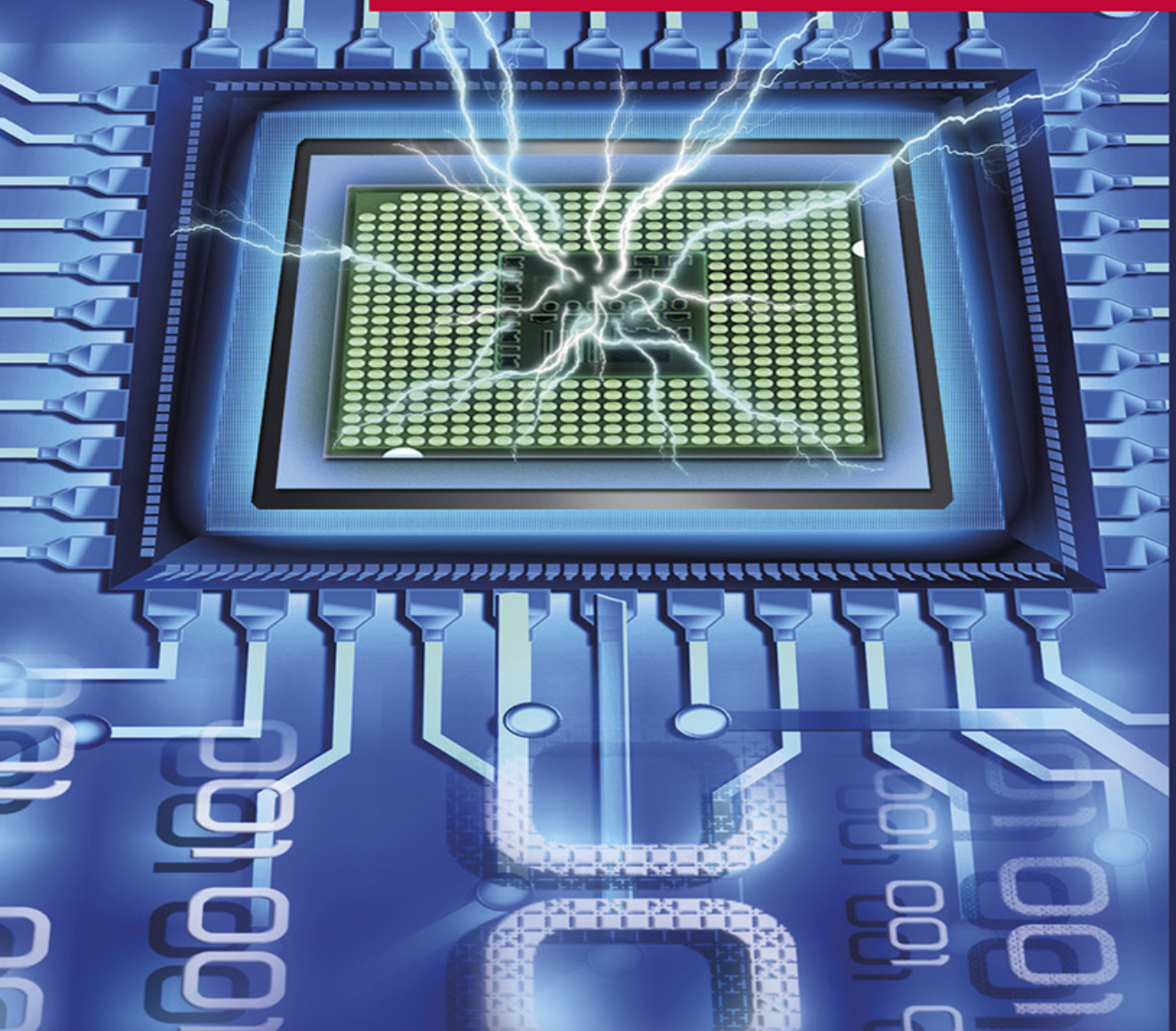
mitp

Ralf
Jesse

STM32

ARM-Mikrocontroller programmieren für Embedded Systems

Das umfassende Praxisbuch



Inhaltsverzeichnis

Ralf Jesse

STM32

ARM-Mikrocontroller programmieren für Embedded Systems

Das umfassende Praxisbuch

Impressum

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 978-3-7475-0193-1
1. Auflage 2021

www.mitp.de
E-Mail: mitp-verlag@sigloch.de
Telefon: +49 7953 / 7189 - 079
Telefax: +49 7953 / 7189 - 082

Lektorat: Sabine Schulz
Sprachkorrektorat: Sibylle Feldmann
Coverbild: © Edelweiss / stock.adobe.com
electronic **pub**lication: III-satz, Husby, www.drei-satz.de

Einleitung

In diesem Buch wird die Programmierung von Mikrocontrollern der STM32F4xx-Familie von STMicroelectronics behandelt. Sie gehören zur Gruppe der *Cortex-M4-Controller*, die von Arm Limited entwickelt wurden. Die Namen der beiden genannten Unternehmen werden im weiteren Verlauf verkürzt als *STM* bzw. als *Arm* bezeichnet.

Arm ist demnach der **Entwickler** des Mikrocontrollerkerns, der **Hersteller** des käuflich zu erwerbenden Mikrocontrollers aber ist die Firma STM. STM lizenziert die Entwicklungsarbeit von Arm und nutzt somit deren sogenannte *Intellectual Property* (geistiges Eigentum). Und hierin liegt auch ein wichtiger Geschäftsbereich von Arm: Gegen die Zahlung von Lizenzgebühren überlässt Arm den Herstellern der Mikrocontroller das Recht an der Nutzung seines geistigen Eigentums, die den Prozessorkern dann um eigene Komponenten erweitern. Dass ich an dieser Stelle nur ganz allgemein von Cortex-Mikrocontrollern spreche, geschieht ganz bewusst: Denn Arm hat nicht nur Cortex-M-, sondern auch Cortex-A- und Cortex-R-Mikrocontroller und weitere entwickelt. Alle genannten Typen sind wiederum in Gruppen unterschiedlicher Leistungsfähigkeit unterteilt, sodass STM insgesamt mehr als 600 verschiedene Cortex-Mikrocontroller anbietet.

Hinweis

Dieses Buch befasst sich – wie bereits oben gesagt – ausschließlich mit den Cortex-M4-Mikrocontrollern von

STM.

STM ist nicht der einzige Hersteller von Cortex-Mikrocontrollern: Basierend auf dem Arm-Kern sind auch NXP, Microchip, Texas Instruments, Toshiba, Infineon und viele weitere Unternehmen Hersteller von Cortex-Mikrocontrollern und somit Kunden von Arm.

Warum STM32F4?

Es gibt verschiedene Gründe, die mich zu dem Einsatz von STM32F4-Mikrocontrollern bewogen haben:

- In den meisten Unternehmen, in denen ich seit mehr als 30 Jahren als Softwareentwickler im Mikrocontrollerbereich arbeite bzw. inzwischen gearbeitet habe, werden seit vielen Jahren Mikrocontroller von STM eingesetzt.

- In den einschlägigen Internetforen sind sehr viele Informationen und Hilfestellungen in Form von Tutorials zu finden. Im Anhang werde ich Ihnen einige Internetadressen nennen, die ich persönlich als besonders hilfreich empfinde.

- Die (englischsprachige) Dokumentation von STM empfinde ich als vorbildlich und klar strukturiert.

- Einer der wichtigsten Gründe besteht darin, dass STM sehr preisgünstige Evaluierungsboards vertreibt. Das in diesem Buch überwiegend eingesetzte Evaluierungsboard *NUCLEO-F446RE* ist bei einem

weltbekannten Onlinehändler bereits zu einem Preis von weniger als 20 Euro erhältlich. Ein Debugger mit Vorrichtung zum Flashen der Software ist hier bereits enthalten!

Hinweis

Der STM32F446RE zählt zu den leistungsstärksten Cortex-M4-Controllern von STM. Dabei hat es STM geschafft, die verschiedenen Mitglieder dieser Familie weitestgehend kompatibel zueinander zu halten. Dies bedeutet, dass die meisten Beispiele, die Sie in diesem Buch sowie auf meiner Webseite `https://www.ralf-jesse.de` finden, nur geringfügige Anpassungen benötigen und leicht auf den anderen Mikrocontrollern der STM32F4-Familie eingesetzt werden können. Unterschiede zwischen den verschiedenen Familienmitgliedern beschränken sich darauf, dass nicht immer alle Peripheriekomponenten integriert sind. Auch ihre Anzahl kann sich unterscheiden. Wichtig ist aber: Die Programmierung dieser Komponenten ist immer identisch.

Zielgruppe dieses Buchs

Ich gehe davon aus, dass jeder Leser dieses Buchs der englischen Sprache so weit mächtig ist, dass er die Originaldokumentation der Hersteller nachvollziehen kann. Dennoch erleichtert es die Entwicklungsarbeit häufig, wenn Dokumentation auch in der eigenen Muttersprache verfügbar ist.

Tipp

Um Ihnen die Suche nach Informationen im Internet zu erleichtern, habe ich in Anhang A eine nach Themen sortierte Sammlung von derzeit gültigen Internetadressen (Stand: Dezember 2020) zusammengestellt. Ich habe mich dabei auf sichere Webseiten (http**s**://...) beschränkt.

Dieses Buch wendet sich genauso an erfahrene Softwareentwickler wie auch an Studierende technischer Fachrichtungen sowie an Einsteiger in die Programmierung von Mikrocontrollern.

Hinweis

Dieser Hinweis gilt besonders für Einsteiger in die Programmierung von Mikrocontrollern: Cortex-M-Mikrocontroller gehören, unabhängig vom jeweiligen Hersteller, derzeit zu den High-End-Mikrocontrollern! Dies bedeutet nicht, dass ihre Programmierung etwa schwieriger wäre als beispielsweise bei den sehr beliebten ATmega-, PIC- oder ATtiny-Prozessoren von Microchip – sie bieten allerdings häufig erheblich mehr Peripheriekomponenten mit mehr Einsatzmöglichkeiten und sind daher komplexer.

Voraussetzungen

Sämtliche Beispiele wurden in der Programmiersprache C entwickelt. Da es sich bei diesem Buch aber nicht um ein Lehrbuch zu dieser Sprache handelt, werden mindestens mittlere Kenntnisse von C vorausgesetzt. Darüber hinaus lässt es sich in einem Buch mit limitierter Seitenzahl niemals vermeiden, auf die Originaldokumentation des Herstellers zurückzugreifen. Grundkenntnisse des technischen Englischs werden somit vorausgesetzt.

Hinweis

Obwohl dieses Buch Kenntnisse in der Programmiersprache C voraussetzt, werden im Embedded-Umfeld teilweise Techniken eingesetzt, die nur zögerlich in die C-Programmierung von PCs einfließen und daher nicht jedem C-Programmierer geläufig sind. In Kapitel 3 werde ich diese Techniken daher zusammenfassen.

Der Einsatz eines Buchs zum Erlernen der Programmierung eines Mikrocontrollers – dies gilt aber gleichermaßen für die Erlernung einer beliebigen Programmiersprache – kann nur dann erfolgreich sein, wenn die Beispiele ausprobiert und von Ihnen auch an eigene Anforderungen angepasst werden können. Sie benötigen daher neben einem Entwicklungs-PC auf jeden Fall eines der preisgünstigen Evaluierungsboards von STM und zusätzlich ein zum jeweiligen Evaluierungsboard passendes USB-Kabel, das für den Upload (Flashen) eines Softwareprojekts und zum Debuggen bei der Fehlersuche benötigt wird.

Hinweis

Die Beispiele in diesem Buch wurden alle mit dem STM32 NUCLEO-64 STM32F446RE getestet. Dies bedeutet auch, dass Peripheriekomponenten, die auf diesem Evaluierungsboard nicht vorhanden sind – hierzu zählen beispielsweise die Ethernet-Schnittstelle oder Komponenten zur Steuerung von Grafikdisplays –, in diesem Buch nicht behandelt werden: Entsprechende Beispiele will ich aber nach und nach auf meiner oben genannten Webseite nachreichen.

Aufbau des Buchs

Dieses Buch ist in mehrere Teile gegliedert. Zur besseren Orientierung folgt hier ein Überblick über den Aufbau des Buchs.

Teil I

Der erste Teil umfasst wichtige Grundlageninformationen, die für alle Nutzer der STM32F4xx-Mikrocontroller nützlich sind.

Kapitel 1 bietet zunächst einen einführenden Überblick über die Mitglieder der Cortex-M4-Mikrocontroller der Firma STM. Am Beispiel des STM32F446RE werden die Features dieser Mikrocontrollerfamilie beschrieben.

Hinweis

Wenn Sie einen anderen Mikrocontroller dieser Familie verwenden, finden Sie die entsprechenden Informationen im jeweiligen Datenblatt (Datasheet).

Im weiteren Verlauf des Kapitels wird die Aufteilung des Adressbereichs, das sogenannte *Memory Mapping*, beschrieben. Anschließend folgt eine Beschreibung der Funktionseinheiten des Cortex-M4-Kerns, der unabhängig vom Hersteller eines Mikrocontrollers immer gleich ist. Hier werden vor allem die Bussysteme, die zum Austausch von Daten zwischen den integrierten Funktionseinheiten verwendet werden, beschrieben.

In diesem Buch werden keine herstellerspezifischen Bibliotheken, wie z.B. `HAL` von STM oder `lpcopen` von NXP, beschrieben: Ihre Verwendung würde die Portierbarkeit von Software auf Mikrocontroller anderer Hersteller erheblich schwieriger gestalten.

Kapitel 2 befasst sich mit der Erstellung der *CMSIS*-Bibliothek (CMSIS = *Cortex Microcontroller Software Interface Standard*). Hierbei handelt es sich um eine Sammlung von Funktionen und Konstanten, die in diesem Buch verwendet wird und die die Basis für die im Buch gemeinsam entwickelte MCAL-Bibliothek darstellt. CMSIS ist dabei die einzige Fremdbibliothek, die hier verwendet wird. Darüber hinaus beschreibt Kapitel 2 den Bootvorgang sowie die Grundinitialisierung des Mikrocontrollers und gibt einführende Hinweise zur Einstellung des Taktsignals. Auch eine Beschreibung, die das Verständnis von Linkerscripts erleichtern soll, finden Sie in Kapitel 2.

Kapitel 3 ist ein sehr kurz gehaltenes Kapitel. Es beschreibt Vorschriften zur Programmierung, die in sicherheitsrelevanten Branchen wie z.B. der Automobilindustrie, der Luft- und Raumfahrt sowie der Kraftwerktechnologie zwingend eingehalten werden müssen.

In **Kapitel 4** werden die Register vorgestellt, die für das Reset-Verhalten und die Steuerung von Taktsignalen (*Reset and Clock Control*, *RCC*) zuständig sind. Das hier Beschriebene ist in allen Projekten zu beachten, die Sie entwickeln!

Wichtig

Die meisten Beispiele in diesem Buch habe ich sehr schlicht gehalten, damit Sie sich auf das Wesentliche konzentrieren können. Während der Entstehungsphase dieses Buchs habe ich natürlich viele Hilfsfunktionen und Bezeichner entwickelt, die ich später immer wieder verwendet habe. So habe ich im Verlauf Funktionen wie z.B. `gpioToggle(GPIO_TypeDef *port, PIN_NUM pin)`, `setSystickMillis(uint32_t *timer, uint32_t delay)` oder `bool isTimerExpired(uint32_t timer)` entwickelt. Durch die Anwendung dieser Funktionen und Bezeichner werden die Beispiele übersichtlicher und erleichtern die Konzentration auf die neuen Themen.

Teil II

In Teil II wird die Programmierung der Kernkomponenten von Mikrocontrollern behandelt. Mit Ausnahme der seriellen Schnittstellen, die in Teil III werden, lernen Sie hier unter anderem GPIOs, Timer sowie A/D- und D/A-Wandler kennen. Teil II ist der umfangreichste Teil dieses Buchs.

So befasst sich **Kapitel 5** mit der Nutzung der *General Purpose Inputs/Outputs* (*GPIO*). Der Fokus liegt hier zunächst auf ihrer Nutzung als digitale Ausgänge zur Ansteuerung externer Komponenten. Sie bieten sehr vielfältige Konfigurationsmöglichkeiten, sodass ein großer Teil den Registern der GPIOs gewidmet ist. In zwei praktischen Beispielen werden wir die in Kapitel 2 erstellte CMSIS-Bibliothek einsetzen.

In **Kapitel 6** stelle ich Ihnen verschiedene Techniken zur Abfrage externer Komponenten vor. Hierbei handelt es sich um die Techniken *Polling*, *Interrupts* und *Exceptions*. Sie lernen Gründe dafür kennen, dass Polling keine gute Lösung darstellt und Interrupts zu bevorzugen sind. In diesem Kapitel werden Sie darüber hinaus lernen, wie Sie die GPIOs durch den Einsatz sogenannter externer Interrupts als Inputs für digitale Signale nutzen können.

In **Kapitel 7** lernen Sie abschließend die sogenannten alternativen Funktionen der GPIO-Pins kennen. Standardmäßig werden die GPIOs für die Ein- bzw. Ausgabe digitaler Größen verwendet. Es gibt aber auch Komponenten, die zur Verarbeitung analoger Größen dienen. Um die Anzahl der Anschlüsse des Mikrocontrollers – und damit die Produktionskosten – so gering wie möglich zu halten, können die GPIOs so konfiguriert werden, dass auch die Verarbeitung nicht-digitaler Signale möglich ist: Zu diesem Zweck verwenden alle Cortex-M-Hersteller die *alternativen Funktionen* (die ich im Verlauf des Buchs auch als *AF* bezeichne).

Beginnend mit **Kapitel 8** werden Sie nach und nach die verschiedenen *Timer* und ihre Einsatzmöglichkeiten kennenlernen. Mit Timern sind besonders die STM-Mikrocontroller reichhaltig ausgestattet. Kapitel 8 befasst sich in verschiedenen Beispielen mit dem SysTick-Timer. Ich beginne hier ganz bewusst mit schlechten Beispielen, da Sie diese in vielen Online-Tutorials ebenfalls finden. In mehreren Schritten werden die schlechten Beispiele stetig verbessert, wobei ich Sie immer auf die besonderen Vorteile der neuen Techniken hinweise.

Neben dem SysTick- und verschiedenen Watchdog-Timern bieten die STM32F4xx-Mikrocontroller drei Klassen von Timern: Basic Timer, General-Purpose Timer und Advanced-Control Timer, die sich zwar in ihrer Mächtigkeit, aber nicht im Prinzip ihrer Programmierung unterscheiden.

Kapitel 9 befasst sich im Anschluss mit den Funktionen und der Programmierung der *Basic Timer*. Neben dem SysTick-Timer sind sie die einfachsten Varianten der verfügbaren Timer.

In **Kapitel 10** werden Sie dann an die *General-Purpose Timer* (GP-Timer) herangeführt. Sie werden bereits hier feststellen, dass Sie Konzepte, die Sie in Kapitel 9 kennengelernt haben, sehr einfach wiederverwenden können. Kapitel 10 bietet zudem eine Einführung in die sogenannte *Pulsweitenmodulation* (*PWM*), da sie eine der wesentlichen Erweiterungen der GP-Timer im Vergleich zu den Basic Timern darstellt.

Kapitel 11 schließt mit der Vorstellung der *Advanced-Control Timer* den Überblick über Timer ab.

Kapitel 12 und **13** befassen sich zum Abschluss von Teil II mit Komponenten, die es uns ermöglichen, die reale analoge

Welt mit digitalen Geräten zu erfassen bzw. zu beeinflussen. Die Themen dieser beiden Kapitel sind daher *Digital-Analog-Wandler* (Kapitel 12) und *Analog-Digital-Wandler* (Kapitel 13).

Teil III

Ein wesentlicher Bestandteil des Einsatzes technischer Geräte besteht in der Übermittlung von Daten zwischen verschiedenen Geräten und/oder Komponenten. Während beispielsweise Drucker (und teilweise auch Scanner) früher oftmals mit parallelen Schnittstellen ausgestattet wurden, haben diese heutzutage praktisch keine Relevanz mehr, sie wurden nahezu vollständig durch serielle Schnittstellen ersetzt. Sie glauben mir nicht? Dann möchte ich Sie auf zwei der wichtigsten seriellen Schnittstellen aufmerksam machen: USB und Ethernet! Waren serielle Schnittstellen früher relativ langsam – Übertragungsraten von wenig mehr als 115 kBit waren eher die Ausnahme als die Regel –, so übertragen moderne serielle Schnittstellen Daten mit einer Übertragungsrate von mehreren Hundert MBit (USB) bis hin zu GBit in den neuesten Ethernet-Entwicklungen.

Die Anwendung der beiden zuletzt genannten Schnittstellen ist nicht trivial. Auf ihre Beschreibung wird in diesem Buch daher verzichtet. Es existieren aber noch weitere serielle Schnittstellen, die, abhängig von ihrem Einsatz, immer noch genügend schnell und vor allem zuverlässig arbeiten.

Kapitel 14 bietet zunächst eine grundlegende Einführung in das Thema *serielle Kommunikation*. Hier werden einige Grundlagen vermittelt, die zum Verständnis der technischen Umsetzung elementar sind. Der große Vorteil serieller Schnittstellen besteht darin, dass der Datenaustausch

häufig über nur eine oder maximal zwei Datenleitungen erfolgt.

Mit der Besprechung und der Anwendung von *UARTs/USARTs* in **Kapitel 15** werden Schnittstellen behandelt, die überwiegend für die Kommunikation zwischen verschiedenen Geräten eingesetzt werden. Mit ihnen können Daten auch über größere Entfernungen, z.B. mehrere Hundert Meter, übertragen werden.

Über derart große Distanzen müssen aber längst nicht alle Daten transportiert werden: Sehr häufig ist es völlig ausreichend, wenn Komponenten Daten nur über Distanzen von wenigen Zentimetern austauschen. Auch zu diesem Zweck wurden serielle Schnittstellen entwickelt, von denen in **Kapitel 16** die Schnittstelle *Inter-Integrated Circuit* (*I²C*) vorgestellt wird.

Eine weitere serielle Schnittstelle mit vergleichbaren Anwendungsgebieten ist das sogenannte *Serial Peripheral Interface* (*SPI*). Dieses wird in **Kapitel 17** behandelt.

Obwohl es noch viel mehr serielle Schnittstellen gibt – allein die STM32F4-Familie unterstützt beispielsweise den CAN-Bus, I²S, SAI usw. –, werden sie in diesem Buch nicht beschrieben. Dies liegt unter anderem daran, dass zusätzlich benötigte Hardware teilweise derart speziell ist, dass sie im heimischen Labor üblicherweise aufgrund hoher Beschaffungskosten nicht verfügbar ist.

Teil IV

Im abschließenden Teil IV dieses Buchs werden noch einige wenige weitere Komponenten vorgestellt, die nicht in den anderen Teilen untergebracht werden konnten, weil sie

- teilweise übergreifend in mehreren Bereichen eingesetzt werden bzw.
- derart wichtige Aufgaben haben, dass sie durch die Auslagerung in einen separaten Buchteil besonders hervorgehoben werden können.

In **Kapitel 18** werde ich Sie daher mit dem sogenannten *Direct Memory Access* (*DMA*) bekannt machen. Diese Technik wird besonders häufig beim Austausch größerer Datenmengen verwendet (z.B. beim Abspielen von Musikdateien), weil sie den Mikrocontrollerkern vom Laden, Bearbeiten und Transferieren der Daten entlastet: Der Mikrocontroller kann in der gleichen Zeit für andere wichtige Aufnahmen parallel weiter genutzt werden.

Im letzten Kapitel (**Kapitel 19**) gehe ich mit den sogenannten *Watchdog-Timern* (*WD*) noch auf eine besondere Timer-Klasse ein, die für spezielle Aufgaben beim sicheren Betrieb von Anwendungen verwendet werden. Sie werden dann eingesetzt, wenn ein Gerät – dies kann auch eine interne Komponente des Mikrocontrollers sein – nicht ausreichend schnell arbeitet (z.B. weil Daten nicht rechtzeitig zur Verfügung stehen oder weil ein solches Gerät defekt ist). Im ordnungsgemäßen Betrieb werden WDs ständig neu geladen und dürfen niemals ablaufen. Läuft ein WD dennoch ab, weil eine Komponente »den Betrieb aufhält«, kann sein Auslösen dazu genutzt werden, die Maschine in einen sicheren Zustand zu überführen.

Anhänge

Ich habe mehrere Anhänge vorbereitet, die Sie bei der Entwicklung von Mikrocontrollersoftware unterstützt.

Da ich selbst weiterführende Literatur (auch online) genutzt habe, werde ich Ihnen in **Anhang A** eine umfassende Liste mit Internetadressen bzw. der verwendeten Literatur liefern.

Das in diesem Buch verwendete NUCLEO-64-Evaluierungsboard wird in allen Beispielen mit einer Frequenz von 16 MHz getaktet. Der STM32F446 kann aber mit bis zu 180 MHz getaktet werden. In **Anhang B** finden Sie einige Informationen zur Konfiguration höherer Taktraten.

Anhang C ist vor allem für diejenigen Leser wichtig, die sich noch am »Anfang der Lernkurve« befinden. Sie finden hier einfache Tipps zur Nutzung eines Debuggers, der Ihnen bei der Fehlersuche sehr gute Hilfe leistet.

Da wir in diesem Buch mit der MCAL eine eigene Bibliothek entwickeln werden – sie wird auch nach dem Druck des Buchs optimiert und erweitert –, kann eine Übersicht über die bereits verfügbaren Funktionen sehr hilfreich sein. **Anhang D** liefert diese Übersicht.

Hinweis

Aufgrund der zu erwartenden Änderungen und Erweiterungen der Bibliothek empfehle ich Ihnen den Einsatz der Online-Dokumentation: Sie wird permanent gepflegt, was in einem Buch naturgemäß nicht möglich ist.

Einsatz von Bibliotheken?