mitp

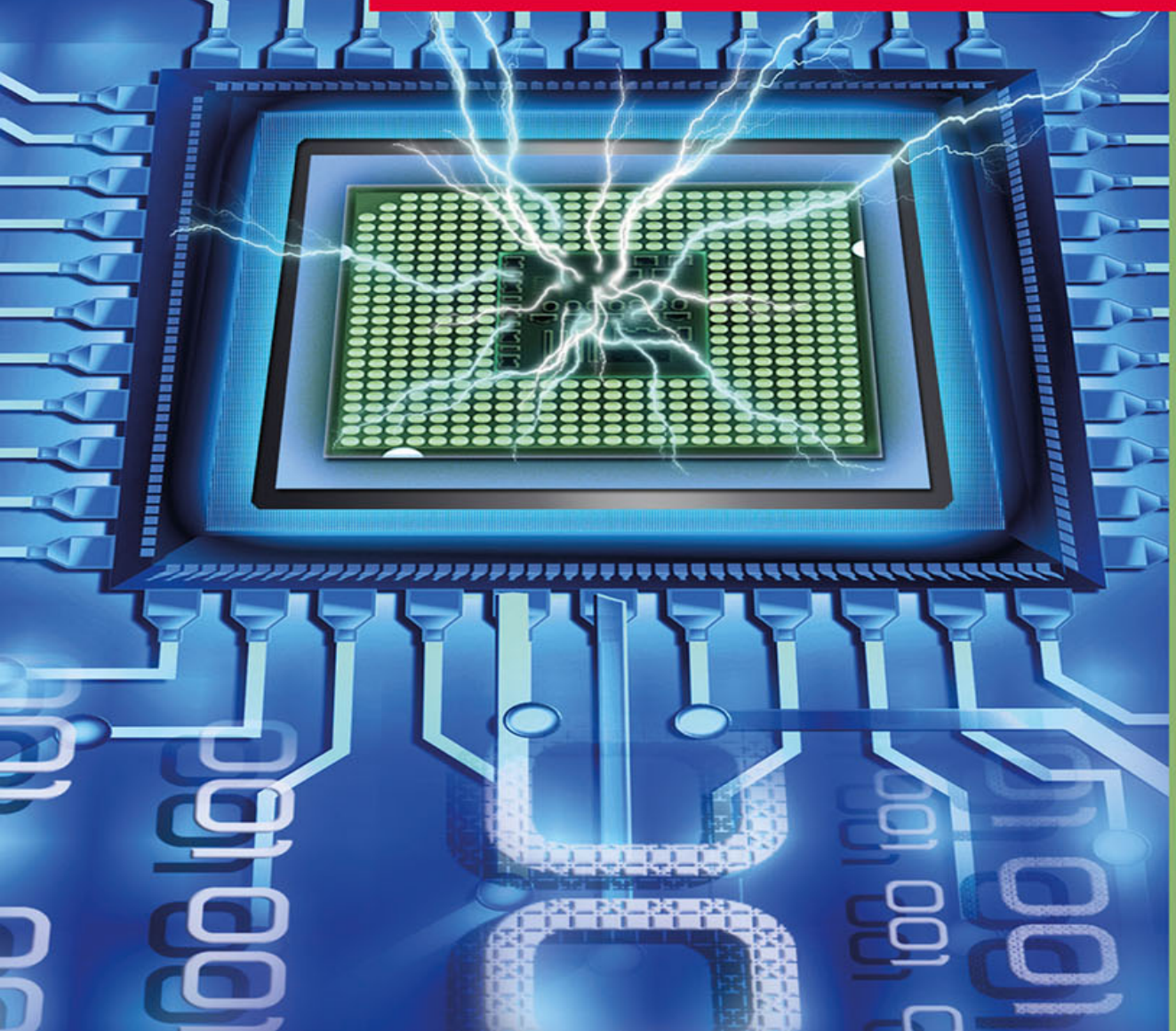
Ralf
Jesse

2. Auflage

STM32

ARM-Mikrocontroller programmieren für Embedded Systems

Das umfassende Praxisbuch



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 978-3-7475-0454-3
2. Auflage 2022

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

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

Inhaltsverzeichnis

Einleitung

Viel früher als erwartet war die erste Auflage dieses erst Ende Februar 2021 erschienenen Buches ausverkauft. Die normale Vorgehensweise wäre nun, das Buch einfach nachzudrucken und in unveränderter Form fortzuführen, da sich in derart kurzer Zeit nicht viel geändert hat/geändert haben kann. Ein vorrangiges Ziel beim Schreiben dieses Buches war aber – neben der Vermittlung der Kenntnisse zur Programmierung der STM32F4-Mikrocontroller – die Entwicklung einer Funktionssammlung in Form einer Bibliothek, die einerseits unabhängig von einem bestimmten Hersteller ist und die Sie andererseits in die Lage versetzen soll, die »Bare Metal«-Programmierung der STM32F4-Mikrocontrollerfamilie – also die Programmierung auf Registerebene – zu verstehen, sie in eigenen Projekten einzusetzen und, falls erforderlich, auf Mikrocontroller anderer Hersteller portieren zu können.

In den letzten Monaten ist diese Funktionssammlung – ursprünglich hatte ich sie einfach nur MCAL genannt, inzwischen nenne ich sie alternativ auch MCAL-STM – erheblich umfangreicher geworden. Umfasste sie in ihrer ursprünglichen Version nur 82 Funktionen, so stehen Ihnen inzwischen mehr als 240 Funktionen zur Verfügung, und ein Ende der Weiterentwicklung ist nicht absehbar! Die durchgeführten Erweiterungen führten dazu, dass viele Funktionen geändert, harmonisiert und optimiert wurden, damit sie leichter und intuitiver anwendbar sind. Dies hat dann naturgemäß zur Folge, dass ich die ursprünglichen MCAL-STM-Beispiele an die neuen Gegebenheiten anpassen musste: Mit der aktuellen Version der MCAL-STM werden die für die erste Auflage entwickelten Beispiel-Projekte überwiegend nicht mehr funktionieren.

Hierzu möchte ich Ihnen ein paar Beispiele nennen:

- Um den Bustakt von Peripheriekomponenten zu aktivieren, habe ich in der ursprünglichen Version der MCAL die entsprechenden Funktionen in den meisten Fällen mit `xxxInitXxx(...)` bezeichnet, wobei »xxx« für eine beliebige Peripheriekomponente steht (also z.B. `gpioInitGPIO(...)`). Allerdings dienen diese Funktionen nur zur Aktivierung des Bustakts der jeweiligen Komponente: Für die Initialisierung – hierunter verstehe ich die Konfiguration für den gewünschten Einsatz – werden andere Funktionen verwendet. Als Folge der Optimierung wird der Bustakt der Komponenten nun in der Form `xxxSelectXxx()`, also z.B. mit `gpioSelectGPIO()`, aktiviert. Die Timer-Funktion zum Aktivieren des Bustakts heißt nun entsprechend `timerSelectTimer()`, die von UARTs/USARTs somit `usartSelectUsart()`.

- Bei den Timern wurden Input-Capture-/Output-Compare-Funktionen ursprünglich in einer gemeinsamen Funktion behandelt. Da sich die Anzahl der verfügbaren Input-Capture-/Output-Compare-Kanäle der einzelnen Timer unterscheidet (sie liegt zwischen null und vier), hätte alleine die Behandlung der möglichen Kanal-Kombinationen acht Fallunterscheidungen erfordert. Zusätzlich müsste die Funktion aber auch prüfen, ob der ausgewählte Timer überhaupt geeignet ist, was weitere Fallunterscheidungen erfordert hätte. Ich habe mich daher letztendlich dazu entschlossen, die Input-Capture-Funktionen völlig von den Output-Compare-Funktionen zu trennen. So entstanden aus ursprünglich einer `timerSetCapCompMode()`-Funktion im Verlauf beispielsweise die Funktionen `timerSetInputCaptureMode()` bzw. `timerSetOutputCompareMode()`.

- Die GPIO-Funktionen `gpioGetPinState()` und `gpioGetPortVal()` habe ich inzwischen stark überarbeitet. Haben sie ursprünglich die gewünschten Ergebnisse in einer Variablen gespeichert, auf die über einen Pointer zugegriffen werden musste, habe ich sie nun so geändert, dass sie die Ergebnisse unmittelbar zurückliefern.

Vergleichbare Anpassungen für die anderen Peripheriekomponenten führten zu der Entscheidung, dass das Buch vollständig überarbeitet werden musste. Es gibt aber auch weitere Änderungen im Vergleich zur ersten Auflage: Eine betrifft z.B. die Einstellung der Taktfrequenzen, für die ich mit Kapitel 5 ein neues Kapitel geschrieben habe. Dies hat zur Folge, dass alle folgenden Kapitel eine neue Kapitelnummer erhalten und zudem Anhang B in seiner ursprünglichen Form entfällt.

Hinweis

Auf meiner Webseite `https://www.ralf-jesse.de` werde ich Sie natürlich auch zukünftig an den neuesten Entwicklungen teilhaben lassen. Da es sich dann aber um neue und bisher nicht beschriebene »Features« handelt, wird dieser zweiten Auflage mit großer Wahrscheinlichkeit eine längere »Lebensdauer« beschert sein.

Worum es geht

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 der wesentliche 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 erwähnt – ausschließlich mit den Cortex-M4-Mikrocontrollern von STM. Aufgrund der sehr guten Skalierbarkeit der Cortex-M-Mikrocontroller von STM lassen sich die in diesem Buch beschriebenen Techniken aber auch mit den neuen

STM32F7xx-Mikrocontrollern einsetzen! Auch Nutzer der Cortex-M0-, Cortex-M3- oder von Cortex-M23-Mikrocontrollern können von diesem Buch profitieren.

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 oder 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 eingesetzte Evaluierungsboard *NUCLEO-F446RE* ist bei einem weltbekannten Onlinehändler bereits zu einem Preis von weniger als 30 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 Buches

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

Muttersprache verfügbar ist. Meines Wissens existiert derzeit nur noch ein weiteres deutschsprachiges Buch zu STMs Cortex-M-Mikrocontrollern, das sich hauptsächlich an Anwender richtet, die erste Erfahrungen in der Arduino-Welt gesammelt haben.

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: März 2022) zusammengestellt. Ich habe mich dabei – mit einer Ausnahme – auf sichere Webseiten (http**s**://...) beschränkt.

Dieses Buch wendet sich genauso an erfahrene Softwareentwickler wie auch an Studierende technischer Fachrichtungen. Aber auch Einsteiger in die Programmierung von Mikrocontrollern sowie Umsteiger von anderen Plattformen werden hier nicht alleine gelassen. Der folgende Hinweis gilt vor allem für Einsteiger in die Programmierung von Mikrocontrollern:

Hinweis

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 Buches 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. Im Verlauf des Buches werden zunehmend auch elektronische Bauelemente verwendet, die Sie bei Bedarf zusätzlich beschaffen müssen.

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 Buches

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

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). Besonders wichtig ist, dass Sie hier zusätzlich die entsprechenden Referenzhandbücher von `https://st.com` herunterladen!

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! Besonders die Abschnitte zum RCC sind nur als Einstieg zu verstehen, da diese Komponente auch für die Einstellung der Systemtaktfrequenzen verwendet wird (siehe Kapitel 5).

Kapitel 5 wurde vollständig neu erstellt, was zur Folge hat, dass sich – im Vergleich zur ersten Auflage – alle Folgekapitel entsprechend verschieben. Hier wird die Einstellung der Taktfrequenzen des Prozessorkerns, der

verschiedenen Busse sowie der Peripheriekomponenten beschrieben. In der ersten Auflage hatte ich zu diesem Thema noch auf ein besonderes Feature der STM32CubeIDE-Entwicklungsumgebung verwiesen: Die dort beschriebene Vorgehensweise verwende ich inzwischen selbst nicht mehr. Werksseitig sind die NUCLEO-Boards so konfiguriert, dass das System mit einer Frequenz in Höhe von 16 MHz getaktet wird. In diesem neuen Kapitel lernen Sie nun, wie Sie vorgehen müssen, um die Mikrocontroller mit ihrer maximalen Frequenz betreiben zu können. Dieses Kapitel ersetzt den alten Anhang B.

Wichtig

Die meisten Beispiele in diesem Buch habe ich sehr schlicht gehalten, damit Sie sich auf das Wesentliche konzentrieren können. Während der Entstehungsphase dieses Buches habe ich viele Hilfsfunktionen und Bezeichner entwickelt, die ich später immer wieder verwendet habe und die schließlich in die selbst erstellte Bibliothek MCAL-STM aufgenommen wurden. Beispiele für MCAL-STM-Funktionen sind `gpioTogglePin(GPIO_TypeDef *port, PIN_NUM pin)`, `setSystickTicktime(uint32_t *timer, uint32_t delay)` oder `bool isTimerExpired(uint32_t timer)`. Durch die Anwendung dieser Funktionen und Bezeichner werden die Beispiele übersichtlicher und erleichtern die Konzentration auf die neuen Themen. Sie können die MCAL-STM jederzeit von der Webseite `https://gitlab.com/rjesse/mcal-stm.git` kostenlos herunterladen und erforschen, wie die enthaltenen Funktionen intern auf Bare-Metal-Basis realisiert wurden. Der Download enthält neben dem vollständigen Sourcecode der MCAL-STM zusätzlich das Verzeichnis

»doc«, in dem Sie die Dokumentation der MCAL-STM im HTML-Format finden.

Hinweis

Da die MCAL (bzw. MCAL-STM) gegenüber der ersten Auflage erheblich umfangreicher geworden ist, wurde es erforderlich, die Beispielprojekte komplett zu überarbeiten. Die weitaus meisten Beispiele bieten nun die Möglichkeit, durch Aktivieren bzw. Deaktivieren des »Softwareschalters« `#define MCAL` zwischen der Bare-Metal-Version auf der einen und der MCAL-Version auf der anderen Seite umzuschalten. Bei den einführenden sehr einfachen Beispielen der Kapitel 3 und 4 habe ich auf diese Möglichkeit noch verzichtet: Sie sind ausschließlich als Bare-Metal-Version vorhanden. Das Beispiel zur Konfiguration der Taktfrequenzen in Kapitel 5 ist hingegen in der Bare-Metal-Variante erheblich komplexer, sodass hier nur die MCAL-Version gezeigt wird.

Teil II

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

Kapitel 6 befasst sich 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 7** 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, warum Polling keine gute Lösung darstellt und weshalb 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 8** 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 oder mittels serieller Schnittstellen für den Datenaustausch zwischen verschiedenen Geräten genutzt werden. 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 anderer digitaler und nichtdigitaler Signale möglich ist: Zu diesem Zweck verwenden alle Cortex-M-Hersteller die *alternativen Funktionen* (die ich im Verlauf des Buches auch als *AF* bezeichne).

Beginnend mit **Kapitel 9** werden Sie nach und nach die verschiedenen *Timer* und ihre Einsatzmöglichkeiten kennenlernen. Mit Timern sind besonders die STM-

Mikrocontroller reichhaltig ausgestattet. Kapitel 9 befasst sich in verschiedenen Beispielen mit dem SysTick-Timer. Ich beginne hier ganz bewusst mit schlechten Beispielen, da Sie diese in vielen Onlinetutorials 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 10 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 11** werden Sie dann an die *General-Purpose Timer* (GP-Timer) herangeführt. Sie werden bereits hier feststellen, dass Sie Konzepte, die Sie in Kapitel 10 kennengelernt haben, sehr einfach wiederverwenden können. Kapitel 11 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. Hier lernen Sie dann auch die sehr mächtigen und hilfreichen Input-Capture- bzw. Output-Compare-Funktionen kennen.

Kapitel 12 schließt mit der Vorstellung der *Advanced-Control-Timer* den Überblick über Timer ab. Alles, was Sie in den bisherigen Kapiteln über Timer gelernt haben, können Sie hier auf die gleiche Weise nutzen. Zusätzlich lernen Sie aber auch neue Dinge kennen, die besonders bei der