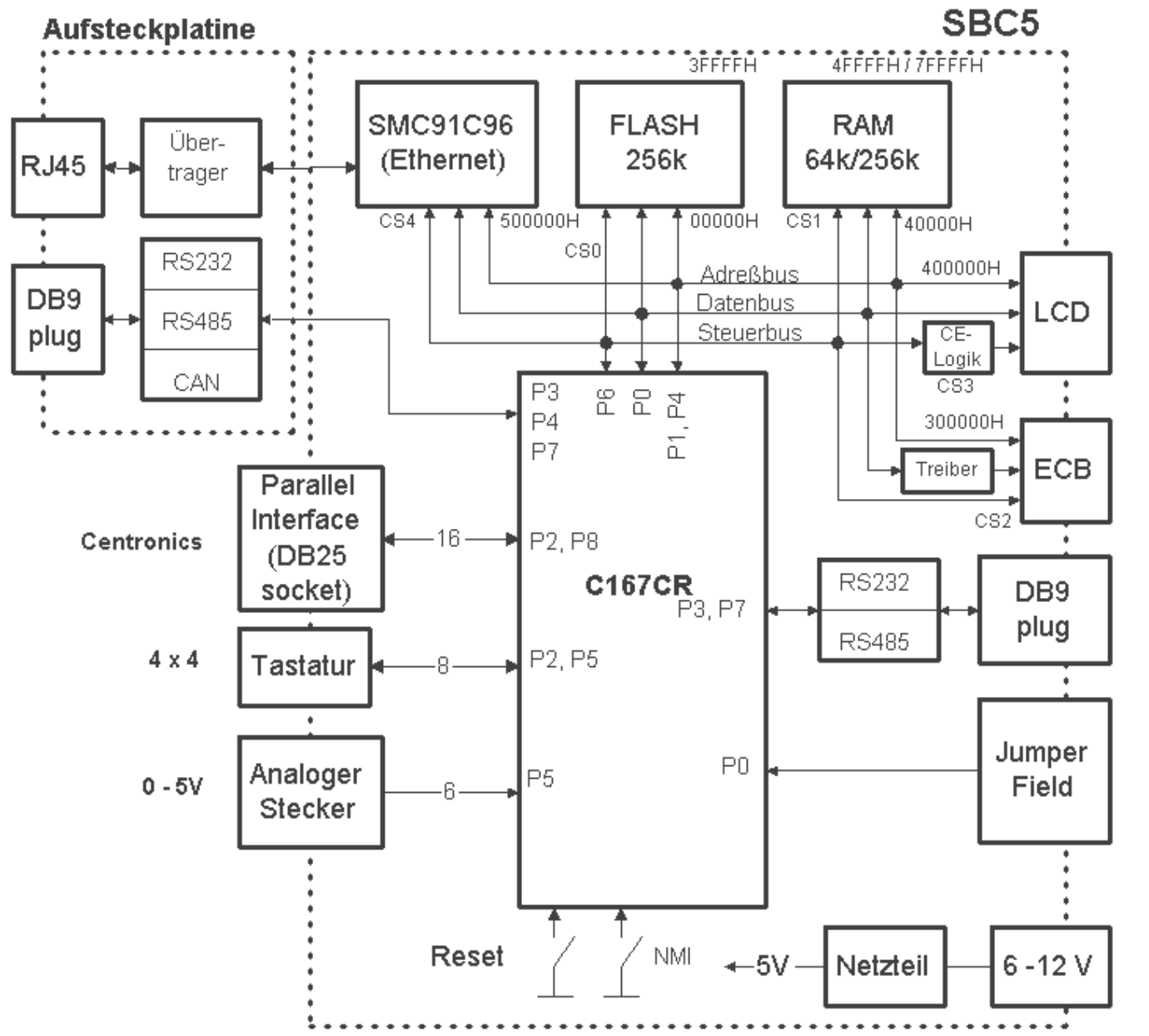
SBC5 (C167CR)

Manfred Resel, Projektteam: Peter Brauneis, Andreas Hummer

In der HTL Hollabrunn wird nun schon seit etwa 10 Jahren das selbst entwickelte SBC3 Mikrocontrollerboard mit einem 80C552 (einem 8051/C500-Derivat von Philips) von jedem Schüler des 4. Jahrganges der Abteilung für Regelungstechnik gebaut. Es war daher an der Zeit, eine Neuentwicklung mit einem schnelleren, leistungsfähigeren Controller und einem optimierenden C-Compiler, der unter Windows läuft, zu beginnen. Herr Ing. Brezovits von Siemens gab schon vor Jahren bei einem Lehrerfortbildungsseminar die Anregung, den C167CR zu verwenden. Da er uns eine Rolle C167CR, 4 Stück Phytec Entwicklungsboards und eine Vollversion des Keil C-Compilers zur Verfügung stellte, war die Entscheidung gefallen. Unsere Aufgabe war es, eine doppelseitige Europakarte mit diesem RISC-Controller zu entwickeln. Beim Phytec-Board kann man über einen Stecker mit Zusatzplatinen eine Peripherieerweiterung vornehmen. Wir sollten im Gegensatz dazu "OnBoard-Schnittstellen" entwickeln, die für Lehr- und Übungsgeräte im Unterricht Standard sind, wie z.B. einen Centronics-Stecker und einen RS485-Feldbusanschluss. Außerdem wurde auch noch ein 16-Bit-Ethernet-Controller auf unserer Platine implementiert. Da viele Bauelemente nur mehr in SMD-Versionen verfügbar sind, erwies sich die Prototypfertigung als "schwierige Lötübung".

Wie man dem Blockschaltbild entnehmen kann, befinden sich auf der Endversion nun folgende Komponenten: ECB-, Centronics-, Tastatur-, Analoge-, LCD- und CAN-Schnittstelle, ein Piezo als Signalgeber sowie 2 Flash- und 2 SRAM-Speicher, der Ethernetcontroller S91C96 mit RJ45-Schnittstelle und natürlich der C167CR. Um die Schnittstellen alle auf

Blockschltbild



Uni-Linz setzt auf EXBO

Anton Kral

Herr Ing. Anton Kral bedankt sich bei Herrn Ing. Wilhelm Brezovits für die gute Kooperation zwischen der Universität Linz und der Siemens AG bei der Abhaltung des 2. Teils der Vorlesung Mikrocomputertechnik an der Universität Linz

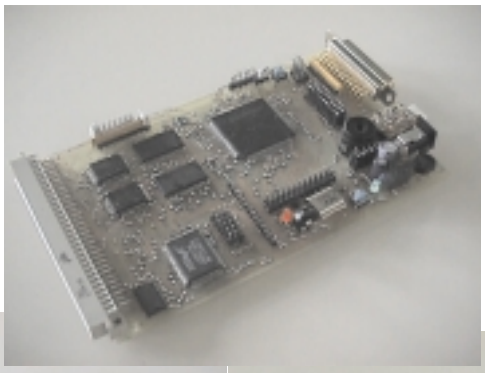
Die Studenten: Im Hintergrund von links nach rechts :Eberl, Chladek, Schlager; im Vodergrund:Krug, Deichstetter, Pfeiffer

Am Institut für Praktische Informatik, Gruppe Systemsoftware werden in den Mikrocomputer-Übungen die C167CR Mikrocontrollerstarterkits und die EXBO-Boards eingesetzt. Die Studenten haben die Aufgabe, eine menügesteuerte Ansteuerung von LED-Lauflichtern und eine Anzeige von gemessenen Analog-Werten zu implementieren.

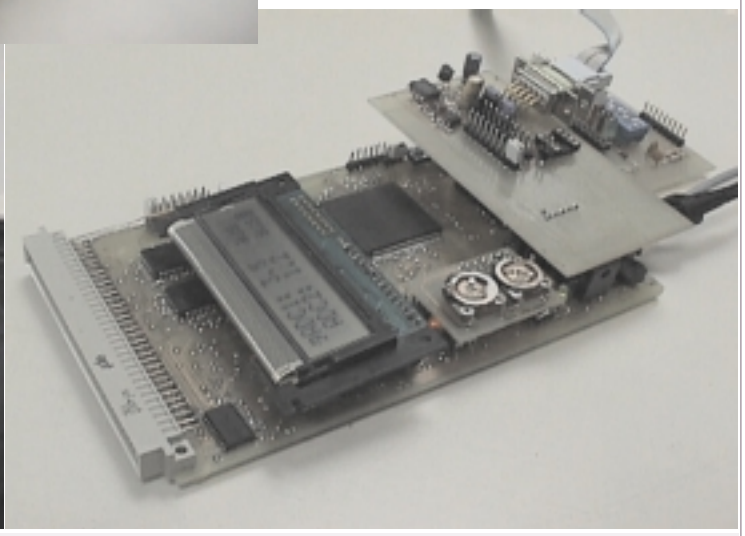
Es stehen 4 Stück EXBO-Boards für die Studenten zur Verfügung, die vom Techniker der Gruppe Herrn Ing. Kral in Eigenregie hergestellt wurden.

der Frontseite zur Verfügung zu haben, haben wir eine Aufsteckplatine entworfen, auf der sich CAN- und Ethernet-Schnittstelle befinden.

Als nächste Entwicklungsschritte sind die Einbindung eines Echtzeitkerns und die Internetanbindung geplant.



Peter Brauneis und Reinhard Hummer (v.l.)



SBC5 mit Peripherie: ECB-Schnittstelle, Ethernet-Chip, Analog Eingänge, ECB-Datentreiber LCD-Steckplatz, Quarz, 2-zeiliges LCD, CAN-Schnittstelle RS232 / 2 RS485

CAN-BUS-IMPLEMENTIERUNG

Christian Böck, Jürgen Moser (Projektteam)
Manfred Resel (Projektbetreuer)

Im Schuljahr 97/98 wurde an der Abteilung für Regelungstechnik der HTBL Hollabrunn außer den Feldbussprojekten Interbus-S und Profibus-DP mit Spezial ASICs auch ein CAN-Bus-Projekt durchgeführt.

Leider gab es dabei zwischen dem neuen 167CR-CAN Board und dem Windows CAN-Analysator Kompatibilitätsprobleme wegen der Übertragungsgeschwindigkeit.

Unsere Aufgabe war es daher, die dort verwendeten SAJ 200 bzw. SAJ 1000 Bausteine durch einen Intel 82527 zu ersetzen. Dieser Chip besitzt nämlich eine sehr ähnliche Registerbelegung wie der integrierte CAN-Controller des 167CR. Dies machte allerdings eine völlige Neuentwicklung sowohl der 8-Bit ECB-Platine (von Christian Böck) als auch der PC-ISA-Karte (von Jürgen Moser) notwendig. Außerdem sollten wir mit dem Phytec 167CR-Evaluation-Board ein eigenes CAN-Analysator-Programm für Windows entwickeln.

Das größte Problem war die Implementierung des 82527 auf der PC-Karte. Zum Programmieren der Register benötigt dieser Controller nämlich einen 256 Byte großen Adressraum, und würde daher den kompletten ISA-Bus (von 200h bis 3FFh) belegen. Aus diesem Grund wurde ein 8-fach-Latch (74LS273) zum Zwischenspeichern der Adresse für den CAN-Controller verwendet. Man benötigt nun für einen Zugriff auf den 82527 2 Zyklen, aber nun braucht die PC-ISA-Karte nur noch 2 Byte (!) im ISA-Adressraum für die vollständige Programmierung.

Neben einer Windows-Visualisierung wurde auch noch ein bedienerfreundlicher Instrumententreiber entwickelt, um das Programmieren des CAN-Controllers zu erleichtern.

Die oben angesprochene 8-Bit-ECB-Platine (mit 82527 onboard) wird über eine Backplane mit einem bereits vorhandenen 8-Bit-Mikrocontroller-Board in der Sprache C51 programmiert.

Ein mit einem Keil C-Compiler geschriebenes Programm für das 167CR-CAN Board und der dazugehörige Windows-CAN-Analysator (ebenfalls von Christian Böck) dienen nun zur Überwachung und Darstellung des Datenverkehrs zwischen dem ECB-Board und der PC-ISA-Karte.

Nach je 400 Stunden intensiver Arbeitszeit ist es uns dann gelungen, die komplette Hard- und Software für ein funktionierendes CAN-Netzwerk zu realisieren.

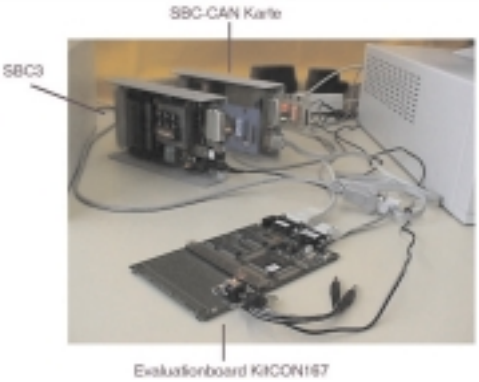
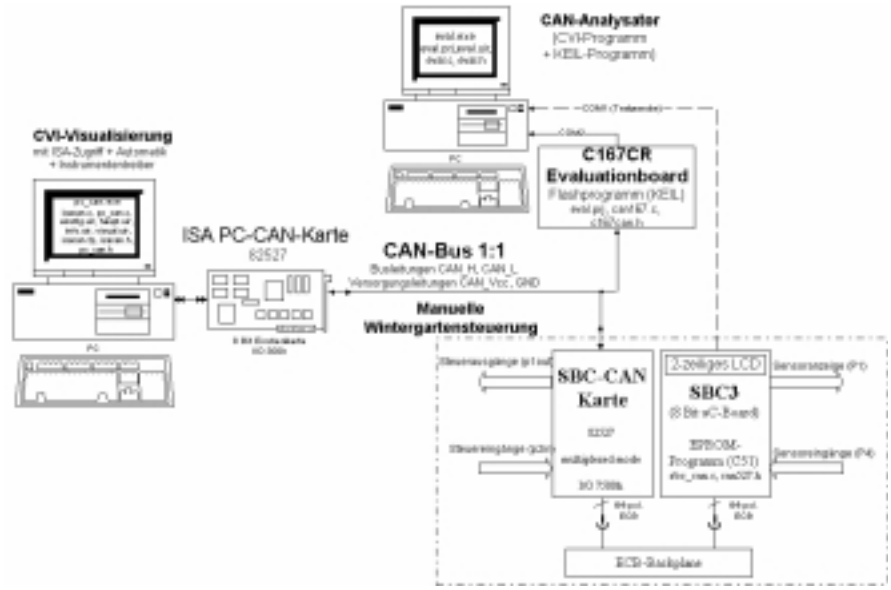
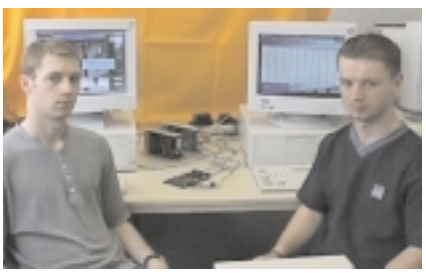


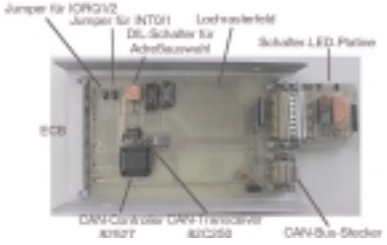
Foto des Aufbaus der Platinen (SBC3, SBC-CAN Karte, Evaluationboard)



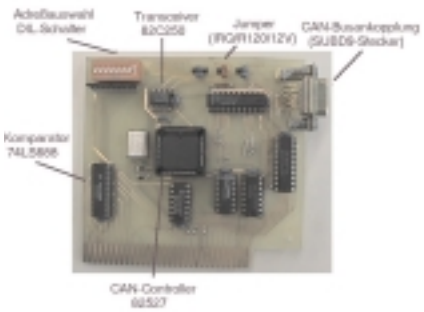
Blockschaltbild des gesamten Aufbaus



Aufbau mit Visualisierung



SBC-CAN Karte



PC-CAN Karte

PHYTEC Starterkit C167CR:

Entwickeln und Testen von Programmen im RAM

Hermann Krammer

Das Speichern von Anwenderprogrammen im Flash ist eine sehr vorteilhafte Sache. Zum Entwickeln und Testen von kurzen Programmen ist jedoch der Ladezyklus (starten im Bootstrap-Mode, Flash-Tools hinunterladen, Flash löschen und programmieren, Programm starten) etwas langwierig. Das Testen von kurzen Programmen im RAM ist eindeutig bequemer. Steht kein geeigneter Hardware-Debugger zur Verfügung, so bietet sich folgende Methode an:

Wir installieren im Flash einen Minimonitor (MM), der in Zusammenarbeit mit einem auf dem PC laufenden Terminalprogramm Intel-Hex-Files ins RAM hinunterladen und starten kann. Dieser Minimonitor geht dabei folgendermaßen vor:

Die auf dem Mikrokontroller C167 zur Verfügung stehenden Chip-Selects 1 bis 4 definieren 4 Fenster im Adressbereich, die in ihrer Größe programmiert werden können. Chip Select 0 deckt den übrigen Adressbereich ab. Beim PHYTEC-Starterkit hängt am Chip Select 0 der Flash-Speicher mit 256 kByte und an Chip Select 1 das RAM mit 64 kByte. Das System ist zu Beginn so konfiguriert, dass das RAM ab Adresse 04:0000, also im Segment 4 liegt. Zunächst wird das Intel-Hex-File ins RAM geladen. Durch den Befehl G (Go) wird das RAM auf Adresse 00:0000 gelegt, also ins Segment 0 eingeblendet, und das Programm wird gestartet. Nach einem Reset gelangen wir zurück in den Minimonitor. Das Programm bleibt im RAM, das jetzt wieder im Segment 4 liegt, gespeichert.

Der Minimonitor MM wurde in C geschrieben und mit dem TASKING-Compiler übersetzt. In der Locator-Datei wurde der im Segment 0 liegende Flash-Bereich mit Ausnahme der Interrupt-Tabelle ausgeblendet, sodass das Programm in das Segment 1 loziert wird. Das ausführbare Programm im Intel-Hex-Format `MM.H86` wird mit `FLASHT.EXE` installiert und bleibt dann im Flash-Speicher erhalten. Nach einem Reset meldet es sich mit #. Es arbeitet mit 57600 Baud.

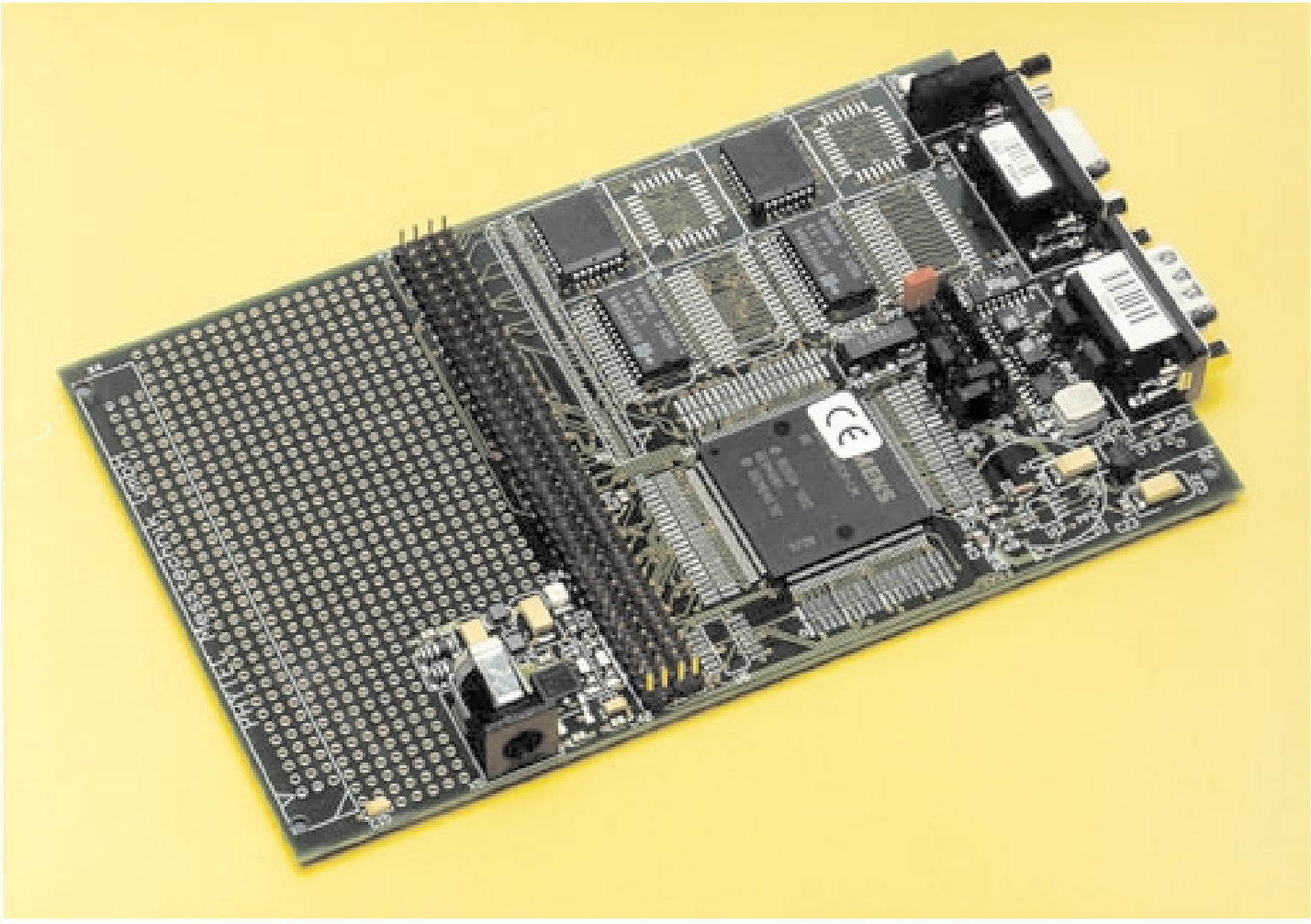
Der Sourcecode des Minimonitors steht als `MM.ZIP` in der Homepage der HTL Braunau zur Verfügung:

http://www.asn-linz.ac.at/schule/htlbraunau/lehrer/krammer/index.htm

Literatur

User´s Manual C167 Derivatives, Siemens AG München 1996

KitCON-167 Hardware-Manual, PHYTEC Messtechnik GmbH Mainz 1997



Windkraft und Mikrocontroller

Andreas Thieme

Das in Völkermarkt ansässige Unternehmen Windtec Anlagenerrichtungs- und Consulting GmbH entwickelt, produziert, verkauft und betreibt weltweit Windkraftanlagen bis 1,5MW. Gewonnen wird die elektrische Energie aus dem Wind und in das öffentliche Netze eingespeist. Eine Windtec 1566 kann an einem guten Standort in Österreich (mittlere Jahreswindgeschwindigkeit 6,2m/s) 3000MWh elektrische Energie pro Jahr erzeugen, das entspricht dem Verbrauch von ca. 600 Haushalten im gleichen Zeitraum. Das Potential an nutzbarer Windkraft in Österreich liegt bei etwa 15% des jährlichen Stromverbrauches von derzeit 50.000 GWh, 0,1% werden derzeit genutzt.

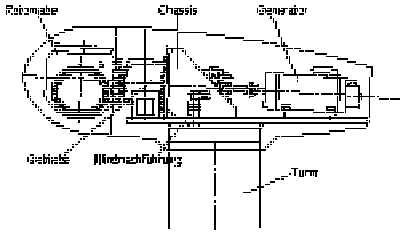


Abbildung 1: Schematische Ansicht der Anlage inklusive Bezeichnung

WT1566, 1,5MW Windkraftanlage,: 60m Turm, 66m Rotorkreisdurchmesser

Windtec Windkraftanlagen sind Dreiblattanlagen mit horizontaler Achse, die speziell für die effiziente Nutzung der Windkraft an Binnenlandstandorten entwickelt wurden. Sie verfügen über eine variable Rotordrehzahl und können somit im Teillastbereich, d.h. von 3,5m/s bis 11,5 m/s, mit optimalem Wirkungsgrad betrieben werden. Bei einer Windgeschwindigkeit von 11,5 m/s erreichen die Anlagen ihre Nennleistung. Die Kombination aus elektrischer Drehmomentregelung und Rotorblattverstellung erlaubt einen Betrieb der Anlage bei Windgeschwindigkeiten von 11,5m/s bis 25,0m/s mit konstanter Nennleistung.

Die wesentlichen Vorzüge von Windtec Windkraftanlagen sind:

- Neue drehzahlvariable Leistungselektronik mit hohem Wirkungsgrad, Leistungsfaktorregelung ohne Flicker- und Oberwellenbelastung für das Netz.
- Wesentlich verbesserte Wirtschaftlichkeit gegenüber dem Stand der Technik
- Rotorblatt mit integriertem Blitzschutz und lärmoptimierten Blattspitzen.

Zum drehzahlvariablen Betrieb der Anlage verfügen Windtec Windkraftanlagen über eine doppeltgespeiste Drehstrommaschine (DDM). Diese besteht aus einem Asynchrongenerator und einem IGBT-Umrichter, der den Rotorkreis des Generators mit variabler Frequenz und Spannung erregt. Durch den Einsatz einer doppeltgespeisten Drehstrommaschine ergeben sich folgende wesentliche technische Vorteile:

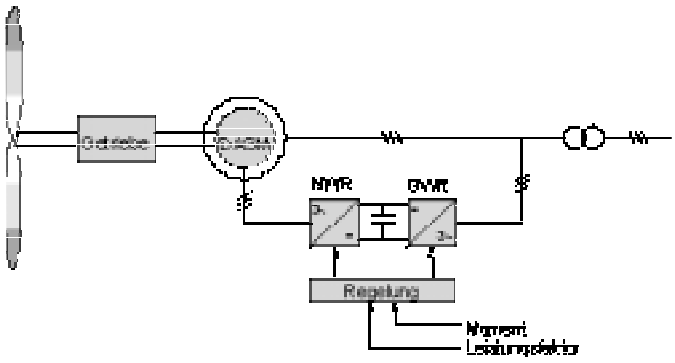


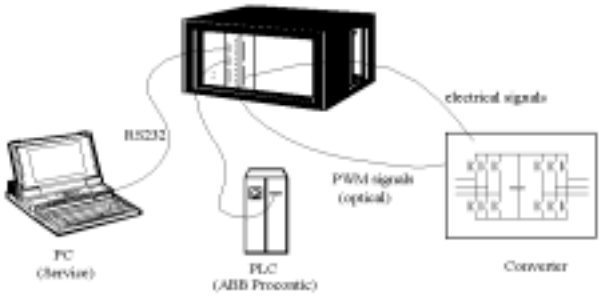
Abbildung 2: Übersicht Generatorsystem

- hoher elektrischer Wirkungsgrad
- Reduktion der Oberwellenbelastung auf eine kaum messbare Größe.

Die Abgabeleistung und der Blindleistungsfaktor (cos j) können über den gesamten Leistungsbereich stufenlos, entsprechend externer Sollwertvorgabe, oder mit einstellbarem Fixwert geregelt werden.

Andreas.Thieme@mail.windtec.com Andreas Thieme

Was haben moderne Windkraftanlagen mit Leistungen bis zu 1,5MW und moderne 16 Bit Mikrocontroller von Siemens/Infineon gemeinsam?

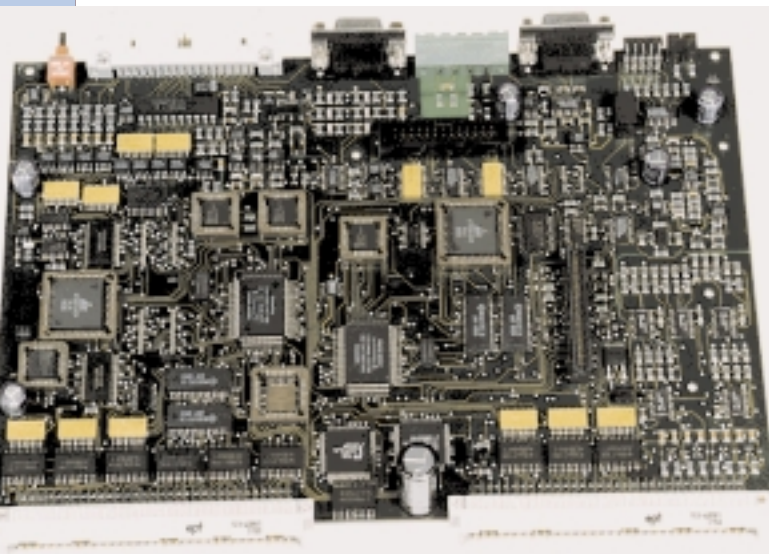


Die neuentwickelte Steuerelektronik besteht aus einem modularem 19" System, dessen Kernstück die Controllerkarte mit 2 Siemens/Infineon SAB 88C166-5M Mikrocontrollern ist. Besonderes Augenmerk wurde auf die Störfestigkeit der Signalverarbeitung gelegt, so werden z.B. die Leistungshalbleiter über Lichtwellenleiter optisch angesteuert.

Der Einsatz von 2 Mikrocontrollern begründet sich durch die Notwendigkeit, für den 4 Quadranten Umrichter 2 Ausgangsspannungen generieren zu müssen. Dafür wird die CAPCOM-Einheit der Mikrocontroller im Dual Compare Mode verwendet. Die beiden Controller können über ein Dual Port RAM Daten austauschen.

Es wird ein nicht gemultiplexter Adress- und Datenbus verwendet. Zusätzlich zum internen Speicher wurden externer 32kByte Flash-EPROM und 64kByte RAM aufgebaut. Die seriellen Schnittstellen der beiden Prozessoren werden zum Einen für den Anschluss eines seriellen EEPROM's zur Sicherung der Anlagendaten und zum Anderen zur Kommunikation mit einem PC verwendet. Die Kanäle des internen ADC werden zur Erfassung der Spannungen und Ströme verwendet. Als Option können auch externe AD-Wandler mit höherer Auflösung verwendet werden.

Die Software wurde in der Programmiersprache C erstellt, als Compiler wird der Tasking Compiler eingesetzt.



Wilhelm Brezovits (Siemens-Bauelemente), DI. Andreas Thieme (Windtec) und Ing. Hermann Sailer (Rekirsch-Toolpartner)

Derzeit wird bei Firma Windtec an dem Redesign der Prozessor-Platine gearbeitet. Im Wesentlichen wird dabei der Übergang zu dem Siemens/Infineon SAB C167CR-16FM / SAB C167CS-32FM vollzogen. Hierbei ist die Unterstützung durch den Herrn Brezovits von Siemens als besonders positiv hervorzuheben. Durch seinen persönlichen kompetenten Einsatz konnte in kurzer Zeit sowohl die Auswahl des geeignetsten Derivats abgeschlossen werden, als auch die Zustellung der notwendigen Dokumentation erfolgen. Für die Softwareentwicklung kommt ein Emulator der Fairma Hitex zum Einsatz. Der Support erfolgt durch die Wiener Firma Rekirsch Elektronik.

WT646, 600kW Windkraftanlage, 50m Turm, 46m Rotorkreisdurchmesser, 1220 Wien, Wagramer Straße (bei der Mülldeponie)

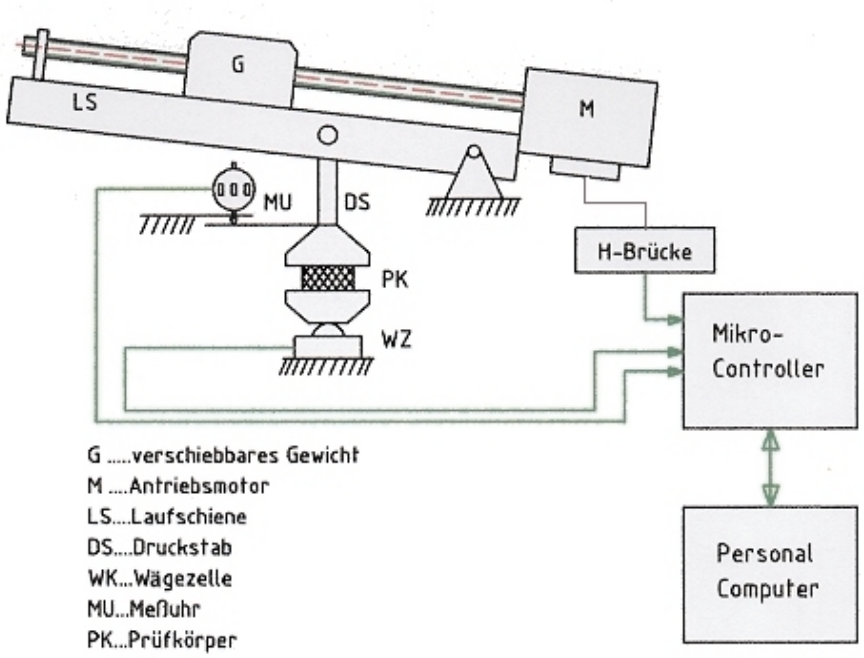
Andreas Thieme Andreas.Thieme@mail.windtec.com

Dauerprüfeinrichtung für Elastomere

Ingenieurprojekt zur Erlangung der Reifeprüfung an der HTL Bregenz – Abteilung Maschinenbau-Automatisierungstechnik.

Christoph Albiez, Stefan Böhler, Klaus Dietrich, Alexander Wörz (Projektteam)
Werner Tomaselli (Projektbetreuer)

Das mechanische Verhalten von polymeren Kunststoffen ist bis zum heutigen Tag nicht bzw. nur unvollständig mathematisch modellierbar. Um gesicherte Aussagen über ihre Eigenschaften machen zu können, bedarf es noch aufwendiger und zeitintensiver Untersuchungen. Von besonderem Interesse war für unsere Auftraggeber – der Firma Getzner Werkstoffe in Bludenz – das Kriech- und Relaxationsverhalten von Elastomeren, die als Dauerschwingungsdämpfer unter Eisenbahntrassen oder im Maschinenfundamenten Verwendung finden sollen. Dazu sollte eine Vorrichtung gebaut werden, die autonom arbeitet und Prüfungen gemäß DIN 53547 an drei voneinander unabhängigen Prüfständen ermöglicht.



G .....verschiebbares Gewicht
M ....Antriebsmotor
LS....Laufschiene
DS....Druckstab
WK...Wägezelle
MU...Meßuhr
PK...Prüfkörper

Prizipieller Aufbau des Prüfstandes

Aufbau

Auf einer drehbar gelagerten Laufschiene ***LS*** befindet sich ein motorisch verschiebbares Gewicht. Dadurch wird eine variable Druckkraft über den Druckstab ***DS*** auf den Prüfkörper ***PK*** aufgebracht, die den Prüfling mechanisch verformt. Die Verformung wird von einer digitalen Messuhr erfasst und über eine asynchrone Schnittstelle an den Mikrocontroller weitergegeben. Die Druckkraft wird von einer Wägezelle auf DMS-Basis aufgenommen und als Analogsignal an den Mikrocontroller weitergeleitet.

Prüfprogramm

Nachdem der Prüfkörper zwischen die Druckplatten gelegt worden ist, fährt das Gewicht in jene Position, die einer definierten Anfangsbelastung entspricht und die Messuhr nullt. Danach werden in annähernd logarithmischen Zeitabständen beginnend mit etwa 2 Sekunden die Verformung und die Druckkraft gemessen und aufgezeichnet. Nach einer vorgegebenen Zeitdauer, die im Extremfall ca. 1000 Stunden beträgt, muss das Gewicht an eine neue Position verfahren und der Messvorgang wiederholt werden.

Zahl und Dauer der Belastungsfälle ist für jeden Prüfling zu Beginn der Messung festzulegen. Da diese Messungen sehr zeitintensiv sind, ist ein solcher Test nur mit einer vollautomatischen und autonom arbeitenden Messstation effektiv und wirtschaftlich möglich. Mit autonom ist gemeint, dass die Steuerung eigenständig Entscheidungen trifft und entsprechend den Messparametern Einstellungen vornimmt. Wir haben uns für eine Mikrocontrollersteuerung entschieden und hierfür den 16-Bit-μC C167CR der Firma Infineon eingesetzt.

Mikrocontroller

Primäre Aufgabe des Mikrocontroller ist die Erfassung der Messdaten und deren Speicherung. Zur Ansteuerung der Messuhren wird über die simulierte asynchrone Schnittstelle des C167 und einen nachgeschalteten Multiplexer eine ASCII-Befehlssequenz übertragen, worauf die angesprochene Messuhr in entsprechender Weise antwortet. Die Messwerte werden in Form von BCD-kodierten Zahlenfolgen gesendet. Die Kraftsignale gelangen über einen Umformer an die Analogeingänge des Mikrocontrollers, dessen A/D-Wandler für die gegebene Situation ausreichend genau arbeitet.

Die im C167 integrierte Pulsweitenmodulation ermöglicht eine einfache Ansteuerung der Gleichstrommotoren. Durch ein variables Puls-Pausen-Verhältnis, dessen Wert in Abhängigkeit von der Zeit S-förmig von Null zum Maximalwert ansteigt bzw. im umgekehrten Fall abfällt, kann ein sanftes, ruckfreies aber dennoch zügiges Anfahren oder Abbremsen des Verstellgewichts erreicht werden.

Zusätzlich ist über den Mikrocontroller die Kommunikation mit einem fallweise angeschlossenen PC abzuwickeln, der die Prüfparameter überträgt und die gesammelten Messergebnisse übernimmt und weiterverarbeitet. Besonderes Augenmerk wurde hierbei auf eine sichere Datenübertragung gelegt. Die Datensicherung mittels Paritätsbit erschien uns zu unsicher, weshalb wir uns dafür entschieden, jedes Datenpaket mit einem 8-Bit-CRC-Glied abzusichern. Die Steuerung des Datentransfers erfolgt mittels eines Softwarehandshake-Verfahrens durch Übertragung von Acknowledge- bzw. Not-Acknowledge-Signalen über die asynchrone RS232-Schnittstelle.

Mikrocontroller an der FH Kapfenberg

Helfrid Maresch, Peter Hintenaus

Das Fachhochschulgebäude in Kapfenberg

Mikrocontroller an der FH Kapfenberg

Der Fachhochschul-Studiengang „Industrielle Elektronik“ bildet in Kapfenberg seit 1995 Studenten in den Bereichen Schaltungs- und Geräteentwicklung, Informatik und Automatisierungstechnik aus. Das 8-semestrige Studium ist stark interdisziplinär ausgerichtet und umfasst neben einem breiten Spektrum von technischen Fächern auch eine solide wirtschaftliche Ausbildung. Geräte und Komponenten müssen nicht nur auf das Preis-Leistungverhältnis und ihre Wirtschaftlichkeit in der Produktion optimiert werden, auch das Marketing darf nicht zu kurz kommen. „Soft skills“ wie Kommunikation, Teamtraining und eine Präsentations- und Verhandlungsschulung in Englisch runden die Ausbildung ab.

Trotz der Breite der Ausbildung ist auch Raum für eine Spezialisierung. Vor allem im 3. und 4. Studienjahr bieten Projekte, ein Berufspraktikum im 7. Semester und eine Diplomarbeit die Möglichkeit zur Vertiefung. Das Spektrum reicht von der Automatisierung über die Geräteentwicklung bis zum FPGA- und ASIC-Design. Dazu stehen modernste Werkzeuge wie z.B. Mentor Graphics zur Verfügung. Software ist in allen Bereichen ein selbstverständlicher Bestandteil.

Die ersten Absolventen, die im Sommer 1999 das Studium abschließen, finden ausgezeichnete Jobangebote vor. Da auch die Diplomarbeiten zumeist in Firmen durchgeführt werden, bringen sie mit Studienabschluss bereits eine einjährige Berufserfahrung und eine weitgehende Methoden- und Sozialkompetenz mit, sodass sich Einschulungszeiten sehr kurz gestalten. Ein großer Teil der Studierenden nützt diese Zeit, um im Ausland (USA, Frankreich, Deutschland) Erfahrungen zu sammeln.

Der Studiengang wird von einem privatwirtschaftlich organisierten Träger geführt, der Technikum Joanneum GmbH. Diese Firma ist der größte Anbieter von Fachhochschul-Studiengängen in Österreich, mit 6 Studiengängen in Graz und 3 in Kapfenberg. Ein weiterer Ausbau dieses Hochschulzweiges, der vor allem Technik und Wirtschaft in einer berufsorientierten, akademischen Ausbildung verbindet, ist geplant.

Studenten- und Industrieprojekte mit Infineon Mikrocontrollern

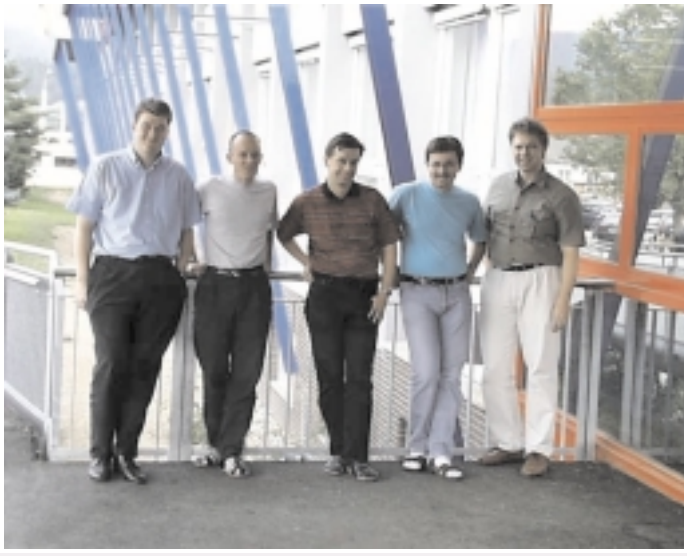
Im letzten Studienjahr wurden von unseren Studenten folgende Geräte entwickelt:

- Kompakter CAN–Analyzer auf Basis C167
- Barcode Lesestift mit Funkübertragung auf Basis C504
- Steuerung für einen magnetostriktiven Aktor auf Basis C167
- Steuerung für eine Solaranlage auf Basis C504
- Netzwerkinterface Ethernet auf achtfach RS485 basierend auf C167, für Anwendungen in der Lagerautomation

Diese Themen stehen in engem Zusammenhang mit den am Transferzentrum für Industrielle Elektronik laufenden Industrieprojekten. Die Studenten können auf zwei Logikanalysatoren der Firma HP und je einen Emulator für die C166-Familie sowie für den 8051/C500 und seinen Derivaten uneingeschränkt zugreifen.

Neben ihren Aufgaben in der Lehre sind die Professoren im Rahmen des Transferzentrums mit Projekten für Kunden aus der Industrie beschäftigt. Schwerpunkte sind Mikrocontroller-Anwendungen, digitale Signalverarbeitung, programmierbare Logik sowie Feldbusanbindungen (CAN). Aus den über 30 bereits abgewickelten Kundenprojekten seien erwähnt:

- Steuerungssoftware für Bewässerungsanlagen auf Basis C167
- Motor- und Kommunikationssteuerung für einen motorintegrierten Frequenzumrichter auf Basis je eines C504
- CAN-Bus-Software für Wegaufnehmer und serielle Klemmen auf Basis C165



Das Mikrocontrollerteam des Studiengangs, von links: DI Klaus Gebeshuber, DI Michael Salloker, Dr. Robert Okorn, Dr. Peter Hintenaus, DI Christian Netzberger

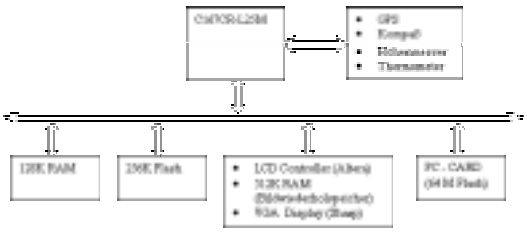
Elektronische Wanderkarte

Helfrid Maresch, Peter Hintenaus

Die Elektronische Wanderkarte wird als internes Forschungs- und Entwicklungsprojekt des Studiengangs mit Beteiligung der Studenten des dritten und vierten Semesters betrieben. Es handelt sich um ein Navigationssystem, optimiert für Freizeitsportler wie Bergsteiger oder Mountainbiker. Das Gerät zeichnet sich durch folgende Eigenschaften aus:

- Helles TFT Farb-LCD (6.5 Zoll, 640 * 480 Punkte, 256 Farben)
- Genaue und detaillierte Kartendaten (ÖK 50)
- Speicherung der Kartendaten in komprimierter Form auf einer PC-Card (bis zu 64 MByte Flash Memory)
- GPS zur Positionsbestimmung
- Elektronischer Kompass
- Höhenmesser auf Luftdruckbasis
- Elektronisches Thermometer

Hardware



Blockschaltbild

Der Prozessorkern besteht aus einem C167CR-L25M mit 25 MHz Taktfrequenz, 256 KByte Flash zur Speicherung des Programms und 128 KByte batteriegepuffertem RAM. In diesem Speicher werden die Programmdaten und die zurückgelegte Route abgelegt.

Zu den Aufgaben des C167 gehören:

- Lesen, Dekomprimieren und Anzeigen der Kartendaten
- Überwachung der Stromversorgung, einschließlich des Ausschaltens
- Erzeugung der Kontrollsignale für die Sensoren sowie Verarbeitung der Messergebnisse
- Aufzeichnung der zurückgelegten Route
- Benutzerschnittstelle

Da sich die Beschaffung eines geeigneten LCD-Controller-Chips als schwierig herausstellte, wurde der Controller im Haus entwickelt. Aufgrund der Anforderungen, die von dem verwendeten Display gestellt werden – es benötigt alle 40nS die Daten für einen Punkt – kommt nur eine Hardwarelösung in Frage. Unsere LCD Ansteuerung besteht aus einem ALTERA EPLD 7192 (programmierbare Logik) und einem schnellen 256k * 16 statischen RAM, das als Bildwiederholspeicher dient. Der Bildwiederholspeicher ist in den Adressraum des C167 abgebildet. Zugriff auf diesen Speicher ist jederzeit, sowohl byte- als auch wortweise möglich. Die Anzeige wird davon nicht beeinflusst.

Der LCD-Controller hat 3 Schnittstellen:

- LCD: Alle 40 ns müssen hier 8 Bit für den nächsten Punkt zur Verfügung stehen. Aus dieser Anforderung ergibt sich die Wahl des Prozessortakts.
- Prozessorbus: um Pins am EPLD zu sparen, wurde der 16 Bit breite gemultiplexte Bus als Betriebsart gewählt. Über diese Schnittstelle greift der Prozessor auf den Bildwiederholspeicher zu.

Bildwiederholspeicher: Dieser Datenpfad ist 16 Bit breit. Um die Zugriffe vom LCD und vom Prozessor zu arbitrieren, steht der Bildwiederholspeicher abwechselnd für 40 nS dem Prozessor und für 40 nS dem LCD zur Verfügung.

Zusätzlich bietet der LCD-Controller Hardwareunterstützung für horizontales und vertikales Scrollen. Dies wird dadurch realisiert, dass die Speicheradresse im Bildwiederholspeicher, die dem linken oberen Eck des Displays zugeordnet ist, von der Software geladen werden kann.

Software

Die gesamte Software verwendet einen Multitasking-Kernel, der am Studiengang entworfen und realisiert wurde. Die Prozesse bekommen nach dem Round-Robin-Algorithmus den Prozessor zugeteilt. Zur Synchronisation von Prozessen dienen Semaphore.

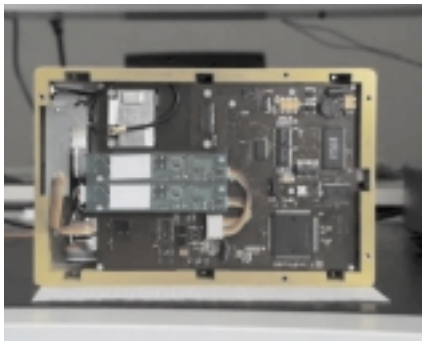
Die Kartendaten werden vom Bundesamt für Vermessungswesen getrennt nach Lagen (z.B. Wald, Gewässer, Grenzen) in Rasterform geliefert. Jede Lage ist ein reines Schwarzweißbild, ohne Graustufen.

Vor dem Speichern auf der Flashkarte werden die Lagen Lauflängen- und dann Huffman-kodiert. Diese komprimierten Daten werden zusammen mit dem Codebaum für den Huffman–Code auf der Flashkarte gespeichert. Die Komprimierung reduziert den Speicherbedarf auf etwa ein Sechstel gegenüber den Ausgangsdaten. Für die Berechnung eines Bildschirminhalts wird ungefähr eine Sekunde benötigt.

Ausblick

Sobald Kartendaten in Vektorformat mit ausreichendem Informationsgehalt erhältlich sind, planen wir folgende Zusatzfunktionen:

- Automatisches Einnorden der Karte
- Panoramadarstellung der Umgebung des jeweiligen Standorts
- Automatische Nachführung des Luftdrucks für den Höhenmesser
- Wetterwarnung bei großen Luftdruckschwankungen
- Zoomen mit automatischem Weglassen feiner Details bei Übersichtsdarstellungen



Vorderansicht und Hardware der elektronischen Wanderkarte

Helfrid.Maresch@fh-joanneum.at Maresch/Hintenaus

BIOMASSE & MIKROCONTROLLER

A&R TECH

Robert Gausterer, Reinhard Radl

Das in Wien ansässige Unternehmen A&R TECH wurde 1990 mit dem Unternehmensziel gegründet, als Ingenieurbüro und Produktionspartner die Wünsche und Ideen der Kunden mit Engagement und Freude sowie mit modernster Technik professionell umzusetzen und in Serie zu produzieren.

A&R TECH ist in Bereichen der • Automatisierungstechnik • Industriellen Kommunikation und Prozessleittechnik • Kälte- und Klimatechnik • Mess- und Prüftechnik • Steuerungs- und Regelungstechnik • Überwachungs- und Störmeldetechnik • Visualisierungstechnik tätig.

A&R TECH ist Österreichrepräsentant der deutschen Firma ***elrest automationssysteme GmbH,*** bei deren Automatisierungskomponenten INFINEON (ehemaliger SIEMENS Bereich Halbleiter) Mikrocontroller der 166er Familie (C167CR) eingesetzt werden.

Was aber haben Biomasse und Mikrocontroller von Infineon gemeinsam?

A&R TECH entwickelt und produziert Steuerungen für das Partnerunternehmen KÖB & SCHÄFER. Das Vorarlberger Spitzenunternehmen verfügt über langjährige Erfahrung und Kompetenz auf dem Gebiet der Biomasseheizverbrennung. Die in ganz Mitteleuropa im Einsatz befindlichen Heizkesselanlagen zeichnen sich durch ein ausgezeichnetes Preis/Leistungsverhältnis und den Einsatz intelligenter und moderner Technik aus:

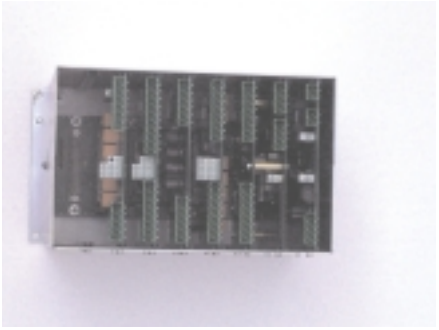
Die Hauptanforderungen an die Steuerung dieser Biomasseverbrennungsanlagen sind:

- Modularer Aufbau zur Abdeckung der unterschiedlichen Kesseltypen
- Höchstmögliche Verfügbarkeit (Zuverlässigkeit) des Systems
- Vernetzbarkeit der Steuerungskomponenten mittels Feldbussystem
- Strukturierte Programmierung ausschließlich in der Hochsprache C
- Optimales Preis-/Leistungsverhältnis

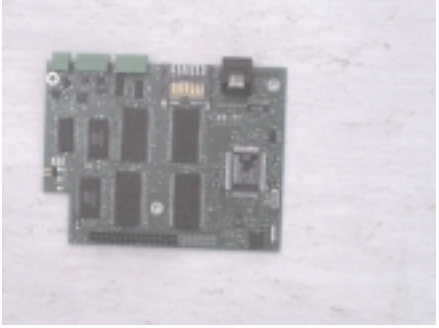
Die Kesselsteuerung wurde als modulares, modifiziertes 19"-Stecksystem mit Busplatine und steckbaren Steuerungsbaugruppen ausgeführt, wobei sämtliche Steckplätze frei belegbar sind. Auf der Busplatine wird nicht der Systembus des µCs, sondern ein gepufferter 8-Bit I/O-Bus mit geographischer Adressierung der Steckplätze geführt. Damit konnte eine höchstmögliche Störsicherheit und leichte Austauschbarkeit der Baugruppen erreicht werden.

Für die Vernetzung der einzelnen Anlagenkomponenten wurde auf den, bereits in anderen Projekten bestens bewährten und sehr störsicheren CAN-Bus zurückgegriffen.

Das Herzstück der Kesselsteuerung bildet die CPU-Platine mit dem Infineon SAB C515C-LM Mikrocontroller. Als Piggypack ist eine 8-Kanal Messverstärkerplatine aufgesetzt. Mit ihr werden die analogen Fühlersignale der verschiedenen Temperaturfühler und einer Lambda-Sonde zur Rest-O2 Bestimmung auf einen für die internen A/D-Wandler des µC geeigneten Wert verstärkt. Die Linearisierung der Fühlerkurven erfolgt durch Softwarealgorithmen.



Je nach Anlagenvariante kann die Steuerung mit einer variablen Anzahl von Ein-/Ausgabebaugruppen bestückt werden. Bei komplexeren Heizanlagen wird die Kesselsteuerungshardware mehrfach eingesetzt, um die notwendigen Zusatzaufgaben wie Wärmeverteilung oder die Beschickung mit Brennstoff zu steuern.



Im Bediengerät der Anlage sitzt die Intelligenz der gesamten Steuerung. Hier erfolgt die witterungsgeführte Programmsteuerung der Heizanlage. Die Eingabe und Anzeige der Heizprogramme und Betriebszustände erfolgen mittels Folientastatur und LC-Display.

Als Mikrocontroller kommt hier ein Infineon SAB C161-RI mit 2MB Flash-EPROM und 256kB SRAM sowie 2kB EEPROM und RTC zum Einsatz. Aus Performancegründen wird ein non-multiplexed 16-Bit Datenbus verwendet. Da das Bediengerät mit einer (oder mehreren) Kesselsteuerungen über den CAN-Bus verbunden ist, wurde ein externer CAN-Controller vorgesehen.

Als besonders vorteilhaft erwies sich die Möglichkeit bei den Infineon 16-Bit µC's den Busmodus (8-Bit/16-Bit, muxed/non-muxed) für jedes Adressfenster getrennt festzulegen (interne Chipselectgenerierung). Dadurch konnte das Hardwaredesign sehr einfach gestaltet werden.

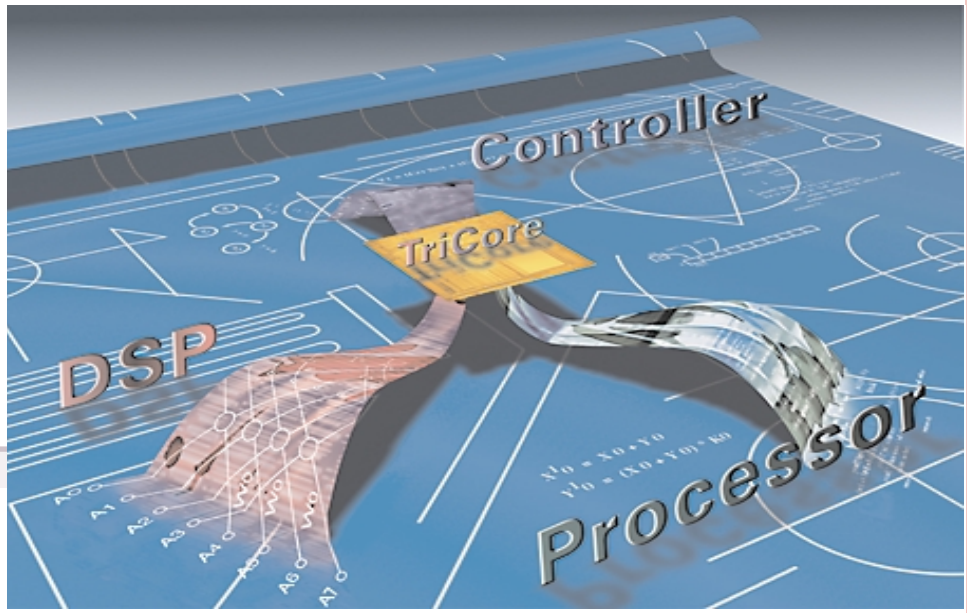
Die komplette Systemsoftware wurde durchgängig in der Hochsprache 'C' mit Hilfe von DAvE und dem Toolkit der Firma Keil erstellt.

Ausschlaggebend für den Einsatz der Infineon Mikrocontroller ist die breite Palette von Derivaten, die CAN- und Hochsprachenunterstützung und nicht zuletzt ein günstiges Preis-/Leistungsverhältnis. Nicht unerwähnt soll auch die umfassende technische Unterstützung durch Firma Siemens in der Person von Herrn Brezovits bleiben.

Robert Gausterer, Reinhard Radl artech@aon.at

TriCore - 32-Bit Power für das nächste Jahrtausend

Renate Schultes



Der Jahrtausendwechsel steht unmittelbar vor der Tür. In vielen Applikationsbereichen der Mikroelektronik laufen die Entwicklungen für die neuen Gerätegenerationen. Immer kleiner, schneller, leistungsfähiger und das bei möglichst wenig Stromaufnahme – das sind die Forderungen an die Entwickler in den verschiedenen Branchen wie:

- Büroautomatisierung (PC, Internet, digitale Kamera, ...)
- Kommunikation (Handy, Videokonferenz, ...)
- Automatisierung (Maschinensteuerungen, Transportsysteme, ...)
- Automotiv (Motormangement, ABS, ASR, Airbag, Navigation, ...)

Um mit diesem Trend Schritt halten zu können, müssen die Chiphersteller immer mehr Funktionen in einem Baustein zusammenpacken. Der Firma Infineon Technologies (bis April 1999 Bereich Halbleiter der Siemens AG) ist dies mit einem neuen Bausteinkonzept für eine 32-Bit Mikrocontroller-DSP-Familie gelungen.

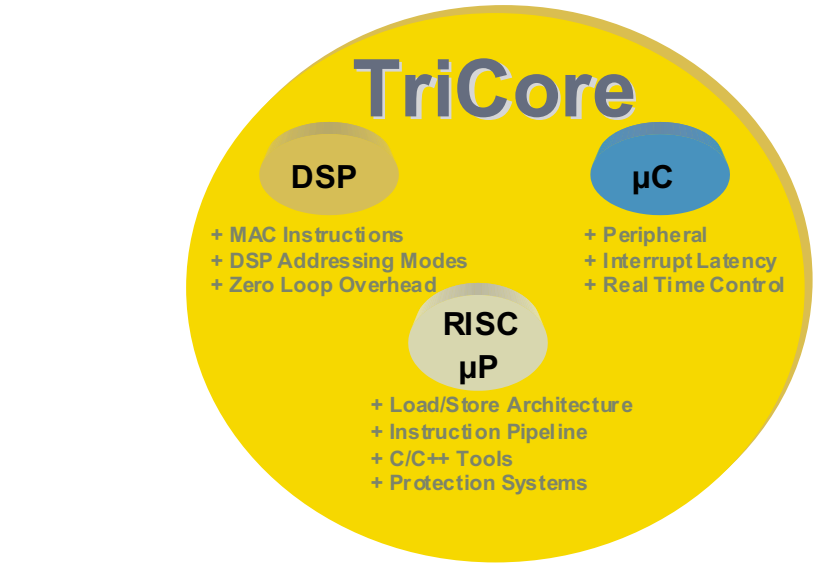


Abb. 1 TriCore - das Beste aus drei Bereichen der Mikroelektronik

Der **TriCore** vereinigt die wichtigsten Eigenschaften aus drei Bereichen der Mikroelektronik auf einem Chip (siehe ***Abb. 1***):

Mikrocontroller-Funktionalität für Echtzeitverarbeitung mit schnellen On-chip Programm- und Datenspeichern, kurzen Interrupt-Latenzzeiten und einer Reihe von leistungsfähigen On-chip Peripherie-Modulen

DSP mit MAC Funktionalität, speziellen Adressierungsmechanismen und Datenformaten

RISC-Architektur mit Load-/Store-Befehlen, vielen Arbeitsregistern, Befehlsabarbeitung in einer Pipeline und HLL/OS Support

Key Features der TriCore-Core-Architektur (siehe ***Abb. 2***):

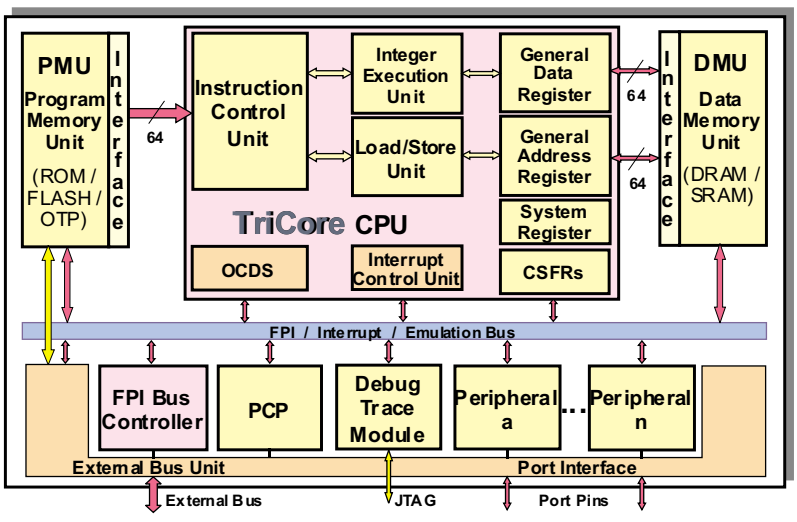


Abb. 2 TriCore Blockschaltbild

- 32-Bit RISC-Architektur (Load/Store)
- Harvard Speicherarchitektur (separate On-chip Instruktions- und Datenbusse)
- 4 Gbyte linearer Adressraum (für Programm, Daten, Peripherie)
- schnelle On-chip Speicher (SRAM, DRAM, ROM, OTP, Flash)
- 10 ns Zykluszeit (@ 100 MHz)
- Super-scalar Core (bis zu drei Befehle können parallel bearbeitet werden)
- 32-Bit und 16-Bit Befehlsformate
- kurze Interrupt-Reaktionszeit (4 Zyklen)
- Multi-Tasking Support (schneller Context Switch)
- DSP Features
- Multiply/Accumulate (1 Zyklus)
- Zero-Overhead Loops
- Saturated-Math Instruktionen
- spezielle Adressierungsmechanismen (circular und bit reverse)
- spezielle Datenformate (Q-Format)
- On-chip Debug System OCDS mit JTAG-Interface
- ein einheitlicher Toolsatz für μC und DSP

Die **32-Bit CPU** kann auf Grund der Super-scalar Architektur (drei Pipelines) einen Arithmetik-/Logik-Befehl, einen Load/Store-Befehl und einen Loop-Befehl **gleichzeitig** abarbeiten. Außerdem steht Packed-Arithmetik zur Verfügung, das heißt zwei 16-Bit Werte (Half-Word) oder vier 8-Bit Werte (Byte) können mit Hilfe **eines Befehls** in der MAC-Unit bearbeitet werden. Dadurch können DSP-Algorithmen noch schneller abgearbeitet werden.

Die **DSP**-Funktionalität des TriCore besteht aus der MAC-Unit, den speziellen Adressierungsmethoden und einem umfangreichen Satz an DSP-Befehlen. Dazu gehören Additionsbefehle für Word (32-Bit), packed Half-Word (16-Bit) und Byte (8-Bit) Datenformate, Multiplikation mit Double-Word (64-Bit) Ergebnis sowie kombinierte Multiply/Accumulate-Befehle. Fractional Arithmetik mit Q-Format Datentypen und im Befehl integrierte Shift-Funktionen um das Ergebnis einer Operation wieder in das passende Q-Format zu wandeln sind für die verschiedenen Filter-Berechnungen genauso notwendig wie die speziellen Adressierungsarten. Mit Hilfe der unterschiedlichen Adressierungsmechanismen sollen möglichst alle Operanden für eine MAC-Operation mit einem Zugriff (Load/Store) gelesen bzw. geschrieben werden können. Dies unterstützt der TriCore durch vielfältige indirekte Adressierungsarten wie Register-indirekt mit post-inkrement, Register-indirekt mit Offset oder Adressierung über ein Registerpaar für Zugriffe auf zirkulare Puffer bzw. Bit-Reverse Adressierung.

Die Load/Store-Architektur wird durch breite Adress- und Datenregister (A0 – A15, D0 – D15) unterstützt (siehe ***Abb. 3***). Da im 16-Bit oder 32-Bit Befehlsformat keine 32-Bit Adresse direkt im Befehl untergebracht werden kann, wird mit Hilfe der 16 Adressregister hauptsächlich indirekt adressiert. Es stehen aber auch Befehle zur Verfügung, die einen Adress-Offset (10-Bit bzw. 16-Bit Offset) bzw. eine direkte Adresse (18-Bit) beinhalten. Basis für eine schnelle Befehlsabarbeitung in der Pipeline sind die **getrennten Bussysteme** für Programmspeicher- und Datenspeicher-Zugriffe. Ein **64-Bit-Datenbus** ist für das Lesen der Befehle (Instruction Fetch) zuständig, **zwei 64-Bit-Datenbusse** sind für Lese- und Schreibzugriffe zum On-chip RAM verfügbar. Der **Flexible Peripheral Bus FPI** steht für alle On-chip Zugriffe (z.B. auf Peripherie-Module, JTAG/OCDS-Modul, ...) und auf die External Bus Unit EBU zur Verfügung.

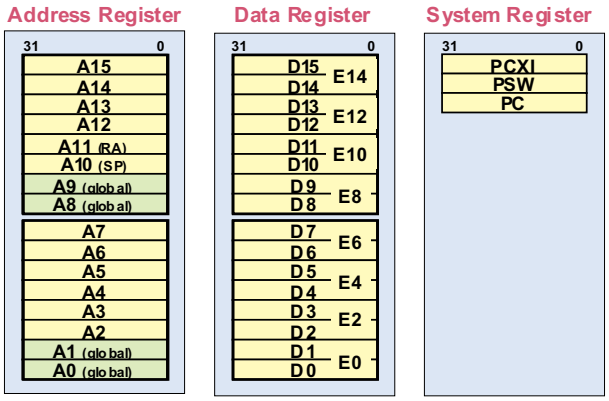


Abb. 3 TriCore Register

Ein umfangreicher Satz an Arbeitsregistern bedeutet häufig, dass bei Unterbrechung durch Interrupt bzw. bei einem Task-Switch viel gesichert werden muss – und damit natürlich viel Zeit benötigt wird. Beim TriCore kann ein Context Switch nur in das interne RAM erfolgen. Hierfür stehen die zwei 64-Bit breiten Datenbusse zur Verfügung, deshalb ist ein sehr **schneller Context Switch** möglich. Die Arbeitsregister werden dazu in zwei Context-Bereiche unterteilt:

- **Lower Context** (D0 – D7, A2 – A7, PCXI, PC) und
- **Upper Context** (D8 – D15, A10 – A15, PSW, PCXI)

Das Sichern des **Upper Context** wird bei Funktionsaufrufen (CALL- Befehl) und Unterbrechung des Programms durch Interrupt **automatisch** durchgeführt und benötigt ganze **zwei Zyklen**. Für das Umschalten des gesamten Context (Upper und Lower) bei einem Task Switch stehen verschiedene Befehle zur Verfügung, das Sichern benötigt hierbei auch nur acht Zyklen.

Um ein effektiv arbeitendes Echtzeitsystem aufbauen zu können stellt der **Interrupt-Controller** (ICU) des TriCore **255 Prioritätsebenen** zur Verfügung. Jedem Interrupt kann dabei **individuell** eine der vorhandenen Ebenen zugewiesen werden. Für die Realisierung von **Interrupt-Gruppen** wird bei einer Programmunterbrechung durch einen Interrupt eine generelle Interruptsperre herbeigeführt. Die aktuelle CPU-Priorität kann in der Interrupt Service Routine per Software verändert und die Interruptsperre anschließend aufgehoben werden (alles mit einem einzigen Befehl).

Neben der Bearbeitung eines Interrupts durch die CPU besteht auch die Möglichkeit, den Interrupt durch den **Peripheral Control Processor PCP** bearbeiten zu lassen (siehe ***Abb.2***). Der PCP ist ein leistungsfähiger DMA-Controller der über den On-chip FPI-Bus-Zugriff auf alle Speicher- und Peripherie-Module hat. Er verfügt über eigene Programm- und Datenspeicher, die nach Reset von der CPU initialisiert werden und damit auch komplexe Interrupt-Abarbeitungen durch den PCP ermöglichen. Benötigt der PCP seinerseits Dienste der CPU, kann er diese über eine Interrupt-Anforderung an den Interrupt-Controller (ICU) der CPU weitergeben.

Für die **Systemsicherheit** im TriCore sorgen Watchdog Timer und das Trap-System. Verschiedene Möglichkeiten einen Baustein-Reset auszulösen und anschließend den Reset-Grund abzufragen, komplettieren die System Überwachung. Der Watchdog Timer löst nicht sofort ein

Reset aus, sondern kann in einer Trap Routine zunächst dafür sorgen, dass ein definierter System-Zustand hergestellt und anschließend mit einem Software-Reset das System definiert rückgesetzt wird. Nach einem Reset kann in einem Status Register abgefragt werden, wodurch der Reset Vorgang ausgelöst und was alles rückgesetzt wurde.

Das Trap-System umfaßt sieben **Trap-Klassen** – von Reset (höchste Priorität) über internal Protection, Instruction Error, Context Management, internal Bus/Peripheral Errors, Assertion und System Call bis hin zum Non-maskable Interrupt NMI (niedrigste Priorität). Ein Trap unterbricht das laufende Programm in jedem Fall (kann nicht gesperrt werden) und verzweigt zum jeweiligen Trap Vektor.

Das Einsatzgebiet des TriCore wird überall dort sein, wo mehrere Aufgaben – quasi parallel – von nur einer CPU bearbeitet werden sollen, das heißt in **Multi-Tasking Systemen**. In diesen Systemen ist ein wichtiger Aspekt der Schutz von Programmen, Daten und Peripheriemodulen der unterschiedlichen Tasks gegen unberechtigte Zugriffe anderer Tasks. Dies unterstützt der TriCore durch einen integrierten **Protection Mechanismus**. Mit zwei bis vier Sätzen von Protection Registern (bestehend aus mehreren Code- und Datenregistern und einem Moderegister) kann für jede Task der jeweilige Programm- und Datenspeicherbereich eingegrenzt werden. Jeder Zugriff außerhalb dieses Bereichs führt zu einem Protection Trap. Ein I/O-Protection Mechanismus legt zusätzlich fest, ob eine Task volles Zugriffsrecht (lesen und schreiben) auf alle bzw. nur auf ungeschützte Peripherie-Module hat oder ob nur gelesen werden darf. Das heißt nur die Task mit vollem Zugriffsrecht (z.B. das Betriebssystem) kann die Betriebsart von Peripherie-Modulen einstellen beziehungsweise ändern. Um diesen Protection Mechanismus in vollem Umfang zu gewährleisten sind im 4 GByte Speicherraum bestimmte Adressbereiche für die On-chip und externe Peripherie reserviert (siehe ***Abb. 4***).

Der Baustein- bzw. Programmtest erfolgt beim TriCore mit Hilfe einer genormten Schnittstelle (JTAG, IEEE 1149) über die der externe Host (z.B. PC oder Emulator) mit der Target-Hardware verbunden wird. Das JTAG-Interface umfasst Pins für den synchronen seriellen Datenaustausch zwischen Host und Target (TDI – Test Data In, TDO – Test Data Out, TCK – Test Clock, TMS – Test Mode Select), Boundary Scan (Schieberegister zwischen den Pins des Bausteins und dem Core) sowie verschiedene Steuer- und Kontrollregister. Zusätzlich zu diesem

Abb. 5 TriCore - Third Party Partner

Standard-Interface verfügt der TriCore über ein **On-chip Debug System OCDS** (siehe ***Abb. 2***) das eine Breakpoint-Logik enthält. Mit Hilfe der OCDS-Register kann definiert werden, welches Ereignis zu einem Break-Event führen und was auf Grund dieses Events an Reaktion erfolgen soll. Um auf externe Ereignisse reagieren zu können, steht zusätzlich ein Break-Input Pin zur Verfügung. Ein Break-Output Pin kann eingesetzt werden, um bei einem Break-Event ein externes Trigger-Signal (z.B. für ein Oszilloskop) auszulösen. Das heißt, der TriCore Standard-Chip verfügt immer über Debug-Optionen, die sowohl für den Test in der Entwicklungsphase als auch für In-System Tests in der fertigen Applikation eingesetzt werden können. Über einen Enable-Eingang kann das OCDS während Reset aktiviert oder deaktiviert werden. Das JTAG-Interface kann auch verwendet werden, um den On-chip Programmspeicher (OTP oder Flash) zu programmieren.

Last but not least stellen verschiedene Tool-Hersteller eine Reihe von Tools zur Verfügung, die vom Assembler bis zum Debugger eine einheitliche Bedienoberfläche bieten und für die Programmierung des μC-Parts und der DSP-Funktionalität gleichermaßen verwendet werden können. Ohne Echtzeit-Betriebssystem (Real Time Operating System RTOS) wird der TriCore sicher nicht eingesetzt, deshalb gibt es auch hierfür verschiedene Anbieter für unterschiedliche Anforderungen von der Automotiv-Applikation bis hin zur Automatisierung (siehe ***Abb. 5***).

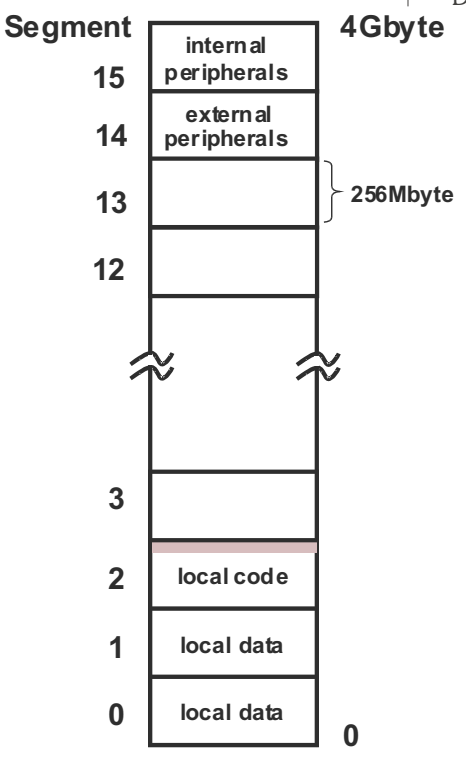


Abb. 4 TriCore Speicheraufbau

Der TriCore Core steht als VHDL-Modell zur Verfügung und kann für kundenspezifische Implementierungen durch unterschiedliche On-chip Peripherie-Module und bei Bedarf durch applikationsspezifische ASICs ergänzt werden. Die verschiedenen Module werden über den FPI-Bus miteinander beziehungsweise mit dem TriCore Core verbunden. Die Art des On-chip Speichers (ROM, OTP, Flash, DRAM, SRAM), der Speicherausbau, Einsatz von Cache Speicher und Scratchpad RAM sowie die Art der On-chip Peripherie-Module ist implementierungsabhängig. Die in diesem Artikel beschriebenen Eigenschaften beziehen sich deshalb nur auf die Core Funktionalität.

Sind Sie neugierig geworden auf mehr Informationen zu dieser neuen 32-Bit-Architektur? Dann besuchen Sie uns im Internet und informieren sich über unser Kursangebot zum TriCore: http://www.microconsult.de/

Mikrocontrollerspezifische C-Erweiterungen

Wilhelm Brezovits

ANSI-C-Compiler für Mikrocontroller verfügen über Erweiterungen, um einen effizienten und vollständigen Zugriff auf die Architektur des Bausteins zu ermöglichen.

Diese Erweiterungen sind natürlich nicht (wie ANSI-C) genormt.

Das bedeutet, dass die Implementierung dieser Erweiterungen - also die entsprechende Syntax, von Compiler zu Compiler abweichen darf – und dies auch tut.

Erweiterungen

Die Erweiterungen lassen sich generell in drei Gruppen zusammenfassen:

1. zusätzliche Datentypen
2. Steuerwörter
3. Intrinsic/Builtin - Funktionen

1. zusätzliche Datentypen

- **Bits**

Im Adressraum der Maschine befinden sich bitadressierbare Bereiche.

Der Befehlsvorrat vom Mikroprozessor im Mikrocontroller unterstützt mit einer eigenen Bitverarbeitungshardware die Bitmanipulation der Bits in den bitadressierbaren Bereichen (boolsche Operationen, Bit-Setzen/Löschen, Sprünge wenn Bit gesetzt/nicht gesetzt, ...).

Um diese Bitbefehle zu nutzen, verfügen Mikrocontroller Compiler über den Datentyp `bit`

`bit`/`bit` (Implementierung: KEIL/TASKING-Syntax).

- **SFR** (Special Function Register)

Die Architektur (Mikroprozessor, Betriebssystem on Silicon – Interruptsystem und die On-Chip-Peripherals) des Mikrocontrollers wird durch sogenannte SFR dargestellt.

Hinter SFR verstecken sich also die Register der CPU, des Interruptsystems aber auch der

On-Chip-Peripherals (wie z.B. Port-Zustände, Timer-Control-Register, AD-Wandler Konfigurations- oder Ergebnisregister, u.s.w.).

Aus Programmierersicht können SFR als „Schnittstelle“ zur „Hardware“ des Mikrocontrollers betrachtet werden.

Der Befehlsvorrat des Mikrocontrollers unterstützt den Zugriff auf die SFR. So kann z. B. mit einem Befehl ein SFR gelesen, manipuliert und zurückgeschrieben werden.

Aus Sicht von ANSI-C handelt es sich bei den SFR im Prinzip um den Datentyp „`unsigned int volatile`“, mit der Eigenschaft, eine „fixe Adresse“ im Adressraum zu haben (SFR-Bereich).

Zur Unterstützung der Architektur des Bausteins verfügen Mikrocontroller Compiler über den Datentyp `sfr` und `sbit` / `sfr`, `esfr` und `sfrbit`, `esfrbit` (Implementierung: KEIL/TASKING-Syntax).

2. Steuerwörter

Um für eine Variable eine bestimmte Adresse (einen bestimmten Platz im Zielspeicher) im internen oder externen Speicher (im Adressraum) festzulegen, ein bestimmtes Adressierungsschema - abweichend vom Speichermodell in dem übersetzt wird auszuwählen oder eine bestimmte Größe von Feldern definieren zu dürfen - gibt es Steuerwörter.

z.B.: `idata`/`iram`, `bdata`/`bitword`, `sdata`/`system`, `near`/`near`, `far`/`far`, `huge`/`shuge`, `xhuge`/`huge`, (Implementierung: KEIL/TASKING-Syntax)

Auch gibt es Steuerwörter für die Definition einer Interrupt Service Routine oder für Registerbankunterstützung (`interrupt`, `using`).

3. Intrinsic/Builtin - Funktionen

Der Mikrocontroller verfügt über Befehle wie z.B. `IDLE` (Enter_Idle_Mode) oder `SRVWDT` (Service_Watchdog_Timer).

Die Hochsprache (ANSI-C) gestattet es nicht, solche Befehle abzusetzen, da dafür keine Syntax vorgesehen ist – die Programmiersprache C ist ja hardwareunabhängig und kennt solche Befehle nicht.

Damit der Programmierer Zugriff auf solche bausteinspezifische Befehle hat, gibt es die intrinsic-/builtin - Funktionen:

einige Beispiele:

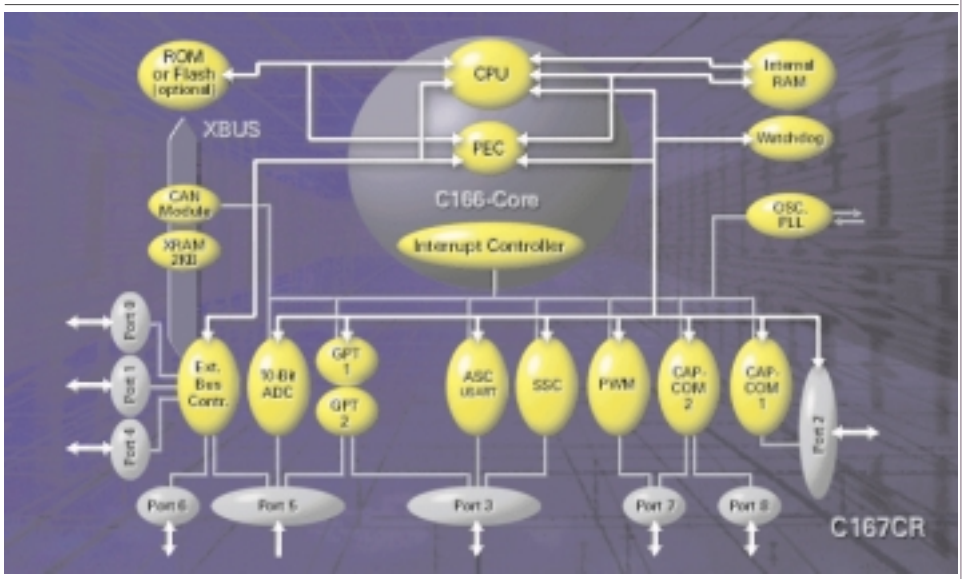
`_idle_()`/`_idle()`, `_srvwdt_()`/`_srvwdt()`, `_trap_(0)`/`_int166(0)`, `_nop_()`/`_nop()`, `_pwrdn_()`/`_pwrdn()`, (Implementierung: KEIL/TASKING-Syntax)

Schlussbemerkung

Solange wir die mikrocontrollerspezifischen Erweiterungen des C Compilers nicht nutzen, programmieren wir portabel, haben aber auch keinen Zugriff auf die Architektur.

Wir werden also bewusst alle Erweiterungen nutzen, um effizienten und vollständigen Zugriff auf die Architektur des Mikrocontrollers zu haben, dadurch sind unsere Programme aber nicht mehr portabel. Genau das wollen wir aber, wir programmieren für Embedded Systems (anwendungsspezifische Software in anwendungsspezifischer Hardware).

Architektur eines Mikrocontrollers



Wilhelm Brezovits wilhelm.brezovits@siemens.at

REKIRSCH - 2

OOP – Objektorientierte Programmierung

Überblick über C++

Wilhelm Brezovits

1. Allgemeines

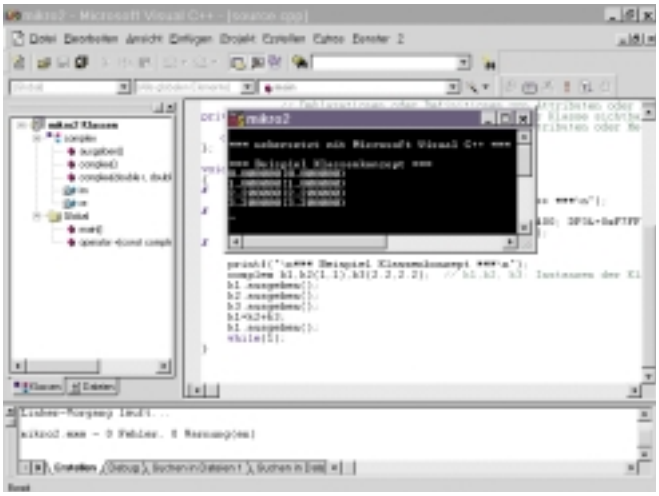
War vor Jahren bei der Programmierung von Mikrocontrollern noch die übliche Diskussion Assembler oder C, so ist heute das Diskussionsthema C, C++ oder EC++.

Embedded C++ (EC++) ist eine „Untermenge“ von C++. Durch „Weglassen“ von Teilen des C++ Sprachumfanges möchte man bei EC++ Laufzeit gewinnen. Eine Spezifikation von EC++ findet man unter *http://www.caravan.net/ec2plus/*.

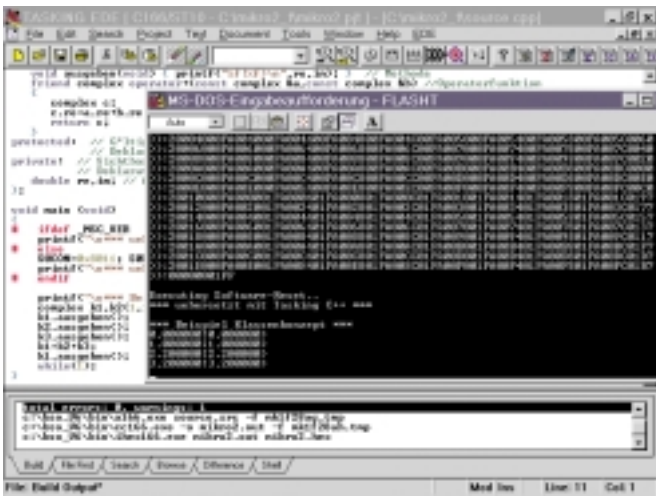
Die folgenden Programmbeispiele (gleicher Source Code!) wurden mit dem Microsoft Visual C++ Compiler und dem Tasking C++ Compiler übersetzt und erfolgreich auf verschiedenen Zielsystemen zum Ablauf gebracht.

C++ Compiler	Zielsystem
Microsoft Visual C++ (Version 6.0 Service Pack 2)	Intel-Mikroprozessor im PC
Tasking C++ (Version 6.0r3)	16-Bit-Mikrocontroller

Screenshot Microsoft Entwicklungsumgebung und „Ergebnis“ Beispielprogramm



Screenshot Tasking Entwicklungsumgebung und „Ergebnis“ Beispielprogramm



Im Gegensatz zum zentralen Aufbau eines Programms (ausgehend von `main()`) bietet die OOP die Möglichkeit, einzelne eigenständige Intelligenzinseln (Klassen) zu definieren.

Der größte Vorteil der OO-Programmierung ist die Wiederverwendbarkeit, d.h. Klassen müssen so entwickelt werden, dass sie alle Verantwortlichkeiten über ihre Daten und ihr Verhalten übernehmen und dadurch auch unter völlig verschiedenen Umgebungen in unterschiedlichen Situationen eingesetzt werden können.

Man unterscheidet zwischen zwei Arten der Wiederverwendung von Klassen: Benutzungsbeziehung („hat ein“) und Vererbungsbeziehung („ist ein“).

2. OOP im Detail

Die drei großen Säulen der objektorientierten Programmierung:

- Klassenkonzept (Kapselung durch Klassen)
- Vererbung (hierarchische Wiederverwendung in der Vererbung)
- Polymorphie (Vielgestaltigkeit)

2.1 Klassenkonzept

Im Prinzip ist eine Klasse (`class`) ein Datentyp, also eine komplexe Datenstruktur so wie in C ein Aufzähltyp (`enum`), eine Struktur (`struct`), ein Bitfeld oder eine Variante (`union`).

Klassen (Datenkapsel, Struktur) können als eine Erweiterung von Strukturen betrachtet werden und beinhalten Daten (Attribute, Eigenschaften, data member) und Funktionen (Methoden, Elementfunktionen, Member-Functions, Botschaften) in einer abgeschlossene Einheit.

Die Klasse muss die volle Verantwortung für ihre Daten und ihr Verhalten übernehmen!

Interne Daten und Funktionen werden innerhalb der Klasse versteckt (Zugriffsattribut, Gültigkeitsbereich: `private` oder `protected`).

Dem Anwender der Klasse wird eine Schnittstelle (`public`) zur Benutzung zur Verfügung gestellt.

Von einer Klasse können beliebig viele Variablen (Instanzen, Instanzvariablen, Objekte) erzeugt werden.

Beispiel

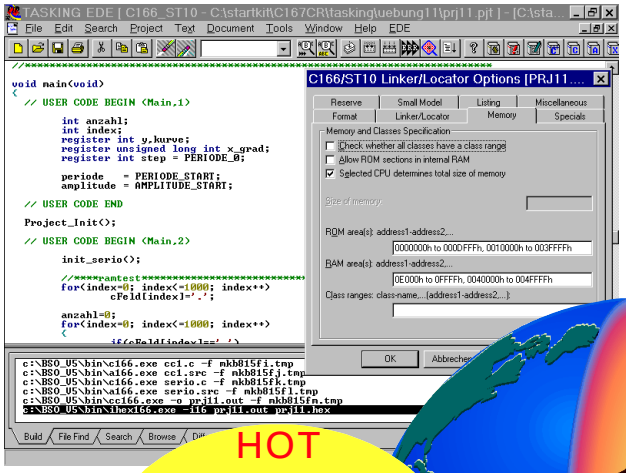
```
#include <stdio.h>
#ifdef _C166  // Compiler=Tasking C++
#include <reg167.h>
#endif

class complex  // complex = Name der Klasse
{
public: // Sichtbarkeitsbereich:
        // Schnittstelle der Klasse nach außen sichtbar
        // Deklarationen oder Definitionen
        // von Attributen oder Methoden:
   complex() : re(0),im(0) {}  // Konstruktor
   complex(double r,double i) : re(r),im(i) {} // Konstruktor
   void ausgeben(void) { printf("%f|%f)\n",re,im); }  // Methode
   friend complex operator+(const complex &a,const complex &b)
      //Operatorfunktion
   {
      complex c;
      c.re=a.re+b.re; c.im=a.im+b.im;
```

MTM SYSTEME
Messen Testen Medien

Ing. Gerhard Muttenthaler
Hirschstettnerstraße 21
A-1220 Wien

(+43 1 2032814
7 +43 1 2021303
e-✉ g.muttenthaler@mtm.at

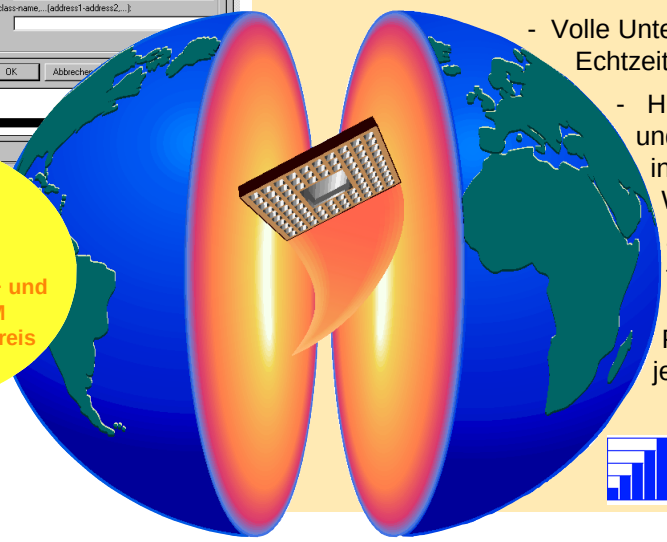
. **www.mtm.at**



C / C++ / EC++ - Compiler und Debugger

Der Standard für Entwicklungstools der C16x Familie

- Hochoptimierender ANSI C, C++ und EC++ Compiler mit allen Erweiterungen um effizient auf die Architektur der Infineon 16 und 32 Bit Mikrocontroller zugreifen zu können.
- EDE: integrierte Entwicklungsoberfläche unter Win98/NT.
- CAN-Bibliothek.
- Volle Unterstützung von unterschiedlichen Echtzeitbetriebssystemen.
- High-speed Simulator-Debugger und ROM-Monitor zum Debuggen in C++, C und Assembler unter Win95/NT.
- "plug-and-play" ROM-Monitor für Evaluationboards. Debugging via CAN. CrossView Pro Simulator Debugger ist in jedem Packet beinhaltet!



HOT SUMMER ACTION

C-Compiler / C++ / EC++ und Vollversion des ROM Debuggers XVW zum Preis des C-Compiler Pakets

TASKING

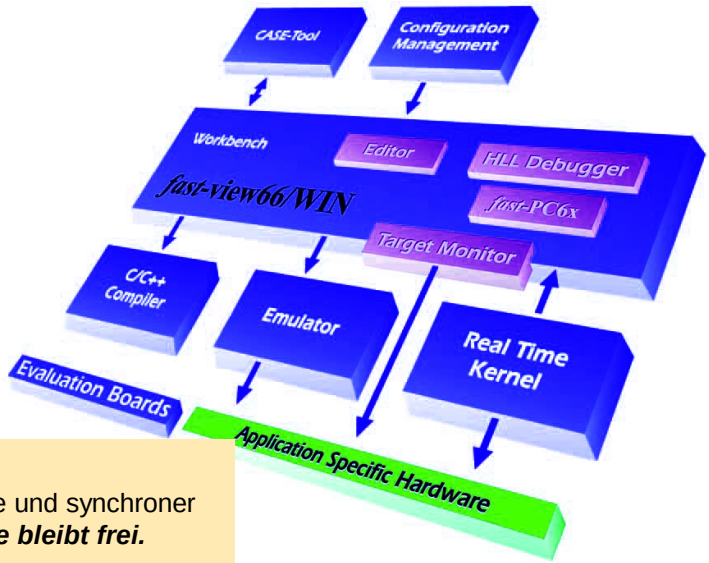
Infineon technologies

TriCore Unified Processor

Die einzig wahre Debug-Lösung für Ihre Infineon 16/32 Bit Mikrocontrollerentwicklung.

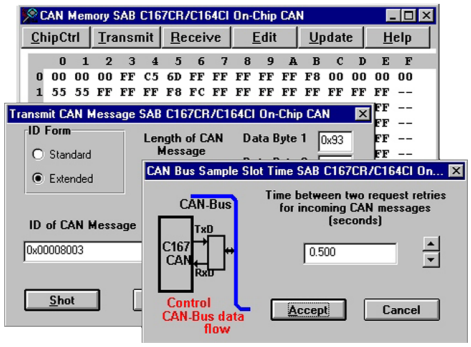
fast-view66/WIN

Oder wissen Sie einen schnelleren Weg zur Fertigstellung Ihres Projekts?



- Unterstützung aller C16X Derivate und TriCore.
- HighSpeed-Datenübertragung über Kommunikationskarte und synchroner Schnittstelle. ***Achtung: Asynchrone Schnittstelle bleibt frei.***
- Kommunikation auch über RS232/RS485, CAN-Bus, JTAG/OCDS.
- CAN-Bus Monitoring.
- In-System-FLASH-Programmierung.
- C++/EC++ und Echtzeitbetriebssystem Unterstützung.
- Syntax-Coloring und Quelltext Browser.
- Leistungsstarker Editor und Projektmanager - Codewright.
- ***NEU:*** Stecken, starten, fertig - Vielfalt im Zielsystemzugang mit fast-AccessDevice, ein Add-On zu ***fast*-view66/WIN**.

pls Programmierbare Logik & Systeme



```
        return c;
     }
protected:// Gültigkeitsbereich:
          // in eigener und abgeleiteter Klasse sichtbar
          // Deklarationen oder Definitionen
          // von Attributen oder Methoden
private:  // Sichtbarkeitsbereich: nur innerhalb der Klasse sichtbar
          // Deklarationen oder Definitionen
          // von Attributen oder Methoden:
   double re,im; // Attribute
};

void main (void)
{
#ifdef  MSC VER     // Compiler=Microsoft C++
    printf("\n*** uebersetzt mit Microsoft Visual C++ ***\n");
#else               // Compiler=Tasking C++
    SOCON=0x8011; S0BG=0x0040;
    P3|=0x0400; DP3|=0x0400; DP3&=0xF7FF; S0TIR=1;
    printf("\n*** uebersetzt mit Tasking C++ ***\n");
#endif

    printf("\n*** Beispiel Klassenkonzept ***\n");
    // k1,k2, k3: Instanzen der Klasse
    complex k1,k2(1,1),k3(2.2,2.2);
    k1.ausgeben();
    k2.ausgeben();
    k3.ausgeben();
    k1=k2+k3;
    k1.ausgeben();
    while(1);
}
```

Konstruktoren und Destruktor

Zwei besondere Methoden (Konstruktor, Destruktor) möchte ich näher beschreiben.

Der Konstruktor hat den gleichen Namen wie die Klasse und wird beim Instanzieren (Erzeugen) eines Objekts der Klasse aufgerufen. Der Destruktor hat auch den gleichen Namen wie die Klasse, beginnt allerdings mit `~` und wird beim Zerstören der Instanz aufgerufen. Dies ist bei dynamisch angeforderten Speicher sehr praktisch, da der Destruktor diesen freigibt.

Operator Overloading

Definiert man z.B. eine Klasse für komplexe Zahlen und addiert man zwei Instanzen der Klasse, so muss die richtige Vorgehensweise bei der Addition definiert werden. Dazu schreibt man eine sogenannte Operator-Funktion. Der Name der Methode lautet `operator+`. Dadurch ist die Anweisung `k2+k3` in unserem Beispiel erst möglich.

2.2 Vererbung

Eine Klasse (abgeleitete Klasse) kann eine andere Klasse (Basisklasse) erben und damit existierenden Code wiederverwenden.

Natürlich kann eine Klasse auch viele andere Klassen erben (=Mehrfachvererbung).

In der abgeleiteten Klasse sind dabei alle Daten/Funktionen der Basisklasse(n) verfügbar.

Beispiel

```
#include <stdio.h>
#ifdef _C166
#include <reg167.h>
#endif

struct datum
{
  int tag,monat,jahr;
};

class mensch                    // Basisklasse
{
 protected:
   datum geburtstag;
   int groesse;
};

class mann : public mensch      // Vererbung,
                                // mann wird von mensch abgeleitet
{
 protected:
   char bart;
};

class frau : public mensch      // Vererbung,
                                // frau wird von mensch abgeleitet
{
 protected:
   int kinder;
};

class paar : public mann, public frau  // Mehrfachvererbung
{
   datum ehe;
};

void main (void)
{
#ifdef _MSC_VER
    printf("\n*** uebersetzt mit Microsoft Visual C++ ***\n");
#else
    SOCON=0x8011; S0BG=0x0040;
    P3|=0x0400; DP3|=0x0400; DP3&=0xF7FF; S0TIR=1;
    printf("\n*** uebersetzt mit Tasking C++ ***\n");
#endif

    printf("\n*** Beispiel Vererbung ***\n");
    paar paar1, paar2, paar3;  // 3 Instanzen der Klasse paar
                               // werden erzeugt
    while(1);
}
```

2.3 Polymorphie

Polymorphie beschreibt die Fähigkeit, an unterschiedliche Objekte (entstanden durch Vererbung, also Ableitung) gleichnamige Botschaften zu senden, die abhängig vom Objekt zu unterschiedlichen Reaktionen führen.

Das bedeutet, bei einem Methodenaufruf wird anhand des Typs des gebundenen Objektes vom Compiler realisiert durch eine vom Compiler selbst generierte VMT-Tabelle (virtuelle Methoden Tabelle) herausgefunden, welche Methode er rufen soll.

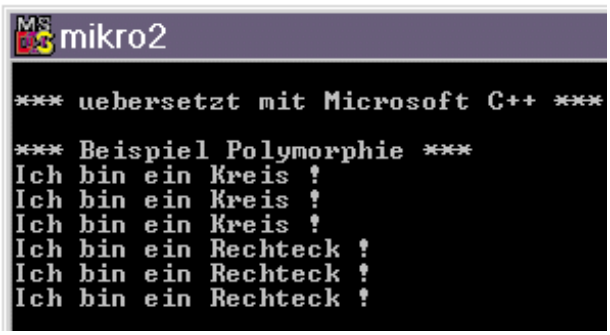
Beispiel

```
#include <stdio.h>
#ifdef _C166
#include <reg167.h>
#endif

class Zeichnung                          // abstrakte Basisklasse
{
public:
   virtual void drucke() = 0; // pure virtuelle Funktion
   // aufgrund "= 0" kann keine Instanz von Zeichnung angelegt werden
   // aufgrund des Steuerwortes virtual generiert der Compiler
   // eine VMT-Tabelle
};

class Kreis : public Zeichnung           // abgeleitete Klasse
{
public:
   void drucke()
   {
      printf("Ich bin ein Kreis !\n");
   }
};

class Rechteck : public Zeichnung        // abgeleitete Klasse
{
public:
   void drucke()
   {
      printf("Ich bin ein Rechteck !\n");
   }
};

void main (void)
{
#ifdef  MSC VER
    printf("\n*** uebersetzt mit Microsoft Visual C++ ***\n");
#else
    SOCON=0x8011; S0BG=0x0040;
    P3|=0x0400; DP3|=0x0400; DP3&=0xF7FF;  S0TIR=1;
    printf("\n*** uebersetzt mit Tasking C++ ***\n");
#endif
```

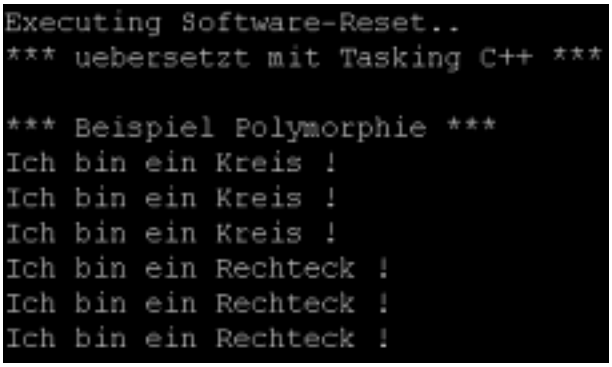
```
    printf("\n*** Beispiel Polymorphie ***\n");

    Kreis kreis1;           // 1 Objekt der Klasse Kreis anlegen
    Rechteck rechteck1;     // 1 Objekt der Klasse Rechteck anlegen

    Zeichnung *p;           // Basisklassenzeiger anlegen

    kreis1.drucke();        // "statische Variante" der Polymorphie
    p=&kreis1;
    p->drucke();            // dynamische Variante der Polymorphie
    (*p).drucke();          // andere Schreibweise

    rechteck1.drucke();     // "statische Variante" der Polymorphie
    p=&rechteck1;
    p->drucke();            // dynamische Variante der Polymorphie
    (*p).drucke();          // andere Schreibweise

    while(1);
}
```

Ergebnis

Screenshot „Ergebnis“ Microsoft C++ Compiler



Screenshot „Ergebnis“ Tasking C++ Compiler



3. Templates (Codeschablonen)

Natürlich beherrschen Mikrocontroller-C++-Compiler auch Templates:

Bei den Grundalgorithmen der Informatik (Listen, Bäume, u.s.w.) ist der Algorithmus immer der gleiche, nur der Datentyp der verwalteten Elemente ist immer ein anderer.

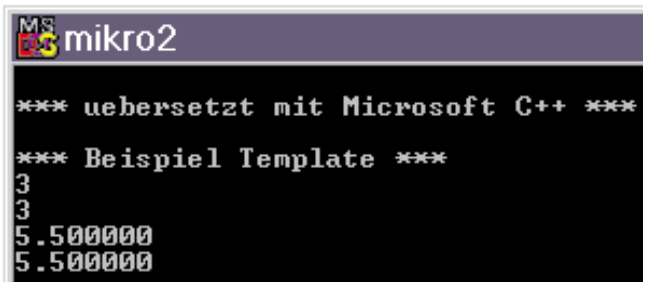
Templates bieten die Möglichkeit, einen oder mehrere Platzhalter für Datentypen bei der Definition von Funktionen oder Klassen anzugeben und so einen allgemeingültigen, typunabhängigen aber typsicheren Algorithmus zu formulieren, wobei der Compiler die entsprechende Funktion/Klasse erst bei der Verwendung für die Platzhalter generiert.

Beispiel

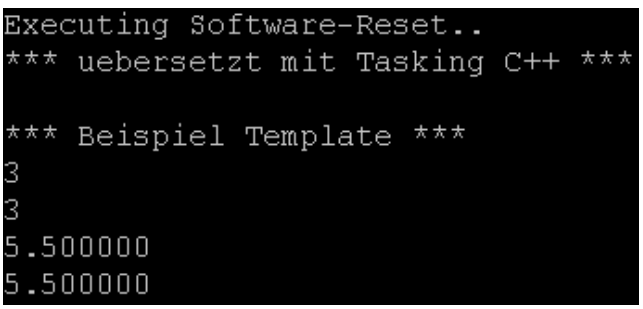
```
#include <stdio.h>
#ifndef  MSC VER
#include <reg167.h>
#endif

// Achtung:
// Typen wie int oder double koennen als bereits definierte Klassen
// betrachtet werden, der Operator > ist hier also bereits definiert.
// Bei selbstdefinierten Typen (Klassen) müssen alle vorkommenden
// Operationen (Operatoren) selbst definiert werden
// (Operator-Overloading).

template <class T1>    // T1... Platzhalter für Template
T1 max(T1 a, T1 b)
{
    return a>b?a:b;
}

void main(void)
{
#ifdef  MSC VER
    printf("\n*** uebersetzt mit Microsoft Visual C++ ***\n");
#else
    S0CON=0x8011; S0BG=0x0040;
    P3|=0x0400; DP3|=0x0400; DP3&=0xF7FF; S0TIR=1;
    printf("\n*** uebersetzt mit Tasking C++ ***\n");
#endif

    printf("\n*** Beispiel Template ***\n");

    int a=2,b=3;
    double x=5.5,y=4.4;

    printf("%d\n",max(a,b));    // entspricht: max(int,int)
    printf("%d\n",max(b,a));    // entspricht: max(int,int)

    printf("%f\n",max(x,y));    // entspricht: max(double,double)
    printf("%f\n",max(y,x));    // entspricht: max(double,double)

    while(1);
}
```

Ergebnis

Screenshot „Ergebnis“ Microsoft C++ Compiler



Screenshot „Ergebnis“ Tasking C++ Compiler



Schlussbemerkung

Für weitere Informationen möchte ich die WWW-Seite eines C, C++ und EC++ Compiler-Herstellers für die Infineon 16-Bit- und 32-Bit-Mikrocontrollerfamilien anführen:

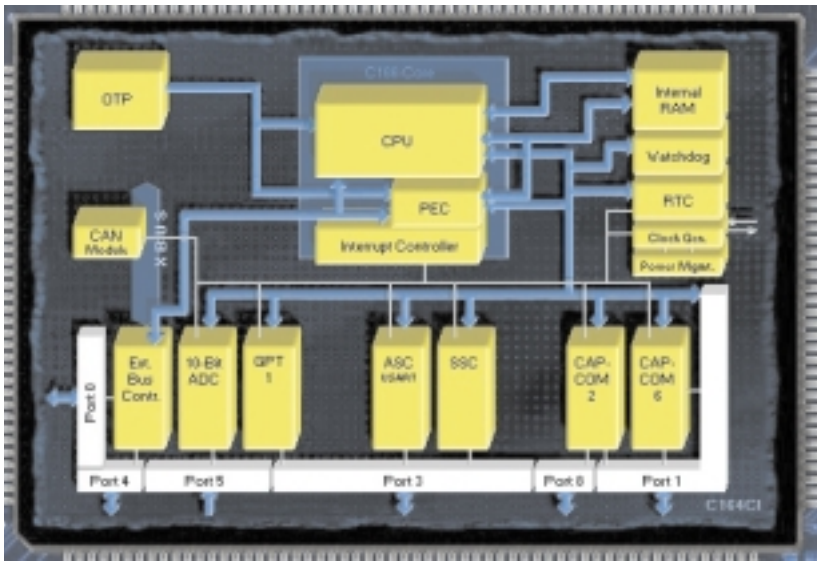
http://www.tasking.com/products/80C166/

Ich hoffe, der Artikel konnte dem Leser einen Überblick über C++ bieten.

Vielleicht animiert er sogar, Software für die Infineon 16-Bit- oder 32-Bit-Mikrocontroller ab sofort in C++ zu implementieren?

C16x Architektur

Christian Perschl



Wir wollen uns im folgenden einen Überblick über die Architektur der C16x-Familie am Beispiel des Bausteins C164CI verschaffen. Eine umfassende Betrachtung der Bausteinarchitektur würde den Rahmen dieses Artikels sprengen, daher beschränken wir uns auf die wesentlichen Merkmale. Der Baustein C164CI zeichnet sich durch zahlreiche **on-Chip Features** (z.B. 64k OTP-Programmspeicher, CAN-Interface V2.0B aktiv, PWM, RTC, flexibles Power Management etc.) aus, wodurch er sich besonders für Applikationen in den Bereichen Automobil- und Industrie-Elektronik eignet. Es sollte hier noch erwähnt werden, dass alle C16x Varianten vom Standard C167 Baustein abstammen.

Die C16x-Architektur vereinigt Vorteile aus der Welt der RISC- und CISC-Prozessoren. Die mächtige CPU (bis zu 10 Millionen Befehle pro Sekunde) ist auf effiziente Weise mit den zahlreichen on-Chip Peripheriemodulen verbunden. Einer der vier Busse, der XBUS, ist eine interne Darstellung des externen Businterfaces. Dieser Bus sieht eine standardisierte Methode zur Anbindung von integrierten applikationsspezifischen Peripherieeinheiten zur Schaffung von C167-Varianten vor.

Die **High-Performance 16-Bit CPU** wird zur Beschleunigung der Befehlsabarbeitung durch eine 4-stage Pipeline unterstützt. Die Mindestbefehlsausführungszeit beträgt bei 20 MHz CPU-Clock lediglich 100ns. Weitere Hauptbestandteile der CPU sind eine 16-Bit ALU (Arithmetisch-logische Einheit) sowie dedizierte Special Function Register (SFR). Zusätzlich sind noch eine separate Multiplizier- und Dividiereinheit, ein Bitmasken-Generator sowie ein Barrel Shifter zu nennen. Für 16x16 Bit Multiplikationen sind 500ns, für 32/16 Bit Divisionen 1μs Ausführungszeit anzusetzen.

Der **Befehlssatz** der C16x-Familie wurde in Hinblick auf Hochsprachen und Betriebssysteme optimiert. Sprünge, Unterprogrammaufrufe und Schleifen benötigen dank ausgeklügelter Mechanismen ein Minimum an Ausführungszeit (Stichwort "Jump Cache"). Programmiertechnisch häufig benutzte CASE-Anweisungen werden somit von C16x von Assembler- bis auf Hochsprachen-Ebene optimal unterstützt.

Weiterhin werden zahlreiche Möglichkeiten zu Bool'scher Bit-Manipulation zur Verfügung gestellt.

Mit dem C164CI sind bis zu 4 MByte Programm- und Datenbereich linear adressierbar (C167: 16MByte). An **on-Chip Speicher** sind 2 KByte RAM sowie 64 KByte OTP (One Time Programmable) vorhanden. Der Takt wird entweder von einer integrierten PLL erzeugt oder er wird direkt von außen (vorgeteilt) eingespeist. Der externe Datenbus kann wahlweise auf 8-Bit oder auf 16-Bit Breite konfiguriert werden. Zur Einsparung von Pins wird der externe Adress-/Datenbus beim C164CI im Multiplex-Modus betrieben.

Vier programmierbare **Chip-Select**-Signale stehen zur Anbindung externer Speicher- und/ oder Peripheriebausteine zur Verfügung. Die Charakteristika des externen Busses (z.B. Adressierungsbereiche, Wait States etc.) sind individuell einstellbar.

Das Register-basierende Design unterstützt eine flexible **Registerbankstruktur**. Mit Hilfe eines variablen Register-Bank-Managements kann man unter anderem Interrupt-Service-Routinen sehr effizient gestalten, da sich einzelne Tasks klar voneinander trennen lassen. Gegenüber den (nicht mehr zeitgemäßen) reinen Akkumulator-Maschinen erspart man sich zudem die zeitraubenden Befehle PUSH und POP. Programmcode gestaltet sich dadurch kompakter und übersichtlicher. Zwischen einzelnen Registerbänken kann in nur einem Zyklus umgeschaltet werden (Context Switching).

Das **Interrupt-System** sieht 16 Prioritäts-Level bei 32 Interrupt-Quellen und einer minimalen Sample Rate von 50ns vor. Dank 8-Kanal PEC (Peripheral Event Controller) kann unter Umgehung der Pipeline unmittelbar auf einen Peripherie-Interrupt reagiert und Daten-Transfer in nur einem Zyklus durchgeführt werden.

Die **on-Chip Peripherie** wird über Special Function Register (SFR) konfiguriert. Beim C164CI stehen dazu 1024 Bytes an SFR Bereich zur Verfügung. Als Peripheriebausteine sind integriert ein 8-Kanal Analog-/Digitalwandler mit 9,7 μs Wandlungszeit, eine 8-Kanal Capture/Compare-Einheit sowie eine 3/6-Kanal 16-Bit Capture/Compare Einheit, die speziell für AC/DC Motorsteuerungs-Applikationen ausgelegt ist. Weiterhin zu nennen sind eine multi-funktionale Timer-Einheit, bestehend aus drei 16-Bit-Timern, ein CAN Interface (V2.0B aktiv), zwei serielle Schnittstellen mit den Betriebsmodi synchron, asynchron sowie high-speed synchron, eine Real Time Clock (RTC), ein programmierbarer Watchdog Timer sowie ein Oszillator Watchdog. Wie bei anderen Mikrocontrollern hat der Großteil der Pins eine Doppelfunktion. Betreibt man den Baustein mit externem Speicher, stehen weniger I/O-Pins zur Verfügung, da ein Teil der Ports für Adress- und Datenbus benötigt wird. Der C164CI bietet maximal 59 I/O-Anschlüsse.

Besonders für Batterieapplikationen interessant sind verschiedene **Stromspar-Modi**. Über externe oder interne Interrupts kann der Baustein aus einem solchen "Power Saving Mode" wieder in den normalen Betriebsmodus gebracht werden (wake-up).

Sehr wichtig für MiniMon (siehe ***Seite 47***) ist der on-Chip **Bootstrap Loader**, der den Aufbau einer seriellen Kommunikation zwischen Host-PC und Mikrocontroller ermöglicht (s.u.).

Den Baustein C164CI gibt es in einem 80-Pin MQFP-Gehäuse mit 0.65 mm Pinabstand.

Der Vollständigkeit halber sollte noch angefügt werden, dass Infineon Mikrocontroller durch eine Vielzahl von Entwicklungs-Werkzeugen unterstützt werden.

SIEMENS

C500 Family

Infineon technologies

Der SIEMENS Halbleiter-Bereich heißt jetzt INFINEON

Type	max. Clock Rate (MHz)	ROM (Byte)	ROM Protection	RAM (Byte)	I/O Lines (Inputs only)	ADC-Inputs/ Resolution	Timer/Counter (16-Bit)	Interrupt vectors/Levels	Serial I/O	FCAN 2.0B active	USB 1.0 Interface	PWM	Mul./Div. Unit	Data Pointers (16-Bit)	Hardware Power Down	Watch-dog Timer	Osc Watch-dog	Packaging	
C501G-L/-1R C501G-E	40	–/8 k 8 k OTP	✓ ✓	256	32	–	3	6/2	USART	–	–	–	–	1	–	–	–	P-DIP-40 P-LCC-44 P-MQFP-44	Fully compatible with 80C52/C32 standard
C504-L/-2R C504-2E	40	–/16 16 k OTP	✓ ✓	512	32	8/10	4	12/2	USART	–	–	6-ch	–	1	–	✓	✓	P-MQFP-44	CCU for DC motor control, Interrupt wakeable Power down
C513-1R C513A-L/-R/-2R	12	8 k – 12 k/16 k	–	256 512	32	–	3	7/2	USART + SSC	–	–	–	–	1	–	–	–	P-DIP-40/P-LCC-44 P-MQFP-44	Low Power Low EMI
C513A-2E	16	16 k OTP	✓	512	32	–	3	7/2	USART + SSC	–	–	–	–	1	–	–	–	P-DIP-40 P-LCC-44 P-MQFP-44	Enhanced power saving modes; EMC optimised
C505L-4E	20[2]	32 k OTP	–	512	46	8/10	3	12/4	USART	–	–	4-ch	–	8	–	✓	✓	P-MQFP-80	LCD Driver on-chip 128 segments; RTC with 32 kHz subclock
C505-L/-2R	20[2]	–/16 k	✓	512	34	8/8	3	12/4	USART	–	–	4-ch	–	8	–	✓	✓	P-MQFP-44	Enhanced power saving modes; Low EMI, Interrupt wakeable Power down
C505C-L/-2R	20[2]	–/16 k	✓	512	34	8/8	3	12/4	USART	✓	–	4-ch	–	8	–	✓	✓	P-MQFP-44	Enhanced power saving modes; Low EMI, Interrupt wakeable Power down
C505A-4E	20[2]	32 k OTP	✓	1280	34	8/10	3	12/4	USART	–	–	4-ch	–	8	–	✓	✓	P-MQFP-44	Enhanced power saving modes; Low EMI, Interrupt wakeable Power down
C505CA-4E	20[2]	32 k OTP	✓	1280	34	8/10	3	12/4	USART	✓	–	4-ch	–	8	–	✓	✓	P-MQFP-44	Enhanced power saving modes; Low EMI, Interrupt wakeable Power down
C515-L/-1R	24	–/8 k	✓	256	56 (8)	Prog. REF 8/8	3	12/4	USART	–	–	4-ch	–	1	–	✓	✓	P-MQFP-80	Compatible with SAB 80C515
C515A-L/-4R	24	–/32 k	✓	1280	56 (8)	8/10	3	12/4	USART	–	–	4-ch	–	1	✓	✓	✓	P-MQFP-80	Compatible with SAB 80C515A
C515C-L/-8R C515C-8E	10[2]	–/64 k 64 k OTP	✓	2304	57 (8)	8/10	3	15/4	USART + SSC	✓	–	4-ch	–	8	✓	✓	✓	P-MQFP-80	Low Power, Low EMI
C517A-L/4R	24	–/32 k	✓	2304	68 (12)	12/10	4	17/4	USART + UART	–	–	21-ch	✓	8	✓	✓	✓	P-MQFP-100	Compatible with SAB 80C517A
C509-L	16[2]	–	–	3328	64 (15)	15/10	5	19/4	USART + UART	–	–	29-ch	✓	8	✓	✓	✓	P-MQFP-100	CMOS/TTL Ports Bootstrap
C541U-1E[1]	12[2]	8 k OTP	✓	256	32/30	–	2	7/2	SSC	–	HUBless	–	–	1	–	✓	✓	P-SDIP-52/ P-LCC-44	USB-Transceiver-on-chip;
SAB 80C515 SAB 80C535	20	8 k –	✓ –	256	56 (8)	Prog. REF 8/8	3	12/4	USART	–	–	4-ch	–	1	–	✓	–	P-LCC-68	
SAB 80C515A SAB 83C515A-5	18	– 32 k	– ✓	1280	56 (8)	8/10	3	12/4	USART	–	–	4-ch	–	1	✓	✓	✓	P-LCC-68	
SAB 80C517 SAB 80C537	16	8 k –	✓ –	256	68 (12)	Prog. REF 12/8	4	14/4	USART + UART	–	–	21-ch	✓	8	–	✓	✓	P-LCC-84	
SAB 80C517A SAB 83C517A-5	18	– 32 k	– ✓	2304	68 (12)	12/10	4	17/4	USART + UART	–	–	21-ch	✓	8	✓	✓	✓	P-LCC-84	

[1] under development [2] CPU-Clock

Published by Semiconductor Group

Siemens Aktiengesellschaft

Ordering No. B 158-H6957-G5-X-7600
Printed in Germany
PS 11985. (8541)