



ARM Cortex-M3 Mikrocontroller

Einstieg und Praxis

Hinweis des Verlages zum Urheberrecht und Digitalen Rechtemanagement (DRM)

Der Verlag räumt Ihnen mit dem Kauf des ebooks das Recht ein, die Inhalte im Rahmen des geltenden Urheberrechts zu nutzen. Dieses Werk, einschließlich aller seiner Teile, ist urheberrechtlich geschützt. Jede Verwertung außerhalb der engen Grenzen des Urheberrechtsgesetzes ist ohne Zustimmung des Verlages unzulässig und strafbar. Dies gilt insbesondere fürervielfältigungen, Übersetzungen, Mikroverfilmungen und Einspeicherung und Verarbeitung in elektronischen Systemen.

Der Verlag schützt seine ebooks vor Missbrauch des Urheberrechts durch ein digitales Rechtemanagement. Bei Kauf im Webshop des Verlages werden die ebooks mit einem nicht sichtbaren digitalen Wasserzeichen individuell pro Nutzer signiert.

Bei Kauf in anderen ebook-Webshops erfolgt die Signatur durch die Shopbetreiber. Angaben zu diesem DRM finden Sie auf den Seiten der jeweiligen Anbieter.

Ralf Jesse

ARM Cortex-M3 Mikrocontroller

Einstieg und Praxis



mitp

Bibliografische Information der Deutschen Nationalbibliothek

Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über <http://dnb.d-nb.de> abrufbar.

ISBN 978-3-8266-9600-8

1. Auflage 2014

www.mitp.de

E-Mail: kundenservice@hjr-verlag.de

Telefon: +49 6221 / 489 -555

Telefax: +49 6221 / 489 -410

© 2014 mitp, eine Marke der Verlagsgruppe Hüthig Jehle Rehm GmbH Heidelberg, München, Landsberg, Frechen, Hamburg

Dieses Werk, einschließlich aller seiner Teile, ist urheberrechtlich geschützt. Jede Verwertung außerhalb der engen Grenzen des Urheberrechtsgesetzes ist ohne Zustimmung des Verlages unzulässig und strafbar. Dies gilt insbesondere für Vervielfältigungen, Übersetzungen, Mikroverfilmungen und die Einspeicherung und Verarbeitung in elektronischen Systemen.

Die Wiedergabe von Gebrauchsnamen, Handelsnamen, Warenbezeichnungen usw. in diesem Werk berechtigt auch ohne besondere Kennzeichnung nicht zu der Annahme, dass solche Namen im Sinne der Warenzeichen- und Markenschutz-Gesetzgebung als frei zu betrachten wären und daher von jedermann benutzt werden dürften.

Lektorat: Sabine Schulz

Sprachkorrektorat: Petra Heubach-Erdmann

Coverbild: © Franz Pfluegl

Satz: III-satz, Husby, www.drei-satz.de

Inhaltsverzeichnis

	Einleitung.....	15
Teil I	Grundlagen	25
I	Orientierung	27
I.1	Kommerzielle IDEs.....	27
I.1.1	Keil μ Vision	28
I.1.2	IAR Workbench	28
I.1.3	Sourcery Codebench.....	28
I.1.4	Atollic TrueSTUDIO	29
I.1.5	CrossWorks for ARM.....	29
I.2	Herstellerebundene IDEs.....	30
I.2.1	ATMEL Studio 6.....	30
I.2.2	Texas Instruments StellarisWare	30
I.2.3	STMicroelectronics STVD.....	31
I.3	Freie IDEs	31
I.3.1	CooCox CoIDE	31
I.3.2	NetBeans for C Developers	31
I.3.3	Code::Blocks	32
I.3.4	emIDE.....	32
I.3.5	Eclipse für C/C++-Entwickler	32
I.4	Vorbereitende Arbeiten.....	33
I.4.1	Hardware	34
I.4.2	Software.....	37
2	ARM und CMSIS	53
2.1	Einige Hintergrundinformationen.....	53
2.1.1	Die Firma ARM Holdings PLC	53
2.1.2	Das Geschäftsmodell	53
2.2	CMSIS	54
2.2.1	Implementierungen.....	54

2.3	Erzeugung der Bibliotheken	56
2.3.1	libboard: Die Bibliothek für das Entwicklungsboard	57
2.3.2	libchip: Die Bibliothek für den Mikrocontroller-Chip	62
2.4	Weitere Software installieren	64
2.4.1	SAM-BA	64
2.4.2	SEGGER J-Link GDB Server via JTAG	67
3	Das erste Eclipse-Projekt	71
3.1	Erstellen einer Projektschablone	71
3.1.1	Workspace einrichten.	72
3.1.2	Anlegen eines neuen Projekts	73
3.1.3	Projektspezifische Einstellungen.	75
3.1.4	C/C++ Build: Settings	78
3.2	Weitere erforderliche Dateien	94
3.2.1	board_cstartup_gnu.c und syscalls.c	94
3.2.2	Linkerscriptdateien	96
3.3	Konfiguration des Debuggers	104
3.4	Fertigstellen des Templates.	108
3.4.1	Anwendung der Erweiterung.	109
4	Hello World!	111
4.1	Grundlegende Hinweise	111
4.2	Erstellen des Projekts o4_oI_Blinky	112
4.2.1	Importieren der Schablone	114
4.2.2	Der Sourcecode von Blinky	116
4.3	Blinky im Debugger ausführen	126
4.4	Debugging light	132
4.4.1	Was Sie benötigen	132
4.4.2	Konfiguration der Schnittstelle	133
4.4.3	Programm laden und ausführen.	133
4.4.4	Vorteile und Nachteile dieser Methode.	134

Teil II Einfache Grundlagen der Elektronik 137

5	Der ATMEL SAM3S4B	139
5.1	Die ATMEL-SAM3S-Familie	140
5.1.1	Übersicht	140

5.2	Das Datenblatt DOC 6500	141
5.2.1	Der Aufbau von DOC 6500	142
5.2.2	Mikrocontroller anderer Hersteller	144
5.3	Elektrische Daten des SAM3S4	145
5.3.1	Minimum- und Maximumwerte	146
5.3.2	Elektrische Versorgungsspannungen	147
5.3.3	Gleichstromwerte	149
5.4	System Controller	152
5.5	CHIP_ID	153
5.5.1	Das Projekt 05_01_CHIPID	154
5.5.2	Erläuterungen	158
5.6	Weiterführende Literatur	162
6	Elektronik	165
6.1	Digitale Ausgänge	165
6.1.1	Ports A, B und C im Reset-Zustand	166
6.2	Schalten kleiner Ströme	174
6.2.1	Current Sourcing	174
6.2.2	Current Sinking	176
6.2.3	Dimensionierung bei Current Sourcing und Current Sinking	177
6.2.4	Vor- und Nachteile beider Betriebsarten	180
6.3	Schalten größerer Ströme	181
6.3.1	Bipolare Transistoren	181
6.3.2	Feldeffekt-Transistoren (FETs)	190
6.3.3	Schalten mit Optokopplern	191
6.3.4	Schalten von Leistungstransistoren	192
6.3.5	Schalten induktiver Lasten (Relais, Elektromagnete, Motoren)	192
6.4	Digitale Eingänge	194
6.4.1	Grundlegende Betrachtungen	194
6.4.2	Einfachste Form der Beschaltung	194
6.4.3	Bessere Form der Beschaltung	195
6.4.4	Erfassen größerer Spannungen I	196
6.4.5	Erfassen größerer Spannungen II	198
6.5	Allgemeine Anmerkungen	198

7	Anwendungen	199
7.1	LC-Displays	199
	7.1.1 Die Hardware	200
	7.1.2 Projekt 07_01_LCD	202
7.2	7-Segment-Anzeigen	223
	7.2.1 Kein Datenblatt verfügbar?	225
	7.2.2 Eine Möglichkeit der Ansteuerung	226

Teil III Basiskomponenten 237

8	NVIC, PMC, Clock Generator und SUPC	239
8.1	Allgemeines zu Interrupts	239
	8.1.1 Asynchrone Ereignisse	240
	8.1.2 Der NVIC – Nested Vector Interrupt Controller	240
	8.1.3 Zuordnung der Interrupt-Quellen	244
	8.1.4 Tail Chaining	246
	8.1.5 CMSIS-Funktionen für den NVIC	247
	8.1.6 (Kein) Beispiel	249
	8.1.7 Software-Interrupts	251
	8.1.8 Tipps und Empfehlungen	252
8.2	Der Clock Generator / Taktgenerator	252
	8.2.1 Funktionen des Clock Generators	252
8.3	Der PMC – Power Management Controller	256
	8.3.1 Aufgaben des PMC	257
	8.3.2 Die Taktsignale des PMC	257
	8.3.3 Weitere Informationen zum PMC	259
	8.3.4 Ausgewählte Register des PMC	260
8.4	Der SUPC – Supply Controller	264
9	Parallel Input/Output Controller	265
9.1	Port-Register und -Betriebsarten	265
	9.1.1 PIOA, PIOB und PIOC	266
	9.1.2 Die Register von PIOA, PIOB und PIOC	269
9.2	Input-Ports in der Praxis	289
	9.2.1 Das Projekt 09_01_INPUT_SAMPLE	290
	9.2.2 Das Ergebnis	296

10	Timer und Counter, Teil 1	299
10.1	Real-time Timer RTT	300
10.1.1	Projekt 10_01_RTT	300
10.1.2	Die Register des RTT	313
10.2	RTC – Die Echtzeituhr	314
10.2.1	Das Projekt 10_02_RTC	315
10.2.2	Projekt 10_02_RTC_Advanced	320
10.2.3	Die Register der RTC	339
10.3	Der Watchdog-Timer WDT	344
10.3.1	Projekt 10_03_WDT	345
10.3.2	Register des WDT	352
10.4	Der System-Timer SysTick	353
10.4.1	Grundlegende Funktion	353
10.4.2	Anwendung von SysTick	354
10.4.3	Konfiguration des SysTick	355
10.4.4	Register des System-Timers SysTick	356
10.4.5	SysTick-Interrupt	358
10.5	Abschlussbetrachtung	358
11	Timer und Counter, Teil 2	359
11.1	Timer/Counter, Grundlagen	360
11.1.1	Einsatzgebiete von Timern und Countern	360
11.1.2	Grundlegende Betrachtungen	361
11.1.3	Triggern der Counter	361
11.2	Timer/Counter programmieren	362
11.2.1	PIO-Controller konfigurieren	362
11.2.2	PMC konfigurieren	365
11.2.3	NVIC konfigurieren	365
11.3	Die Register der Timer/Counter	366
11.3.1	TC- und TC-Channel-Register	366
11.4	Projekt 11_01_TIMER_COUNTER	375
11.4.1	global.h	375
11.4.2	tcWave.h und tcWave.c	376
11.4.3	tcCapture.h und tcCapture.c	383
11.4.4	main.c	385

Teil IV Weiterführende Komponenten	389
12	Peripheral DMA Controller (PDC) 391
12.1	Prinzipieller Aufbau 391
12.1.1	Voll-Duplex-fähige Peripherie 392
12.1.2	Halb-Duplex-fähige Peripherie 392
12.1.3	Monodirektionale Peripherie 393
12.1.4	Voll-Duplex- und Halb-Duplex-Kanäle 393
12.1.5	Monodirektionale Kanäle 393
12.2	PDC-Register 393
12.2.1	Receive Pointer Register (PERIPH_RPR) 394
12.2.2	Receive Counter Register (PERIPH_RCR) 394
12.2.3	Transmit Pointer Register (PERIPH_TPR) 394
12.2.4	Transmit Counter Register (P_TCR) 394
12.2.5	Weitere Receive- und Transmit-Register 394
12.2.6	Transfer Control Register (PERIPH_PTCR) 395
12.2.7	Transfer Status Register (PERIPH_PTSR) 395
12.3	Schlussbetrachtung 395
13	PWM – Pulsweitenmodulation 397
13.1	Was ist Pulsweitenmodulation? 397
13.2	Pulsweitenmodulation – aber wozu? 398
13.3	Der PWMC der AT91SAM3S-Familie 398
13.3.1	Abhängigkeiten des PWMC 399
13.3.2	Die CMSIS-Funktion des PWMC 400
13.3.3	Zuordnung der PWM-Anschlüsse 409
13.4	Projekt I3_01_PWM 410
13.4.1	board_olimex.h 410
13.4.2	pulsewidthmod.h 414
13.4.3	pulsewidthmod.c 414
13.4.4	terminal.c 416
13.4.5	main.c 418
13.5	Drehzahlregelung eines DC-Motors 423
13.5.1	Dimensionierung der Schaltung 423
13.5.2	Drehrichtungswechsel 424
14	Analoge und digitale Größen 425
14.1	Vereinfachte Grundlagen 425
14.2	DACC – Digital-to-Analog Converter Controller 426
14.2.1	DACC-Register 427

14.2.2	CMSIS-Funktionen zum DACC	429
14.2.3	I4_01_DACC_SIGNAL_GENERATOR_WITH_INTERRUPT	430
14.2.4	Hilfsprogramm: I4_02_TABLE_GENERATOR	434
14.3	ACC – Analog Comparator Controller	442
14.3.1	Die Register des ACC	443
14.3.2	CMSIS-Funktionen zum ACC	445
14.3.3	Projekt I4_03_ACC	446
14.4	ADC – Analog-to-Digital Converter	448
14.4.1	Eigenschaften des ADC	448
14.4.2	Die ADC-Register	449
14.4.3	CMSIS-Funktionen des ADC	457
14.4.4	Projekt I4_06_ADC_TS_UND_POTI	458
Teil V Serielle Kommunikation		475
15	Serielle Schnittstellen I	477
15.1	Hardware	478
15.1.1	RS-232 (EIA 232)	479
15.1.2	RS-485	481
15.1.3	TWI (I ² C)	482
15.1.4	Serial Peripheral Interface (SPI)	482
15.1.5	Synchronous Serial Controller (SSC)	482
15.2	Serielle Schnittstellen der AT91SAM3S-Familie	483
15.2.1	Grundlegende Begriffe	483
15.3	Universal Asynchronous Receiver Transceiver (UART)	489
15.3.1	UART-Eigenschaften beim AT91SAM3S	489
15.3.2	UARTs auf dem Olimex SAM3-P256	489
15.3.3	UART-Register	490
15.3.4	RS232_0 und Retargeting	493
15.4	Universal Synchronous Asynchronous Receiver Transceiver (USART)	496
15.4.1	USART-Eigenschaften beim AT91SAM3S	497
15.4.2	USARTs auf dem Olimex SAM3-P256	498
15.4.3	USART-Register	499
15.5	Two-wire Interface (TWI)	507
15.5.1	TWI-Eigenschaften beim AT91SAM3S	509
15.5.2	TWI auf dem Olimex SAM3-P256	509
15.5.3	TWI-Register	509

16	Serielle Schnittstellen II	515
16.1	SD Card (stark vereinfacht)	515
16.1.1	Ausführungsformen und Anschlüsse	516
16.1.2	Versorgung und Stromaufnahme	517
16.1.3	Speicherkapazitäten und Zugriffsraten	517
16.2	SD-Karten im SPI-Modus	518
16.2.1	Grundlagen zum SPI	518
16.2.2	Initialisierung des SPI	518
16.2.3	Lesen und Schreiben von Rohdaten	536
16.3	High Speed MultiMedia Card Interface (HSMCI)	538
16.3.1	Merkmale des HSMCI	538
16.3.2	Informationen zu den Protokollen	539
16.3.3	Anschluss eines SD-Kartenslots	540
16.3.4	Die HSMCI-Register	540
16.3.5	Hinweis zur Nutzung des HSMCI	543
16.4	Synchronous Serial Controller (SSC)	543
16.4.1	Merkmale des SSC	544
16.4.2	Die wichtigsten Register des SSC	545
A	Glossar	547
A.1	Architektur	547
A.2	ARM	547
A.3	ARM-Befehlssatz	547
A.4	Big.LITTLE-Konzept	548
A.5	BSS	548
A.6	CMSIS	548
A.7	Cortex	548
A.8	Debugging	549
A.9	Echtzeit-Betriebssysteme	549
A.10	Embedded Linux	549
A.11	FIFO	550
A.12	Firmware	550
A.13	Heap	550
A.14	JTAG	550
A.15	LIFO	551
A.16	OCDC	551
A.17	SAM-BA	551
A.18	Stack	551

A.19	SWD	552
A.20	TDMI	552
A.21	Text-Segment	552
A.22	Thumb-Befehlssatz	553
B	Ressourcen	555
B.1	Hardware	555
B.2	Software	556
C	Literatur	559
C.1	Literatur (Buchversion)	559
C.2	Literatur (Online-Version)	559
C.3	Weitere allgemeine Quellen	561
D	Erfahrungen	563
D.1	Wechsel der Toolchain	563
D.2	GNU Tools for ARM Embedded Processors	563
D.3	Nochmals: Verwendung der Nano-Libs	564
D.4	Updates von Eclipse und dem CDT	564
D.5	Andere Probleme mit Eclipse und dem CDT	564
D.6	Debugger	565
D.7	Versionsverwaltung	565
	Stichwortverzeichnis	566

Einleitung

Bitte lesen Sie diese Einleitung – sie ist wichtig und hilft Ihnen bei der Entscheidung, ob dieses Buch das bietet, was Sie erwarten!

Einführende Bemerkungen

Dieses Buch beschreibt den Einsatz und die Programmierung von ARM-Cortex-M3-Mikrocontrollern am Beispiel des ATMEL AT91SAM3S4B, einem Mitglied der ATMEL-SAM3S-Familie. Dies bedeutet für Sie als Leser, dass die Beispielprojekte unverändert auf AT91SAM3S4B und auf leistungsstärkeren Varianten dieses Mikrocontrollers getestet wurden und dort nutzbar sind. Cortex-M3-Mikrocontroller werden aber nicht nur von ATMEL, sondern auch von vielen anderen Herstellern hergestellt: Zu diesen Herstellern zählen beispielsweise die Firmen NXP Semiconductors (LPC13xx, LPC17xx, LPC18xx), Texas Instruments (C2000-, Tiva- oder Hercules-Familie) oder STMicroelectronics mit ihren sehr beliebten und weitverbreiteten STM32-Mikrocontrollern.

Hinweis

Die Mikrocontroller der verschiedenen Hersteller unterscheiden sich teilweise erheblich voneinander, sodass die Beispielprojekte zu diesem Buch auf den Mikrocontrollern anderer Hersteller erst nach Modifikationen funktionieren werden! Um den Portierungsaufwand für die Projekte so gering wie möglich und überschaubar zu halten, habe ich die Software für die verwendeten Peripheriegeräte in eigenständige Dateien ausgelagert.

Die Mikrocontroller unterscheiden sich unter anderem in der Ausstattung mit Peripheriekomponenten: So verfügt beispielsweise der hier eingesetzte AT91SAM3S4B über zwei USARTs, zwei UARTs und über jeweils einen DAC bzw. ADC. Andere Hersteller können hier andere Schwerpunkte gesetzt haben und beispielsweise CAN-Controller integriert haben. Dies bedeutet, dass es nicht möglich ist, in einem Buch auf alle denkbaren Varianten einzugehen. Um beim Beispiel »USART« zu bleiben: Jeder der genannten (und auch weiterer) Hersteller ist in der Wahl des integrierten USART frei. Da hier ebenfalls eine schwer überschaubare Vielfalt existiert, werden sich die USARTs der Hersteller in der Zahl und in

der Programmierung der verschiedenen Register mit großer Sicherheit voneinander unterscheiden. Eine 1:1-Umsetzung der Beispiele ist aus einem weiteren Grund nicht möglich: Verwendet ATMEL zur Programmierung von GPIO-Ports beispielsweise ein Register mit dem Namen »PIO_OER« (PIO Output Enable Register), so könnte das Register mit einer entsprechenden Funktion bei NXP »PINSEL« (Abkürzung für Pin Select Register) heißen. Auf jeden Fall wird sich aber die Adresse, unter der die betrachtete Komponente im adressierbaren Speicherraum angesprochen wird, unterscheiden. Das PIOA_OER-Register des AT91SAM3S4B wird beispielsweise unter der Adresse 0x400E0E10 angesprochen: Bei einem Mikrocontroller von NXP, STMicroelectronics oder Texas Instruments wird die vergleichbare Komponente mit Sicherheit unter einer anderen Adresse angesprochen werden.

Aufgrund der Vielfalt der Hersteller und den sehr großen jeweiligen Produktpaletten ist es gar nicht möglich, das ultimative und für alle Anwendungen im gleichen Maße nutzbare »ARM-Cortex-M3«-Buch zu schreiben! Um ihre Produkte für Anwender interessant und möglichst einfach nutzbar zu machen, stellt jeder Hersteller umfangreiche Software-Bibliotheken im Quelltext zur Verfügung, die kostenlos auch in kommerziellen Produkten genutzt werden können.

Hinweis

Damit dieses Buch aber auch für Anwender der Mikrocontroller anderer Hersteller interessant ist, habe ich großen Wert darauf gelegt, Sie in der Nutzung des Datenblatts zur AT91SAM3S-Familie zu trainieren: Die Datenblätter unterscheiden sich in ihrem Aufbau nur sehr geringfügig. Während ATMEL die gesamte Familie mitsamt einer ausführlichen Registerbeschreibung und Darlegung der physischen Daten in einem einzigen Dokument zusammenfasst, werden Sie bei STMicroelectronics im Regelfall drei Einzeldokumente finden (ein Datenblatt mit den physischen Daten, ein Referenz-Handbuch und ein Programmierhandbuch): In ihrem Informationsgehalt entsprechen diese drei Dokumente aber dem einen Datasheet von ATMEL!

Zielgruppe und Voraussetzungen

Dieses Buch wendet sich an Ingenieure, Studenten technischer Fachrichtungen und Hobby-Elektroniker, die sich erstmals mit der Programmierung von Mikrocontrollern befassen oder veraltete Kenntnisse auffrischen wollen. Dabei werden mindestens durchschnittliche Kenntnisse in der Programmierung in der Programmiersprache C vorausgesetzt: Dieses Buch ist, obwohl sämtliche Beispiele in C entwickelt wurden, kein Lehrbuch für diese Programmiersprache. Obwohl ich

mich bemüht habe, die Beispiele in »verständlichem« C zu schreiben, und daher auf die »wildesten Tricks«, die in C möglich sind, verzichtet habe, ist dieses Buch definitiv nicht zum Erlernen von C geeignet!

Ebenfalls vorausgesetzt werden Basiskenntnisse der Elektrotechnik, wobei die Anforderungen hier aber wesentlich niedriger angesetzt sind (die Kenntnis des ohmschen Gesetzes und der kirchhoffschen Regeln reicht aus). Wer als Elektronik-Bastler weiß, dass man einen heißen Lötkolben nicht an seiner Spitze anfassen sollte, hat gute Chancen, dieses Buch sinnvoll zu nutzen. Die Dimensionierung einfacher Transistorschaltungen wird in den entsprechenden Kapiteln in dem Maße kurz erläutert, wie es zum Verständnis erforderlich ist. Auf weiterführende Literatur kann natürlich zugegriffen werden, erforderlich ist dies aber nicht.

Wenn Sie die höhere Mathematik mit Differenzial- und Integralrechnung inkl. Differenzialgleichungen verstehen: Herzlichen Glückwunsch! In diesem Buch reicht es aber aus, wenn Sie die vier Grundrechenarten beherrschen. In Kapitel 14 werden Sinus- und Exponentialfunktionen genutzt. Verständnis hierfür ist hilfreich, aber keine zwingende Voraussetzung.

Sie sollten aber mittlere bis gute Kenntnisse der englischen Sprache besitzen. Sehr häufig wird auf Datenblätter zu den verschiedenen eingesetzten Komponenten verwiesen, und diese Datenblätter sind fast ausschließlich in englischer Sprache verfügbar. Das Datenblatt zum in diesem Buch beschriebenen Mikrocontroller ATMEL AT91SAM3S4B ist beispielsweise nur in englischer Sprache verfügbar. In Ausnahmefällen existieren für ausgewählte Komponenten auch japanische oder chinesische Versionen der Datenblätter, andere Sprachen werden aber nicht auffindbar sein.

Technische Voraussetzungen

Auf die technischen Voraussetzungen soll an dieser Stelle nicht detailliert eingegangen werden, da diese in Kapitel 1 ausführlich beschrieben sind. Grob zusammengefasst benötigen Sie folgende Hard- und Software:

- Zwingend erforderlich ist ein sogenanntes Evaluierungsboard mit dem ATMEL-AT91SAM3S4B-Mikrocontroller. Aus Kostengründen und aus Gründen der leichten Beschaffbarkeit habe ich mich für das Olimex-Board SAM3-P256 entschieden.
- Es ist hilfreich, wenn Ihnen zumindest ein handelsübliches Messgerät zum Messen von Spannungen, Strömen und Widerständen zur Verfügung steht. Dies ist aber nicht zwingend erforderlich.

Hinweis

Wenn Sie bereits ein anderes Evaluierungsboard besitzen oder aus persönlichen Gründen einen Mikrocontroller eines anderen Herstellers einsetzen wollen oder müssen: Kein Problem! Allerdings müssen die Beispielprojekte in einem solchen Fall mit großer Sicherheit portiert werden. Wenn Sie diesen Hinweis nur beim »Überfliegen« dieser Einleitung in einer Buchhandlung gefunden haben, empfehle ich Ihnen dringend, den Unterpunkt e.1 dieses Kapitels ebenfalls zu lesen!

- Ebenfalls hilfreich ist es, wenn Sie Zugriff auf ein digitales Speicheroszilloskop haben. Ob es sich hierbei um ein eigenständiges Gerät oder um eine Softwarelösung handelt, spielt keine Rolle. Die Verfügbarkeit eines solchen (in der Regel sehr teuren) Messgerätes ist aber auch nicht zwingend erforderlich: Dort, wo es wichtig ist, bietet das Buch Fotografien von Oszillogrammen.
- Sehr zu empfehlen ist die Anschaffung eines sogenannten In-Circuit-Emulators (ICE). Eine solche Hardware unterstützt das Debuggen von Programmen erheblich. Es ist zwar ebenfalls möglich, Ausgaben der Programme über die serielle Schnittstelle in einem Terminal-Emulator (unter Windows z.B. putty oder TeraTerm) anzuzeigen: Auf Dauer ist dies aber unpraktisch und umständlich!

Weitere Empfehlungen und Hinweise

Um den größtmöglichen Nutzen aus diesem Buch ziehen zu können, sollten Sie zusätzliche Ausgaben von zunächst ungefähr 50 bis 150 Euro einkalkulieren. Die Höhe der Aufwendungen richtet sich nach Ihren besonderen Interessengebieten. Sie sind beispielsweise nicht dazu gezwungen, sämtliche Kapitel dieses Buches durcharbeiten: Welche Kapitel für Sie von Bedeutung sind und welche nicht, hängt von Ihren persönlichen Anforderungen ab. Um die Kosten so gering wie möglich zu halten, habe ich mich in den Beispielprojekten auf Komponenten beschränkt, deren Beschaffung mit weniger als 10 Euro zu Buche schlägt.

Ergänzende Literatur zu diesem Buch finden Sie in Form von Datenblättern in ausreichender Zahl im Internet. Hilfreich sind auch viele Foren: Besonders positiv ist die Webseite <http://www.stackoverflow.com> hervorzuheben, weil hier eine sehr angenehme Kommunikationskultur durch die Forenmitglieder gewahrt wird.

Sie werden in diesem Buch nahezu alle internen Komponenten des AT91SAM3S4B kennenlernen. Ausgenommen ist nur der USB Device Port, da seine effiziente Programmierung extrem aufwendige Maßnahmen, wie z.B. einen Protokoll-Analyser, erfordern. Bei den seriellen Schnittstellen wurden einige recht detailliert, andere weniger detailliert beschrieben. Dies liegt vor allem daran, dass zur Erlangung vollwertiger Spezifikationen unter Umständen mehrere Tausend

Dollar berappt werden müssen oder dass weitere Hardware beschafft werden muss.

Um nicht einfach nur das Datenblatt zu kopieren, wird es für die Arbeit mit diesem Mikrocontroller als Nachschlagewerk hinzugezogen. Auf diese Weise wird erreicht, dass auch die Dokumentation anderer als des verwendeten Mikrocontrollers und weiterer benötigter Hardwarekomponenten verständlicher wird und sich ein zukünftiger Umstieg auf andere Hardware leichter gestaltet.

Als besonders wichtig erscheint mir die korrekte Dimensionierung externer Komponenten, wie z.B. Widerstände, Dioden (auch LEDs), Transistoren usw. Damit der Mikrocontroller durch den Anschluss dieser Komponenten und weiterer Geräte (7-Segment-Anzeigen, LC-Displays, Relais, DC-Motoren, ...) nicht beschädigt oder gar zerstört wird, wird anhand bewusst einfach gehaltener Berechnungsbeispiele gezeigt, wie externe Komponenten dimensioniert werden müssen.

Aufbau des Buches

Diese Einleitung soll Ihnen die Orientierung in diesem Buch erleichtern. Natürlich dienen hierzu in erster Linie das Inhaltsverzeichnis und der Index; hier wird Ihnen darüber hinaus aber auch ein grober Umriss gezeigt, wovon die einzelnen Kapitel handeln. Auf diese Weise soll Ihnen die Entscheidung erleichtert werden, welche der folgenden Kapitel wichtig für Sie sind und welche Kapitel Sie zu einem späteren Zeitpunkt bearbeiten können: Ich habe bei der Konzeption Wert darauf gelegt, dass jedes Kapitel in sich abgeschlossen ist, und Verweise auf andere Kapitel zu vermeiden versucht. Das Buch ist in mehrere Teile gegliedert, die in sich abgeschlossene Aspekte getrennt betrachten.

Teil I

Teil I des Buches befasst sich mit den Themen, die für einen problemlosen Aufbau eines Entwicklungssystems zuständig sind.

In Kapitel 1 erhalten Sie einen Überblick über verfügbare und frei nutzbare, aber auch über kommerzielle Entwicklungsumgebungen, die Sie für die Entwicklung eigener Projekte einsetzen können. Der Schwerpunkt liegt hierbei auf freien Entwicklungsumgebungen, da die Beschaffung kommerzieller Varianten mit erheblichen Kosten verbunden ist. Darüber hinaus stellt Kapitel 1 eine Übersicht über sinnvolle Hardware bereit, über deren Anschaffung Sie zumindest nachdenken sollten. Den Abschluss von Kapitel 1 bildet die Installation von Eclipse mitsamt aller Plug-ins, die für die Entwicklung eigener Mikrocontroller-Projekte erforderlich sind.

Kapitel 2 liefert zunächst ein paar grundlegende Informationen zur Firma ARM Holdings PLC und ihrer Geschäftsidee. Den größten Anteil nimmt aber die

Beschreibung der Erzeugung von Bibliotheken zum Entwicklungsboard Olimex SAM3-P256 ein sowie dem darauf verwendeten Baustein ATMEL SAM3S4B.

Kapitel 3: Die Konfiguration von Eclipse, der verschiedenen Softwarekomponenten Compiler, Linker usw. ist zwar nicht sonderlich schwierig, aber umfangreich. Wenn Sie häufig Projekte auf der Basis dieses Cortex-M3-Bausteins entwickeln, wird Ihnen Kapitel 3 sehr behilflich sein: Hier wird eine Schablone entwickelt, die Ihnen die Konfiguration von Eclipse bei weiteren Projekten erspart. Diese Schablone enthält sämtliche Einstellungen für den Compiler, den Linker und für weitere Software, deren Einsatz erforderlich wird. Auch die Konfiguration des Debuggers ist Bestandteil dieses Kapitels, sodass Sie durch einfaches Importieren der Schablone in eigene Projekte diese umfangreichen Aufgaben nie wieder durchführen müssen.

Kapitel 4: Auf der Basis der in Kapitel 3 entwickelten Schablone wird das von ATMEL/Olimex gelieferte Beispielprogramm Blinky neu entwickelt. Dabei stehen folgende Aspekte im Vordergrund: die Arbeit mit der Schablone sowie mein Verständnis von einem guten Programmierstil.

Teil II

Das wichtigste Anwendungsgebiet von Mikrocontrollern ist die Steuerung beliebig großer Maschinen. Ob es sich hierbei um einfache Kaffeemaschinen oder andere Haushaltsgeräte handelt oder um komplexe Maschinen in industriellen Produktionsbetrieben: Immer werden externe Hardwarekomponenten (Motoren, Anzeigergeräte, Tastaturen, Lichtschranken und Lichttaster etc.) an Mikrocontroller angeschlossen. Automobile sind heute ohne den Einsatz von Mikrocontrollern gar nicht mehr vorstellbar: ABS, ESP, Servolenkung, Navigationssysteme, das sogenannte Infotainment in Autos usw. stellen einen Luxus dar, auf den heute vermutlich kein Autokäufer mehr verzichten möchte. In diesem Teil werden wichtige Aspekte zur korrekten Dimensionierung zusätzlicher Elektronikkomponenten vorgestellt, damit die vorhandene Hardware zuverlässig und sicher funktioniert. Ganz nebenbei wird auch die Anwendung von Datenblättern der verschiedenen elektronischen Bauelemente eingeführt und immer wieder geübt.

Kapitel 5 beginnt mit einer Übersicht über die Ausstattungsvarianten sämtlicher Mitglieder der ATMEL-SAM3S-Mikrocontroller-Familie. Das wichtigste Dokument, das Sie für die erfolgreiche Hard- und Softwareentwicklung für die Mikrocontroller der SAM3S-Familie von ATMEL benötigen, das Datenblatt, wird im Anschluss beschrieben. Danach werden die wichtigsten elektrischen Eigenschaften, die sogenannten Electrical Characteristics, des Mikrocontrollers ATMEL SAM3S4B vorgestellt. Um Schäden an Menschen und Maschinen vorzubeugen, ist die Beachtung dieser Kennwerte von größter Bedeutung. Ein weiterer wichtiger Aspekt bei der Anwendung von Mikrocontrollern ist das Verhalten des Controllers im Reset-Zustand. Obwohl es nicht wirklich zu den Elektronik-

Grundlagen gehört, die in diesem zweiten Teil des Buches vorrangig betrachtet werden, soll auch der Software-Aspekt berücksichtigt werden. So werden Sie ein Programm entwickeln, mit dem Informationen aus dem Baustein herausgelesen werden können, was es ermöglicht, verschiedene Ausführungsvarianten des gleichen Gerätes innerhalb einer Firmware zu realisieren.

Kapitel 6 befasst sich mit den elektronischen Grundlagen, die bei der Ansteuerung externer Geräte (Output) und bei der Erfassung von Zuständen (Input) dieser Geräte zu beachten sind. Dabei werden zunächst die sogenannten *diskreten Größen* betrachtet, die von den *kontinuierlichen Größen* unterschieden werden. Dabei ist es wichtig, den Zustand des Mikrocontrollers unmittelbar nach dem Reset zu betrachten bzw. welche Folgen ein mechanischer Defekt des Reset-Tasters hat. Mit einem einfachen Programm wird diese Frage beantwortet. Darüber hinaus befasst sich dieses Kapitel mit der korrekten Dimensionierung von externer Hardware, die mit dem Mikrocontroller gesteuert bzw. deren Zustand abgefragt werden soll. Verfahren, wie das sogenannte *Current Sourcing* und das *Current Sinking* werden genauso behandelt, wie die Dimensionierung von nachgeschalteten Transistoren, wenn der Mikrocontroller nicht in der Lage ist, die angeschlossene Last direkt anzusteuern. Die Ansteuerung induktiver Lasten (Motoren, Relais, Elektromagnete) und die Beschreibung von Maßnahmen zum Schutz des Mikrocontrollers schließen Kapitel 6 ab.

In Kapitel 7 wenden wir uns erstmals konkreten Anwendungen, wie z.B. der Ansteuerung von LC-Displays und 7-Segment-Anzeigen, zu. Der Fokus liegt hier auf dem Einsatz der Ports als Ausgang. Die Verwendung der Ports als Eingang wird ab Kapitel 9 beschrieben.

Teil III

Kapitel 8 ist wieder etwas theoretisch, da hier aufgrund noch fehlender Kenntnisse kein praktisches Beispiel entwickelt wird. Dennoch ist die Behandlung von Interrupts und die Vorstellung des sogenannten Nested Vector Interrupt Controllers (NVIC) von extrem großer Wichtigkeit, um die Mikrocontroller der SAM₃S-Familie effizient nutzen zu können (eigentlich gilt das hier Ausgeführte für alle anderen Mikrocontroller auf dem Markt in gleichem Maße). Darüber hinaus werden die Komponenten PMC (Power Management Controller), Taktgenerator (Clock Generator) und Supply Controller (SUPC) vorgestellt.

Kapitel 9 befasst sich ausführlich mit den Registern der PIO-Controller. Neben ihren Aufgaben und ihrer Adressierung wird auch gezeigt, wie sie in Programmen genutzt werden. Prinzipiell gibt es verschiedene Möglichkeiten, dies zu realisieren: der Zugriff über vordefinierte Registernamen, der Zugriff über C-Strukturen sowie der Zugriff über die ATMEL-Version von CMSIS. Alle diese Zugriffsmöglichkeiten werden in diesem Kapitel (und für die anderen Peripheriegeräte in den folgenden Kapiteln) gezeigt.

Die Kapitel 10 und 11 befassen sich mit Timern und Countern. Die Aufteilung auf zwei Kapitel kommt zunächst einmal daher, dass beide Kapitel relativ umfangreich sind. Der zweite Grund ist: Meiner Meinung nach sind die in Kapitel 11 beschriebenen TC (Timer/Counter) viel universeller einsetzbar im Vergleich zu den relativ einfachen Komponenten, die in Kapitel 10 beschrieben werden. Diese Universalität rechtfertigt die herausgehobene Behandlung in einem eigenständigen Kapitel.

Teil IV

Kapitel 12 beschreibt den sogenannten Peripheral DMA Controller (PDC). Häufig ist es erforderlich, große Datenmengen zu verarbeiten, was bei herkömmlicher Programmierung die CPU des Mikrocontrollers stark belasten kann. Mithilfe des PDC haben viele interne Peripheriegeräte aber einen von der CPU unabhängigen Zugang zu Massenspeichern, sodass auch die Verarbeitung größerer Datenmengen die CPU kaum oder gar nicht belastet.

Die Pulsweitenmodulation PWM, die in Kapitel 13 behandelt wird, ist eine sehr gängige Möglichkeit, externe Geräte, wie z.B. LEDs oder Lampen, DC-Motoren oder Audiosignale, zu beeinflussen. Bei LEDs oder Lampen lässt sich beispielsweise die Lichtstärke durch Dimmen gezielt einstellen, bei DC-Motoren wird PWM häufig zur Regelung der Drehzahl eingesetzt.

Eines der Hauptanwendungsgebiete von Mikrocontrollern ist die Erfassung physikalischer Größen, wie z.B. Temperatur, Kraft (und hiermit auch Drehmomente), Druck oder Feuchtigkeit. Hierbei handelt es sich um sogenannte analoge Größen, die, bevor sie mit einem Mikrocontroller verarbeitet werden können, zunächst in digitale Werte umgesetzt werden müssen. In Kapitel 14 lernen Sie den Einsatz von Analog-/Digital-Umsetzern genauso kennen wie auch den umgekehrten Weg der Digital-/Analog-Wandlung. Hierzu werden DAC, ACC und ADC vorgestellt.

Teil V

Ein ebenfalls sehr wichtiges Gebiet betrifft die Kommunikation von Mikrocontrollern und Geräten über serielle Schnittstellen. In den Kapiteln 15 und 16 werden die meisten seriellen Schnittstellen beschrieben. Aufgrund von sehr vielfältigen Einsatzmöglichkeiten, die mit der Beschaffung zusätzlicher Hardware verbunden sind, habe ich hier aber nur aufgezeigt, wie die Komponenten nutzbar gemacht werden können. So werden beispielsweise in Kapitel 16 nur die wesentlichen Grundlagen zum Einsatz von SD-Karten mit dem SPI-Protokoll aufgezeigt. Eine vollständige Anwendung ist aus Platzgründen aber gar nicht möglich. Immerhin werden Ihnen hier konkrete Hinweise geliefert, wo Sie Zusatzinformationen finden können, wenn Sie ein bestimmtes Anwendungsgebiet in besonderem Maße interessiert. Auf die Beschreibung des USB Device Ports habe ich aus guten Gründen verzichtet. Zunächst gilt: Wer den USB-Port professionell nutzen möchte,

benötigt ganz andere Angaben, die nur gegen Zahlung sehr hoher Gebühren erhältlich sind. Darüber hinaus ist das Entwickeln von USB-Geräten ohne einen ebenfalls sehr teuren Protokoll-Analyzer kaum möglich. Da mir selber solche Geräte nicht zur Verfügung stehen, musste ich auf die Beschreibung notgedrungen verzichten.

Anhänge

Anhang A: Glossar

Anhang B: Hardwarebeschaffung / Internetlinks

Anhang C: Ein sehr ausführlicher Index

Anhang D: Erfahrungen

An dieser Stelle bleibt mir nur noch der Wunsch, dass Ihnen dieses Buch eine wertvolle Hilfe bei der Realisierung eigener Projekte sein wird.

Teil I

Grundlagen

In diesem Teil:

- **Kapitel 1**
Orientierung 27
- **Kapitel 2**
ARM und CMSIS 53
- **Kapitel 3**
Das erste Eclipse-Projekt 71
- **Kapitel 4**
Hello World! III

Orientierung

In diesem Buch wird eine *Entwicklungsumgebung* (*Entwicklungsplattform* = IDE, *Integrated Development Environment*) für Cortex-M3-Mikrocontroller eingesetzt, die auf der frei verfügbaren Eclipse-Plattform basiert. Doch diese Plattform ist bei Weitem nicht die einzige, die für die Programmierung von Cortex-M3-Mikrocontrollern herangezogen werden kann. Im Gegenteil: Die Zahl der verfügbaren Plattformen ist nahezu unüberschaubar! Je nach Anspruch (und finanziellen Möglichkeiten) haben Sie die freie Wahl zwischen

- kommerziellen IDEs, deren Anschaffung aber unter Umständen mit erheblichen Kosten verbunden ist,
- herstellergebundenen, aber kostenlosen, IDEs und
- freien Entwicklungsumgebungen.

In den folgenden Abschnitten dieses Kapitels gebe ich Ihnen einen kleinen Überblick über die am Markt verfügbaren Entwicklungsumgebungen. Dieser Überblick erhebt bei Weitem keinen Anspruch auf Vollständigkeit! Es werden aber die meiner Meinung nach am weitesten verbreiteten Vertreter dieser drei Gruppen kurz vorgestellt.

1.1 Kommerzielle IDEs

Viele Entwickler von Mikrocontrollerschaltungen (hier sind besonders die im industriellen Umfeld tätigen Unternehmen gemeint) legen großen Wert auf einen guten Support durch die Hersteller von Entwicklungsumgebungen. In diesem Umfeld spielen Lizenzkosten, die bis zu 5.000 Euro betragen können, eine eher untergeordnete Rolle, zumal die Hersteller auch Mehrbenutzerlizenzen mit großzügigen Rabatten anbieten, was den Durchschnittspreis pro Lizenz beim Kauf mehrerer Lizenzen erheblich reduzieren kann. In den meisten Fällen bieten die kommerziellen Anbieter auch Evaluierungsversionen ihrer Software an. Diese sind nach erfolgter Registrierung in der Regel kostenlos verfügbar und entsprechen mit Einschränkungen häufig den Vollversionen. Diese Einschränkungen betreffen beispielsweise den Nutzungszeitraum oder die Größe des erzeugten Codes. Fast niemals dürfen diese Softwareversionen produktiv, das heißt kommerziell, eingesetzt werden. Auch die Nutzung für Ausbildungszwecke ist hiermit häufig untersagt, da beispielsweise Lehrer durch das Erteilen des Unterrichts Geld

verdienen: Diese Arbeit wird von den meisten Herstellern durch sogenannte Educational Licenses zu Vorzugspreisen unterstützt. Über den Unterricht hinausgehende kommerzielle Nutzung dieser Lizenzen ist aber auch dann untersagt. Einige Beispiele für kommerzielle Entwicklungsumgebungen sowie die Links zu den Herstellerseiten finden Sie in den folgenden Abschnitten.

1.1.1 Keil μ Vision

Die Firma Keil, früher ein eigenständiges Unternehmen, wurde inzwischen von der Firma ARM übernommen. Bei *Keil μ Vision* handelt es sich um eine IDE, die von Haus aus die Chips unterschiedlicher Hersteller direkt unterstützt. Nach der Entscheidung für einen Chip können Sie die für Ihr Projekt benötigten Komponenten eines Chips durch einfaches Anklicken in das Projekt übernehmen, wobei die Initialisierung und Konfiguration dieser Komponenten über eine bequem zu nutzende Benutzeroberfläche durchgeführt werden kann. Als Folge dieser Auswahl »baut« die IDE dann das Gerüst Ihres Projekts auf, sodass Sie sich »nur noch« um die eigentlichen Funktionalitäten kümmern müssen. Für jeden von dieser IDE unterstützten Chip steht unmittelbar eine ausführliche Dokumentation zur Verfügung: Die häufig mühsame Suche nach der korrekten Dokumentation entfällt somit. Eine Vielzahl von Beispielanwendungen erleichtert die Entwicklung eigener Projekte. Abgerundet wird die Software durch eine Debugging-Software, die es ermöglicht, den Zustand des Mikrocontrollers bis hinein in die internen Register der CPU zu kontrollieren. Weitere Informationen finden Sie unter <http://www.keil.com>.

1.1.2 IAR Workbench

Genauso mächtig und umfangreich ist die Workbench der Firma IAR, was bei professionellen Entwicklern ebenfalls zu einer großen Beliebtheit dieser IDE geführt hat. Neben der Unterstützung der gesamten Produktpalette von ARM-Bausteinen (angefangen beim ARM7TDMI bis hin zum Cortex-A 15) gehört auch hier eine vollständige Dokumentation aller unterstützten Bausteine. Mehr als 3.100 Anwendungsbeispiele lassen das Herz jedes Entwicklers höher schlagen. Die Bedienung der Workbench ist vergleichbar strukturiert wie bei Keil und ebenso einfach und intuitiv möglich. Dass die IAR Workbench ebenfalls über eine integrierte Debugger-Lösung verfügt, versteht sich von selbst. Weitere Informationen finden Sie unter <http://www.iar.com>.

1.1.3 Sourcery Codebench

Diese Software von der Firma *Mentor* basiert auf umfangreichen Erweiterungen für Eclipse unter Einsatz der GNU Compiler Collection. Die Sourcery Codebench gibt es für sehr viele verschiedene Zielplattformen: Für unseren Anwendungsfall

wäre die Version *ARM EABI* die richtige Variante. Der Einsatz kostenloser Basis-Werkzeuge, wie der genannten Eclipse-IDE und der GNU Compiler Collection, ermöglicht es der Firma Mentor, die Codebench etwas preisgünstiger anzubieten. So kostet die Professional-Version als Einzelplatzlizenz 2.800 US-Dollar (Nettopreis). Es existiert auch eine kostenlose Version: Sourcery Codebench lite. Hierbei handelt es sich aber um eine kommandozeilenbasierte Version, sodass der Komfort einer grafischen Entwicklungsumgebung (z.B. Eclipse oder NetBeans) nur durch umfangreiche Konfigurationsarbeiten erreicht werden kann. Weitere Informationen finden Sie unter <http://www.mentor.com>.

1.1.4 Atollic TrueSTUDIO

Für ca. 1.000 Euro ist das ebenfalls Eclipse-basierte *Atollic TrueSTUDIO* erhältlich. Hierbei handelt es sich um die Professional-Version, die kostenlose Version *Atollic TrueSTUDIO Lite* ist ebenfalls verfügbar. Neben der Unterstützung von mehr als 1.300 ARM-basierten Mikrocontrollern werden noch ungefähr 1.000 Beispielanwendungen mitgeliefert, sodass auch hier mehr als ausreichend Vorlagen für die Umsetzung eigener Projektideen genutzt werden können.

Es existieren noch weitere kommerzielle Entwicklungsumgebungen. Die vier oben genannten sind aber die – meiner Einschätzung nach – am weitesten verbreiteten. Dies äußert sich auch bei Recherchen im Internet durch die Vielzahl von privat geführten Foren, in denen umfangreicher Support geleistet wird.

1.1.5 CrossWorks for ARM

CrossWorks for ARM ist ebenfalls eine auf Eclipse basierende kommerzielle Entwicklungsplattform, die von der Firma *Rowley Associates* entwickelt wird. Wie die bereits vorgenannten kommerziellen Entwicklungsumgebungen, liefert auch *CrossWorks for ARM* sozusagen »alles aus einem Guss«. *Rowley Associates* bietet die IDE in drei Varianten an, die sich nur im Lizenzmodell unterscheiden. Hierbei handelt es sich um

- eine persönliche Lizenz, die sich an Hobby-Programmierer und Schüler bzw. Studenten wendet. Wie es der Name bereits andeutet, ist diese Lizenz an eine Person (und nicht an einen Computer, der von mehreren Personen genutzt wird) gebunden. Bei einem Preis von 150 US-Dollar ist diese Version sehr günstig, wobei die kommerzielle Nutzung ausgeschlossen ist.
- eine Lizenz zu Ausbildungszwecken für Lehrer und Dozenten. Diese kostet 300 US-Dollar und darf ausschließlich für den Zweck der Ausbildung genutzt werden. Im Gegensatz zur persönlichen Version ist diese Lizenz an einen Computer gebunden, der von verschiedenen Personen genutzt werden darf. Die Entwicklung darüber hinausgehender kommerzieller Anwendungen ist nicht zulässig.

- eine Profi-Lizenz zu einem Preis von 1.500 US-Dollar. Auch diese Lizenz ist wieder personengebunden. Die kommerzielle Nutzung ist mit dieser Version gestattet.

1.2 Herstellergebundene IDEs

Viele Hersteller von Mikrocontrollern haben erkannt, dass sie den Nutzen ihrer Produkte für die Anwender erheblich steigern können, wenn sie ihren Kunden eine eigene kostenlose Entwicklungsumgebung anbieten. Natürlich sind diese nicht so universell einsetzbar wie die kommerziellen Varianten: Sie beschränken sich im Regelfall auf die eigene Produktpalette, was ich allerdings für absolut legitim halte. Eine kleine Auswahl der bekanntesten herstellerebenen Entwicklungsumgebungen finden Sie in den folgenden Abschnitten.

1.2.1 ATMEL Studio 6

Da das von mir gewählte Entwicklungsboard den ATMEL-Chip SAM3S4B einsetzt, beginne ich bei der Beschreibung auch mit dem von ATMEL bereitgestellten *ATMEL Studio 6*. Bei dieser IDE handelt es sich um eine Entwicklungsumgebung, die sehr übersichtlich aufgebaut ist und eine besonders komfortable Fehlersuche (Debuggen) ermöglicht: Mit dieser IDE kann man während der Laufzeit »in den Prozessor« hineinschauen und sehen, was sich dort gerade abspielt. Das ATMEL Studio 6 basiert auf *Microsofts Visual Studio*, und bietet somit alle Möglichkeiten, die viele Entwickler im Microsoft-Umfeld in anderen Situationen kennen- und schätzen gelernt haben. Der möglicherweise einzige Nachteil im Vergleich zu den meisten anderen IDEs ist, dass sie ausschließlich für Windows-Betriebssysteme von Microsoft verfügbar ist. Bei einem Marktanteil von ca. 90% muss dieser Nachteil aber nicht gravierend sein. Wer mit einem Windows-Betriebssystem arbeitet und einen ATMEL-Chip einsetzt, sollte diese Entwicklungsumgebung auf jeden Fall in Betracht ziehen. Dies gilt umso mehr, als man nicht zwingend auf ein Evaluierungsboard von ATMEL angewiesen ist und beispielsweise selbst entwickelte Boards einsetzen kann. Da bereits ein Browser in die IDE integriert ist, kann man nahtlos und ohne Einsatz weiterer Programme auf Internetressourcen zugreifen. Der Link zu dieser IDE lautet <http://www.ATMEL.com/tools/ATMELstudio.aspx>.

1.2.2 Texas Instruments StellarisWare

StellarisWare ist eine Entwicklungsumgebung, die von der Firma *Texas Instruments* für die Programmierung sämtlicher Stellaris-MCUs (MCU = Microcontroller Unit) kostenlos und ohne bekannte Einschränkungen zur Verfügung gestellt wird. Auch hier stehen sämtliche denkbaren und für den effizienten Einsatz notwendigen Hilfsmittel einschließlich der Dokumentation bereit. Zum Einsatz dieser Ent-