mitp

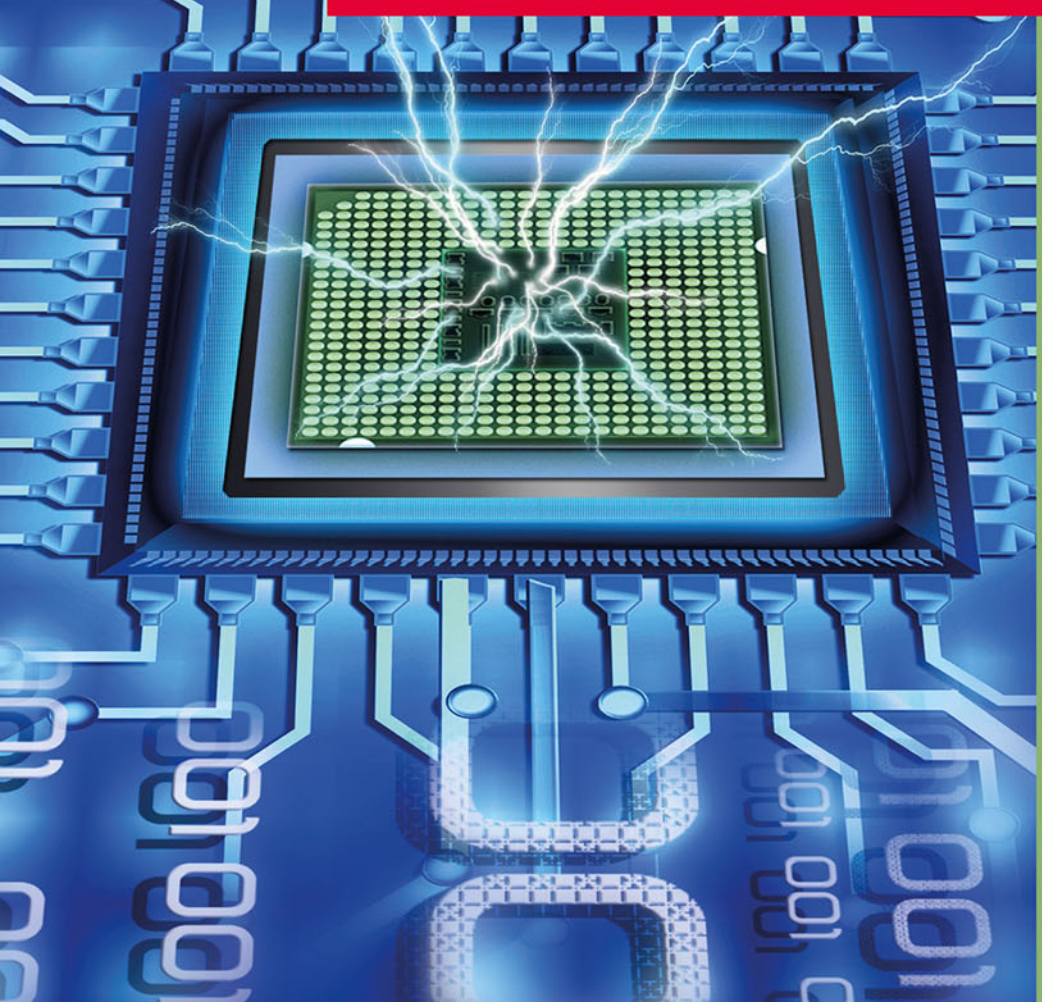
Ralf
Jesse

2. Auflage

STM32

ARM-Mikrocontroller programmieren für Embedded Systems

Das umfassende Praxisbuch



Ralf Jesse

STM32

ARM-Mikrocontroller programmieren für Embedded Systems

Das umfassende Praxisbuch

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 978-3-7475-0453-6
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
Satz: III-satz, www.drei-satz.de

Inhaltsverzeichnis

Einleitung 11
Worum es geht 12
Warum STM32F4? 13
Zielgruppe dieses Buches 14
Voraussetzungen 15
Aufbau des Buches 16
Einsatz von Bibliotheken? 21
Entwicklungsumgebungen 24
Hinweis zu den Prozessorregistern 24
Support 25
Anmerkungen des Autors 25

Teil I Grundlagen 27

1 STM32F4xx-Mikrocontroller 29
1.1 Überblick über die STM32F4xx-Familie 29
1.2 Der STM32F446 31
1.2.1 Varianten des STM32F446 31
1.2.2 Speicherbelegung/Memory-Mapping 33
1.2.3 Interner Aufbau des STM32F446 37
1.2.4 Bussysteme des STM32F446 41

2 CMSIS und MCAL erstellen, der Bootprozess, Anwendung der CMSIS 43
2.1 Warum sollte eine CMSIS-Bibliothek erstellt werden? 43
2.2 Die CMSIS-Bibliothek 45
2.2.1 Beschaffung der CMSIS-Dateien 45
2.2.2 Aufräumarbeiten 46
2.2.3 Erstellen der Bibliothek 48
2.2.4 Fertigstellen der CMSIS-Bibliothek 51
2.2.5 Einstellen von Suchpfaden 51
2.2.6 Abschlussarbeiten an der CMSIS-Bibliothek 53
2.2.7 Verwenden von CMSIS in eigenen Projekten 54

2.3 Der Bootvorgang 60
2.3.1 Der Reset-Handler 61
2.3.2 SystemInit 64
2.3.3 Grundlagen der Takterzeugung 64
2.3.4 Linkerscripts 65
2.4 MCAL / MCAL-STM 73

3 Embedded C vs. Standard C 77
3.1 Kommentare 78
3.2 Benennung von Funktionen, Variablen und Konstanten 78
3.3 Verwendung geschweifter Klammern 79
3.4 if()-Vergleiche mit Konstanten 80
3.5 Verwendung von Konstanten 81
3.6 Verwendung globaler Variablen 81
3.7 Datentypen 82
3.8 Datentypen in der STM-Dokumentation 84

4 RCC, SYSCFG und SCB 85
4.1 Reset and Clock Control (RCC) 85
4.1.1 Reset: Verschiedene Reset-Arten 86
4.2 System Configuration Controller (SYSCFG) 90
4.3 System Control Block (SCB) 91

5 Einstellung von Taktfrequenzen 95
5.1 Einflüsse der Taktfrequenz 95
5.2 Taktsystem 102
5.2.1 Orientierung im »Clock tree« 102

Teil II Kernkomponenten der STM32F4xx-Mikrocontroller 117

6 GPIO: General Purpose Input/Output 119
6.1 Features und Grenzdaten 120
6.2 GPIO-Register 121
6.3 Zwei einfache Beispiele 130
6.3.1 Rechtecksignal mit dem ODR-Register 131
6.3.2 MCAL: Neue Funktion(en) 133
6.3.3 Rechtecksignal mit BSRR 133

7 Polling, Interrupts und Exceptions 137
7.1 Allgemeines zu Polling und Interrupts 138
7.1.1 Polling 138
7.1.2 Interrupts 139
7.1.3 Interrupts beherrschen 140
7.1.4 Maskierbare und nicht-maskierbare Interrupts 141
7.1.5 Globale Interrupts 142
7.1.6 Der Nested Vector Interrupt Controller NVIC 143
7.2 Externe Interrupts 148
7.2.1 External Interrupt/Event Controller (EXTI) 149
7.2.2 Beispiel zu externen Interrupts 154
7.2.3 Erkennung mehrerer externer Interrupts 160
7.3 Exceptions 167

8 Alternative GPIO-Funktionen 169
8.1 Wiederholung 169
8.2 Aktivieren alternativer Funktionen 171
8.3 Auswahl alternativer Funktionen 171

9 System Tick Timer (SysTick) 175
9.1 Verwendung des SysTick-Timers 177
9.2 Steuerung mehrerer Funktionen 182
9.3 Steuerung mehrerer Funktionen: Ein verbesserter Ansatz 188
9.4 Register des SysTick-Timers 200

10 Timer-Grundlagen und Basic Timer 203
10.1 Allgemeines zu Timern und Countern 206
10.2 Basic Timer TIM6 und TIM7 208
10.2.1 Beispiel mit Basic Timer TIM6 212
10.3 Prescaler (Vorteiler) der Busse 217

11 General-Purpose Timer (GP-Timer) 221
11.1 GP-Timer, Teil 1: TIM9 bis TIM14 222
11.1.1 Register der GP-Timer TIM9 bis TIM14 223
11.1.2 Beispiel 1 zum Einsatz von TIM12 228
11.1.3 Beispiel 2 zum Einsatz von TIM12 232
11.2 GP-Timer, Teil 2: TIM2 bis TIM5 237
11.2.1 Einschub: Pulsweitenmodulation (PWM) 239
11.2.2 Beispiel: Dimmen einer LED mittels PWM 244

12 Advanced-Control Timer ... 251
12.1 Neue Register der Advanced-Control Timer ... 252
12.2 Einschub: Schalten induktiver Lasten ... 256
12.3 Beispiel zur Totzeitgenerierung mit dem STM32F446 ... 258
12.3.1 Das Projekt Kap11-ACTIM-01-Center-and-Edge-Align ... 258

13 Digital-Analog-Konverter ... 265
13.1 Technische Verfahren der D/A-Wandlung ... 265
13.1.1 Die Parallelwandlung ... 266
13.1.2 Das Zählverfahren ... 267
13.1.3 Das R-2R-Verfahren ... 267
13.2 DACs in der STM32F4xx-Familie ... 268
13.2.1 Datenhalteregister ... 269
13.2.2 Datenformate ... 270
13.3 Die Register des DAC ... 271
13.4 Ein einfaches Anwendungsbeispiel ... 273
13.5 Tipps für eigene Anwendungen ... 278

14 Analog-Digital-Wandlung ... 279
14.1 ADCs in der STM-Familie ... 280
14.2 Register in den STM-ADCs ... 284
14.3 Anwendungsbeispiel ... 289

Teil III Serielle Schnittstellen 295

15 Serielle Kommunikation ... 297
15.1 Grundlegende Begriffe ... 298
15.1.1 Kommunikationsrichtungen ... 298
15.1.2 Aufbau der Daten ... 299
15.1.3 Datenpegel ... 301
15.1.4 Übertragungsgeschwindigkeit ... 302
15.1.5 Übertragungsprotokolle ... 302
15.1.6 Asynchrone vs. synchrone Datenübertragung ... 303
15.2 Ausführungsformen einfacher RS-232-Schnittstellen ... 304

16 UARTs und USARTs ... 305
16.1 Was sind UARTs und USARTs? ... 306
16.1.1 Die UART/USART-Register ... 307
16.1.2 Empfangen und Senden von Daten ... 312

17 Inter-Integrated Circuit I²C 321
17.1 Die ursprüngliche Idee hinter I²C 321
17.2 Prinzipieller Aufbau einer I²C-Schaltung 322
17.3 Betriebsarten/Protokoll von I²C 325
17.3.1 Vier Betriebsarten 325
17.3.2 Das I²C-Protokoll 327
17.4 I²C in der STM32F4xx-Familie 329
17.5 Ein Beispiel mit dem PCF8574 336
17.5.1 Der PCF8574 338
17.5.2 Das Programmlisting 339
17.6 Daten lesen von I²C-Komponenten 348
17.7 Anmerkungen zu den Beispielprojekten 352

18 Serial Peripheral Interface SPI 357
18.1 Datentransfer in SPI-Interfaces 360
18.1.1 CPHA = 1 360
18.1.2 CPHA = 0 361
18.1.3 Anwendung von SPI 362
18.2 SPI-Register der STM32F4xx-Familie 362
18.3 Ein einfaches Beispiel mit dem MAX7219 368
18.3.1 Kurzbeschreibung des MAX7219 378
18.4 Eine kleine Übung 379

Teil IV Weitere Komponenten 381

19 Direct Memory Access (DMA) 383
19.1 Funktionsweise 384
19.2 DMAC(s) in der STM32F4xx-Familie 385
19.3 Beispiel: Memory → USART2 → PC mit DMA 392
19.3.1 Erläuterung der Funktionsweise 397
19.3.2 »Probleme« von DMA 399

20 Watchdog 401
20.1 Independent Watchdog (IWDG) 401
20.2 Window Watchdog (WWDG) 403
20.2.1 Funktionsweise 404
20.3 Debuggen und Watchdogs 406

Anhang

A Internetadressen und Literaturnachweise 407
A.1 Literaturnachweise 407
A.2 Infos zur STM32F4xx-Familie 408
A.3 Programmierung in C 408
A.4 Tastaturkürzel von STM32CubeIDE 408
A.5 Internettutorials 408
A.6 Support 409

B Dokumentation der MCAL 411
B.1 Grundlegender Aufbau der Dokumentation 411
B.2 Erläuterung einzelner Funktionen 412

C Einführung in das Debuggen 415
C.1 Debugger einrichten 416
C.2 Variablen »beobachten« 420
C.3 Anzeige von Prozessorregistern 422
C.4 Anzeige von SRAM-Inhalten 423
C.5 Vorsicht bei Watchdogs! 423
C.6 Ein weiteres tolles Feature 425

Stichwortverzeichnis 427

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 `xxx`**`Init`**`Xxx(...)` bezeichnet, wobei »xxx« für eine beliebige Peripheriekomponente steht (also z.B. `gpio`**`Init`**`GPIO(...)`). 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 `xxx`**`Select`**`Xxx()`, also z.B. mit `gpio`**`Select`**`GPIO()`, 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 (https://...) 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 beispiels-

weise 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 Ansteuerung von elektrischen Antrieben unter Einsatz der sogenannten H-Brücken von elementarer Bedeutung sind.

Kapitel 13 und **14** 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 13) und *Analog-Digital-Wandler* (Kapitel 14).

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 Schnitt-

stellen 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 15 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 16** 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 17** die Schnittstelle *Inter-Integrated Circuit* (*I²C*) vorgestellt wird.

Eine weitere serielle Schnittstelle mit vergleichbaren Anwendungsgebieten ist das sogenannte *Serial Peripheral Interface* (*SPI*). Es wird in **Kapitel 18** behandelt.

Obwohl es noch viel mehr serielle Schnittstellen gibt – allein die STM32F4-Familie unterstützt beispielsweise den CAN-Bus, I²S, SAI, SDIO 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 Buches 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 19** werde ich Sie daher mit dem sogenannten *Direct Memory Access* (*DMA*) vertraut 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 20**) 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 unterstützen.

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.

Anhang B 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.

Anhang C ersetzt die ursprüngliche durch Anhang D verfügbare Auflistung der MCAL-Funktionen. Hatte ich in der ersten Auflage noch sämtliche MCAL-Funktionen nach Anwendungsgebieten sortiert aufgelistet, so zeige ich Ihnen an dieser Stelle anhand ausgesuchter Beispiele nur noch, wie Sie sich in der neu erstellten HTML-Dokumentation zurechtfinden. Sie sollte bei eigenen Projekten die erste Anlaufstelle sein, wenn Sie MCAL-Funktionen verwenden wollen. Der ursprüngliche Anhang D entfällt somit vollständig!

Einsatz von Bibliotheken?

Bereits seit 2012 empfiehlt STM den Einsatz der hauseigenen *HAL-Bibliothek*, die die bei vielen Softwareentwicklern beliebte *SPL* (*Standard Peripheral Library*) ersetzen sollte. HAL hat bisher allerdings nicht nur Freunde gefunden. Ich habe den Einstieg in HAL selbst ausprobiert und bin nicht überzeugt! Meine Eindrücke sind:

- HAL ist nicht vollständig. Ich habe dies exemplarisch beim Einsatz des USART erfahren, bei dem nicht alle Daten übertragen wurden (ich habe hierfür aber eine andere funktionierende Lösung entwickelt).
- Die Dokumentation ist noch nicht ganz ausgereift.
- Wenn ein Hersteller (dies gilt nicht nur für STM) entscheidet, eine neue Bibliothek zu entwickeln und die »alte« Bibliothek nicht weiterzupflegen, entsteht bei den Anwendern irgendwann der Druck, ihre Software entweder auf die neue Bibliothek zu portieren oder sogar vollständig neu zu entwickeln. Dass dies mit erheblichen Kosten verbunden ist, liegt auf der Hand!
- Die Entwicklung einer herstellerspezifischen Bibliothek halte ich grundsätzlich für legitim: Schließlich wollen die Hersteller Kunden an ihre Produkte binden! Es ist aber auch vorstellbar, dass aus bestimmten Gründen der Umstieg auf Mikrocontroller anderer Hersteller erforderlich wird. Einer der Gründe, die mir in meiner langjährigen Berufspraxis immer wieder genannt wurden, waren hohe Beschaffungspreise der Mikrocontroller, wenn große Stückzahlen bestimmter Produkte produziert werden sollten. Ein weiterer Grund war oft, dass Kunden bereits Erfahrungen mit den Mikrocontrollern anderer Hersteller hatten und die Kosten und die Zeit für die Einarbeitung in die erforderlichen neuen Techniken scheuten. Unabhängig vom Grund gilt: Die Portierung bereits vorhandener Software würde durch den Einsatz herstellerspezifischer Bibliotheken erheblich erschwert.
- Korrekterweise darf man HAL auch nicht als Bibliothek bezeichnen: Vielmehr handelt es sich hierbei nur um eine weitere Abstrahierung, denn sie verwendet lediglich weitere noch tiefer liegende Funktionen: HAL »umhüllt« sozusagen die Low-Level-Funktionen, weshalb viele Programmierer sie nur als »Wrapper« (»Umhüller«) bezeichnen.

Ich habe mich daher für den »althergebrachten« Weg der Programmierung über die Register entschieden. Diese Vorgehensweise bringt – ganz nebenbei und unter Vermeidung der bereits erwähnten Nachteile – viele Vorteile für Sie mit sich:

- Sie lernen die Programmierung eines Mikrocontrollers »von der Pike auf« und können die erworbenen Kenntnisse später auf andere Mikrocontroller übertragen.
- Sie werden quasi gezwungen, sich intensiver mit dem *Reference Manual* von STM zu befassen, da in einem Buch von knapp 400 Seiten nicht der Inhalt von mehr als 1.300 Seiten der offiziellen Dokumentation zusammengefasst werden kann.

Hinweis

Der Einsatz von *CMSIS* (*Cortex Microcontroller Software Interface Standard*) stellt die Ausnahme von dieser selbst auferlegten Regel dar. Hierbei handelt es sich um

eine Sammlung von Funktionen, Makros und Definitionen, die den von Arm entwickelten Prozessorkern betreffen, da dieser von allen Herstellern gleichermaßen genutzt wird (eventuell mit Ausnahme der optional ebenfalls verfügbaren Fließpunkteinheit FPU). Ein weiterer Grund für den Einsatz von CMSIS besteht darin, dass hier der jeweils verwendete Mikrocontroller vollständig von seinem Hersteller beschrieben wurde, das heißt, sämtliche Registernamen aller Komponenten sind mitsamt ihren Adressen im verfügbaren Speicherbereich bereits definiert. Allein die Beschreibung des STM32F446xx umfasst ca. 16.000 Zeilen!

Hinweis

Neben der CMSIS wird im weiteren Verlauf natürlich auch die selbst entwickelte MCAL-STM-Bibliothek verwendet. Die MCAL-Bibliothek ist zwar immer noch nicht fertig, sie ist aber – besonders im Vergleich zur ersten Auflage dieses Buches – so umfangreich geworden, dass ich (weiter oben habe ich es bereits erwähnt) die Beispielprojekte vollständig überarbeitet habe. Fast alle Beispiele zeigen nun die Bare-Metal-Version (als Standard), die durch einfaches Entfernen der Kommentarzeichen vor dem Eintrag `#define MCAL` in die MCAL-Version umgesetzt werden können. Im Gegensatz zur Bare-Metal-Version, die sich immer auf eine bestimmte Komponente bezieht (z.B. `GPIOA` oder Timer `TIM9`), ist die MCAL universell für sämtliche Komponenten einsetzbar.

Den Quelltext der Bibliothek stelle ich Ihnen unter `https://gitlab.com/rjesse/mcal-stm.git` kostenlos in Form freier Software zur Verfügung. Die Namen der entsprechenden Dateien beginnen immer mit `mcal`. Die GPIO- oder Timer-Funktionen finden Sie entsprechend in den Dateien `mcalGPIO.h`/`mcalGPIO.c` oder `mcalTimer.h`/`mcalTimer.c`. Die Einführung neuer MCAL-Funktionen wird im Anschluss an jedes Beispiel besonders erwähnt. Die Funktionen selbst beginnen stets mit dem Komponentennamen, also z.B. `gpioSetPin()` oder `gpioSelectMode()`.

Hinweis

Sehr viele Softwareentwickler entwickeln mit zunehmender Erfahrung ihre eigenen Bibliotheken, damit sie »das Rad nicht immer aufs Neue erfinden müssen«. Die Entwicklung der MCAL erfolgt hier auch mit dem Wunsch, dass sie von den meisten Lesern dieses Buches genutzt und weiterentwickelt wird. Lassen Sie mich bitte an Ihren Erkenntnissen teilhaben unter `embedded@ralf-jesse.de`. Anders als in den gedruckten Listings erfolgt die Kommentierung der MCAL in englischer Sprache: Vielleicht wird die MCAL genau aus diesem Grund auch von internationalen Programmierern eingesetzt, die der deutschen Sprache nicht mächtig sind. Die Dokumentation wird unter Einsatz des Open-Source-Tools Doxygen generiert.

Entwicklungsumgebungen

Für die Entwicklung der Beispielprojekte habe ich die kostenlose und auf Eclipse basierende *STM32CubeIDE* verwendet, die nach einer Registrierung von der Webseite `https://st.com` heruntergeladen werden kann. Auf eine Bedienungsanleitung zu dieser *Entwicklungsumgebung* (*IDE, Integrated Development Environment*) habe ich aber verzichtet, da Tutorials zu Eclipse bzw. auf Eclipse basierenden Entwicklungsumgebungen in großer Zahl im Internet zu finden sind (teilweise auch in deutscher Sprache). Vielleicht bevorzugen Sie aber auch eine andere Entwicklungsumgebung, wie z.B. *IAR Workbench*, *Keil µVision* oder *Embedded Studio* von der deutschen Firma Segger, bei denen es sich aber um sehr teure (je nach Ausstattung kosten sie mehrere Tausend Euro) kommerzielle Entwicklungsumgebungen handelt.

Hinweis

Damit die verschiedenen Beispiele in diesem Buch auf einer gemeinsamen Basis aufsetzen können, habe ich in Kapitel 2 eine ausführliche Anleitung erstellt, die zeigt, wie Sie die CMSIS-Bibliothek für den STM32F446xx selbst erstellen können. Diese Arbeit ist nur für Nutzer einer Eclipse-basierten Entwicklungsumgebung (IDE) geeignet: Anwender einer der genannten kommerziellen IDEs benötigen sie nicht (sie können aber sicherlich auch etwas daraus lernen).

Tipp

Keil µVision, die IAR Workbench oder auch die Entwicklungsumgebung von Segger werden als Evaluierungsversionen kostenlos angeboten: Dann müssen Sie allerdings üblicherweise Einschränkungen akzeptieren, etwa dass die entwickelte Software nicht kommerziell genutzt werden darf. Auch die Größe der Softwareprojekte ist möglicherweise beschränkt.

Hinweis zu den Prozessorregistern

Ich habe lange überlegt, ob ich die Register in den Beispielprogrammen in diesem Buch vollständig beschreiben soll. Dieser Ansatz wäre sehr »seitenfüllend« geworden. Ich habe mich schließlich dazu entschlossen, nur die Bits der Register zu beschreiben, die zum Verständnis des jeweiligen Beispiels notwendig sind. Für diese Entscheidung habe ich mehrere Gründe:

- Der Umfang des Buches fällt durch diesen Ansatz geringer aus, was sich schließlich auch im Kaufpreis widerspiegelt.

- Beim Abschreiben des Referenzhandbuchs hätten sich womöglich Fehler eingeschlichen.
- Ein Aspekt beim Schreiben dieses Buches war, dass ich Sie zum Umgang mit der Originaldokumentation von STM motivieren möchte.
- Sie können sich leichter auf die wesentlichen Dinge in den einzelnen Beispielen konzentrieren. Mit ein wenig Übung können Sie sich bei eigenen Projekten die entsprechenden Funktionen selbst herleiten.

Support

Dies ist nicht das erste Buch, das ich geschrieben habe. Bereits für andere Bücher habe ich unter `https://www.ralf-jesse.de` eine Webseite angelegt, von der Sie die Beispielprogramme herunterladen können. Darüber hinaus bin ich für meine Leser unter der E-Mail-Adresse `embedded@ralf-jesse.de` für einen allgemeinen Austausch sowie für Fragen, Hilfestellungen und Anregungen erreichbar. Über die Jahre hat sich hier bereits eine stattliche Anzahl von Kontakten ergeben. Mein Versprechen an Sie: Jede E-Mail wird von mir zeitnah und persönlich beantwortet!

Anmerkungen des Autors

Seit der Veröffentlichung der ersten Auflage dieses Buches habe ich die MCAL-Bibliothek erheblich erweitert. Dies wirkt sich unmittelbar auf die enthaltenen Beispielprojekte aus: So habe ich seinerzeit für die meisten Beispiele noch Hilfsfunktionen entwickelt, die erst in einem weiteren Schritt in die MCAL-Bibliothek aufgenommen wurden. Außer zur Erläuterung von Konzepten werden viele dieser Hilfsfunktionen gar nicht mehr benötigt! Stattdessen habe ich die meisten Beispiele so umgestaltet, dass Sie beide Versionen, also die MCAL- **und** die »reine« Bare-Metal-Version enthalten. Durch das Hinzufügen bzw. Löschen der Kommentarzeichen `//` vor `#define MCAL` können Sie nun zwischen beiden Versionen hin- und herschalten.

Hinweis

Diese Vorgehensweise ließ sich leider nicht in allen Beispielen konsequent umsetzen! Immer dann, wenn es um die Einführung neuer Konzepte geht – dies werden Sie besonders deutlich z.B. in Kapitel 9 erfahren (hier wird der SysTick-Timer beschrieben) –, ist die Verwendung von Funktionen der MCAL-Bibliothek nicht sinnvoll, da sie den letzten von teilweise mehreren Entwicklungsschritten darstellen. Die Hilfsfunktionen dienen der Erläuterung der Konzepte, und die Verwendung der endgültigen Version einer MCAL-Funktion würde das Verständnis der Verbesserungen bzw. Zwischenschritte erschweren.

An dieser Stelle bleibt mir nur noch, Ihnen viel Spaß und Erfolg beim Durcharbeiten dieses Buches zu wünschen.

Zum Zeitpunkt der Veröffentlichung dieses Buches werden Sie auf meiner Webseite `https://www.ralf-jesse.de` auch die Beispielprojekte zu diesem Buch herunterladen können. Hier werde ich aber nach und nach auch weitere Projekte veröffentlichen. Geplant sind z.B. die Ansteuerung von Grafikdisplays und später auch die Implementierung einer Ethernet-Anbindung (dann allerdings mit einem anderen STM32F4- bzw. STM32F7-Controller, da der STM32F446 eine solche Schnittstelle nicht enthält). Auch die MCAL-Bibliothek wird weitergepflegt und immer wieder erweitert. Ich weise aber ausdrücklich darauf hin, dass die aktuelle Version der MCAL-STM **nicht mehr** auf meiner Webseite zu finden ist: Alternativ habe ich unter `https://gitlab.com/rjesse/mcal-stm.git` ein Gitlab-Repository angelegt. Nutzen Sie dieses Repository, wenn Sie die jeweils neueste Version der MCAL-STM benötigen. Es lässt sich nicht immer vermeiden, dass ich durch neue Erkenntnisse Namen von Funktionen ändern werde. Dann entfallen die veralteten Funktionen aber nicht: Vielmehr werden sie in der HTML-Dokumentation dann als »deprecated« (veraltet) bezeichnet. Sie finden dann immer einen Hinweis auf die Funktion, die Sie anstelle der alten Version nutzen sollten.

Teil I

Grundlagen

In diesem Teil:

- **Kapitel 1**
  STM32F4xx-Mikrocontroller 29
- **Kapitel 2**
  CMSIS und MCAL erstellen, der Bootprozess, Anwendung der CMSIS 43
- **Kapitel 3**
  Embedded C vs. Standard C 77
- **Kapitel 4**
  RCC, SYSCFG und SCB 85
- **Kapitel 5**
  Einstellung von Taktfrequenzen 95

Kapitel 1

STM32F4xx-Mikrocontroller

Will man Software für einen Mikrocontroller entwickeln, ist es hilfreich, wenn grundlegende Kenntnisse zur Hardware vorhanden sind. Die erforderlichen Kenntnisse müssen nicht so tief gehend sein, wie dies für Hardwareentwickler erforderlich ist, man sollte aber in der Lage sein, sich mit ihnen auf einer gemeinsamen Basis verständigen zu können.

Daher beginne ich dieses Buch mit einem kurzen Überblick über die Mitglieder der STM32F4xx-Familie.

1.1 Überblick über die STM32F4xx-Familie

Wie bereits in der Einleitung erwähnt, besteht die STM32F4xx-Familie aus einer Vielzahl von Mikrocontrollern, die, abhängig von ihrer Ausstattung und ihrer Leistungsfähigkeit, in verschiedene *Produktlinien* eingeordnet sind:

- Die *Advanced Line* umfasst die Ausführungen STM32F469, STM32F429 und STM32F427. Diese bieten die größte Funktionsvielfalt und enthalten unter anderem Ethernet-Interfaces. Bis auf den STM32F427 enthalten die Mitglieder der Advanced Line auch einen TFT-/LCD-Controller. Die genannten Mikrocontroller können mit einer Taktfrequenz von bis zu 180 MHz eingesetzt werden. Je nach Modell verfügen sie über einen Flashspeicher von bis zu 2 MByte und bis zu 384 KByte SRAM.
- Die sogenannte *Foundation Line* umfasst die Typen STM32F446, STM32F407 und STM32F405. Die maximale Taktfrequenz des STM32F405 beträgt 168 MHz, die beiden anderen können – wie die Mitglieder der Advanced Line – mit einer Frequenz von bis zu 180 MHz getaktet werden.
- Die *Access Line* umfasst mit den Mikrocontrollern STM32F401, STM32F410, STM32F411, STM32F412 und STM32F413 die Familienmitglieder, die am wenigsten leistungsstark sind. Dies betrifft weniger die Größe von Flashspeicher (maximal 1.536 KByte) und SRAM (bis zu 320 KByte), sondern vielmehr die Zahl serieller Schnittstellen und anderer Peripheriekomponenten. Die Taktfrequenz ist mit maximal 100 MHz aber immer niedriger verglichen mit den Mitgliedern der Advanced Line bzw. der Foundation Line.

Abbildung 1.1 zeigt die derzeit verfügbaren Mikrocontroller dieser Familie (Stand: März 2022). Der in diesem Kapitel exemplarisch beschriebene Mikrocontroller STM32F446 ist durch einen Rahmen hervorgehoben.

	Produktlinie	FCPU (MHz)	Flash (kBytes)	RAM (KB)	Ethernet-I/F* IEEE 1588	2x CAN-I/F*	Kamera-I/F*	SDRAM-I/F*	Dual-QUAD SPI	SAI	SPDIF Rx	CHROM-ART Graphic Acc. ™	TFT-/LCD-Controller	MIPI DSI
						Advanced Lines								
• ART Accelerator TM • SDIO • USART, SPI, I²C • I²S + Audio-PLL • 16- und 32-Bit Timer • 12-Bit ADC (0,41 µs)	STM32F469	180	512k bis 2056k	384	✓	✓	✓	✓	✓	✓		✓	✓	✓
	STM32F429	180	512k bis 2056k	256	✓	✓	✓	✓		✓		✓	✓	
	STM32F427	180	1024k bis 1024k	256	✓	✓	✓	✓		✓		✓		
						Connectivity Lines								
	STM32F446	180	256k bis 512k	128		✓	✓	✓	✓	✓	✓			
	STM32F407	168	512k bis 1024k	192	✓	✓	✓							
	STM32F405	168	512k bis 1024k	192		✓								
						Access Lines								
• True Number Random Generator • Batch Acquisition Mode • Low Voltage (1,7 - 3,6V) • Temperaturei: -40°C - 125°C	Produktlinie	FCPU (MHz)	Flash (kBytes)	RAM (KB)	Stromaufn. µA/MHz (Min.)	Stromaufn. (Stopp) (Min.)	Abm. (min.) in mm	FSMC (NOR/PSRAM /)	QSPI	DFSDM	DAC	TRNG	DMA Stapel-verarbeitung	USB 2.0 OTG FS
	STM32F401	84	128k bis 512k	bis zu 96	128µA	10µA	3x3							✓
	STM32F410	100	64k bis 128k	32	89µA	6µA	2,553 x 2,579				✓	✓	BAM	-
	STM32F411	100	256k bis 512k	128	100µA	12µA	3,034 x 3,22						BAM	✓
	STM32F412	100	512k bis 1024k	256	112µA	18µA	3,653 x 3,651	✓	✓	✓		✓	BAM	✓
	STM32F413	100	1024k bis 1536k	320	115µA	18µA	3,951 x 4,039	✓	✓	✓	✓	✓	BAM	✓

*I/F = Interface (Schnittstelle)

Abb. 1.1: Überblick über die STM32F4-Familie

Hinweis

Obwohl ich mich nachfolgend auf den STM32F446 konzentriere, lässt sich seine Beschreibung nicht nur auf die anderen Mitglieder der STM32F4-Familie übertragen, sondern auch – mit Ausnahme der Arm-spezifischen Bestandteile sowie der herstellerspezifischen Peripheriekomponenten – auf beliebige andere Mikrocontroller. Natürlich gibt es Unterschiede bei der Programmierung der integrierten Peripheriekomponenten, was sich nicht zuletzt in unterschiedlichen Benennungen und Adressen der Register wie auch in der Anschlussbelegung zeigt, das Funktionsprinzip ist aber überall sehr ähnlich. Die Portierung der Kenntnisse auf Mikrocontroller anderer Hersteller sollte recht einfach sein, da Sie hier ja die Programmierung von der Pike auf lernen.

1.2 Der STM32F446

Der STM32F446 ist in verschiedenen Ausführungsvarianten erhältlich. Im folgenden Abschnitt werden diese näher beschrieben.

1.2.1 Varianten des STM32F446

Tabelle 1.1 zeigt die Ausführungen und ihre Unterschiede:

Peripheriekomponenten		STM32F446							
		MC	ME	RC	RE	VC	VE	ZC	ZE
Flash in KByte		256	512	256	512	256	512	256	512
SRAM in KByte	System	128 (112 + 16)							
	Backup	4							
Controller für flexiblen Speicher (FMC)		Nein				Ja			
Timer	GP	10							
	Advanced	2							
	Basic	2							
Kommunikations-Interfaces	SPI/I²S	4/3 (simplex)							
	I²C	4/1 davon als FMP+ (Fast mode Plus)							
	UART/USART	2/4							
	USB OTG FS	Ja, 6 Endpunkte							
	USB OTG HS	Ja, 8 Endpunkte							
	CAN	2							
	SAI	2							
	SDIO	Ja							
	SPDIF-Rx	1							
	HDMI-CEC	1							
	Quad SPI	1							

Tabelle 1.1: Ausstattung des STM32F446

Peripheriekomponenten	STM32F446							
	MC	ME	RC	RE	VC	VE	ZC	ZE
Kamera-Interface	Ja							
GPIOs	63		50		81		114	
12-Bit-ADC (Anzahl/Kanäle)	3/14		3/16		3/16		3/24	
12-Bit-DAC	Ja, 2 Kanäle							
Maximale Taktfrequenz	180 MHz							
Betriebsspannung	1,8 bis 3,6 V							
Gehäuse und Pins	WLCSP81		LQFP64		LQFP100		LQFP144U FBGA144	

Tabelle 1.1: Ausstattung des STM32F446 (Forts.)

Wie Sie Tabelle 1.1 entnehmen können, sind allein vom STM32F446 acht Ausführungen erhältlich, die durch nachgestellte Buchstaben und Ziffern unterschieden werden. Abbildung 1.2 zeigt, wie die Bezeichnungen bei STM interpretiert werden.

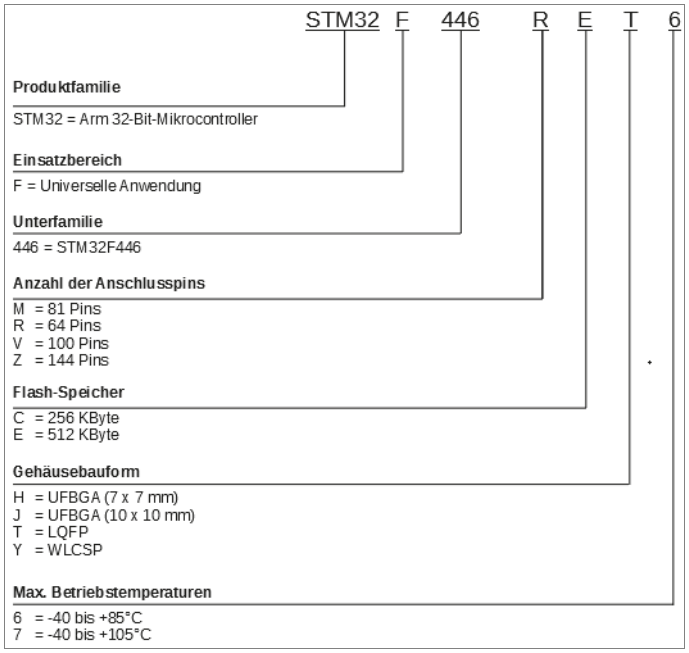


Abb. 1.2: Produktbezeichnungen der STM32F4xx-Mikrocontroller

Hinweis

Möchten Sie einen anderen Mikrocontroller aus der STM32F4xx-Familie verwenden, finden Sie die entsprechenden Tabellen auf der Webseite `https://www.st.com/en/microcontrollers-microprocessors/stm32f4-series.html` im Menü RESOURCES.

1.2.2 Speicherbelegung/Memory-Mapping

Der adressierbare Speicher ist in acht Blöcke mit einer Größe von jeweils 512 MByte unterteilt, es können somit maximal $2^{32} - 1$ Byte = 4.294.967.296 Byte adressiert werden. Dieser Wert stellt aber nur das theoretische Maximum dar, tatsächlich verfügt keiner der genannten Mikrocontroller über derart viel Speicher, sodass die höchste adressierbare Speicherstelle deutlich kleiner ist.

Abbildung 1.3 zeigt eine grobe Unterteilung dieser Blöcke und der hier verfügbaren Funktionen. Im rechten Teil von Abbildung 1.3 sehen Sie einige Abkürzungen, wie z.B. AHB1/2/3 und APB1/2: Was es damit auf sich hat, wird weiter unten in Abschnitt 1.2.4 erläutert.

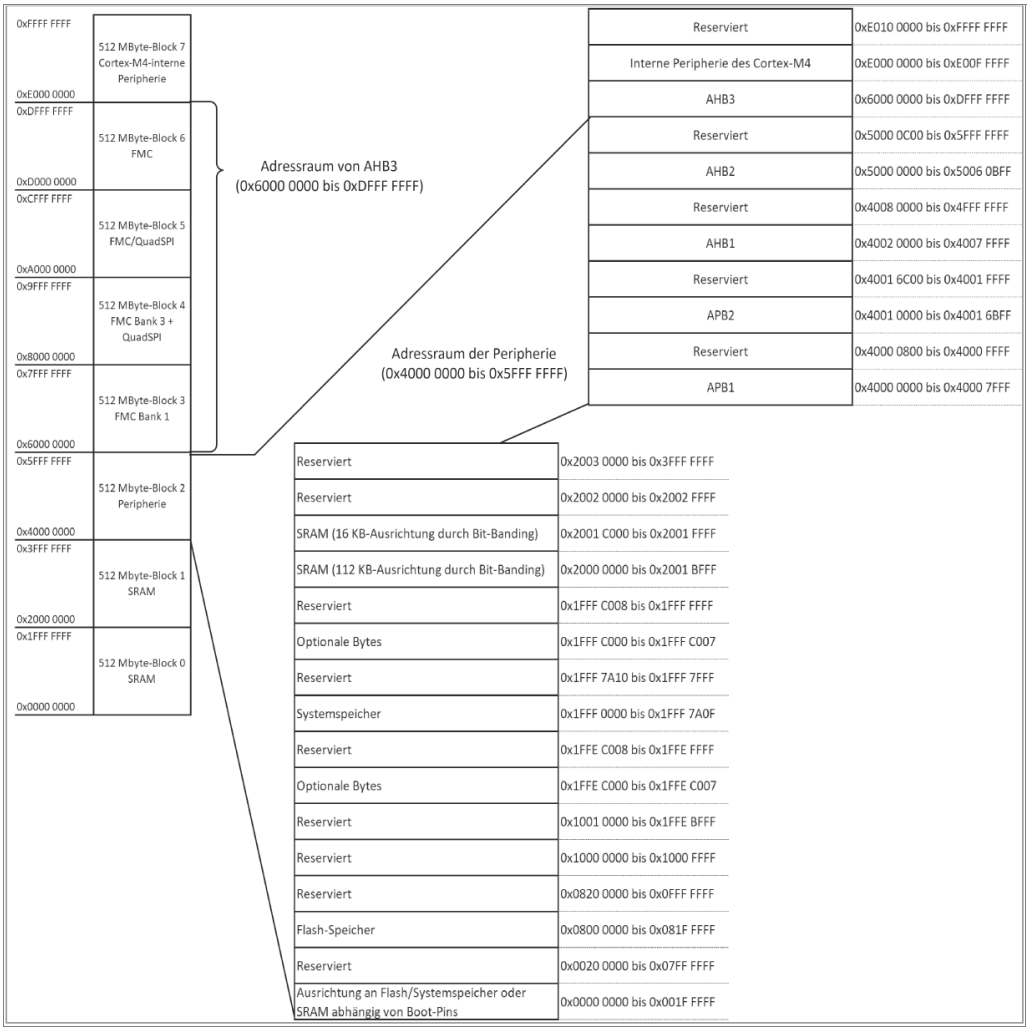


Abb. 1.3: Speicherbereiche (Memory-Mapping), aus dem Datenblatt entnommen

Die folgenden Tabellen zeigen die verfügbaren Peripheriekomponenten, den jeweils genutzten Bus, über den sie angesprochen werden, sowie die Adressen, die den Speicherbereichen und Registern zugeordnet sind.

Wichtig

Zwei Adressbereiche möchte ich vorab besonders hervorheben: Sie sind zwar in Abbildung 1.3 angegeben, fallen dort aber nicht besonders deutlich auf: Es handelt sich um den Flashspeicher (ab Adresse 0x0800 0000) und das SRAM (ab Adresse 0x2000 0000). Die Größe des Speichers variiert bei den verschiedenen Mikrocontrollern.

Hinweis

Die Adressbereiche in den folgenden Tabellen sind nach den jeweiligen Bussen (Cortex-M4, AHB und APB) sortiert. Dies führt dazu, dass die Blockgrenzen in Abbildung 1.2 quasi »ungültig« werden. Darüber hinaus sind große Adressbereiche im Speicherraum reserviert, sodass sie nicht genutzt werden können. Reservierte Speicherbereiche wurden in den folgenden Tabellen nicht aufgeführt.

Bus	Adressgrenzbereiche	Komponente(n)
Cortex-M4-interner Bus	0xE000 0000–0xE00F FFFF	Cortex-M4-interne Komponenten (z.B. CPU-Kern, NVIC, Floating-Point-Unit (FPU) usw.)

Tabelle 1.2: Adressbereich von Cortex-M4-internen Komponenten

Bus	Adressbereiche	Komponente(n)
AHB3	0xD000 0000–0xDFFF FFFF	FMC Bank 6
	0xC000 0000–0xCFFF FFFF	FMC Bank 5
	0xA000 1000–0xA000 1FFF	QuadSPI Control Register
	0xA000 0000–0xA000 0FFF	FMC Control Register
	0x9000 0000–0x9FFF FFFF	QuadSPI
	0x8000 0000–0x8FFF FFFF	FMC Bank 3
	0x6000 0000–0x6FFF FFFF	FMC Bank 1

Tabelle 1.3: Komponentenadressen, die über den AHB3-Bus angesprochen werden

Bus	Adressbereiche	Komponente(n)
AHB2	0x5005 0000–0x5005 03FF	DCMI (Digital Camera Interface)
	0x5000 0000–0x5003 FFFF	USB OTG FS (USB on-the-go, Full Speed)

Tabelle 1.4: Komponentenadressen, die über den AHB2-Bus angesprochen werden

Bus	Adressbereiche	Komponente(n)
AHB1	0x4004 0000–0x4007 FFFF	USB OTG HS (USB on-the-go, High Speed)
	0x4002 6400–0x4002 67FF	DMA2 (Direct Memory Access)
	0x4002 6000–0x4002 63FF	DMA1
	0x4002 4000–0x4002 4FFF	Backup-SRAM
	0x4002 3C00–0x4002 3FFF	Flashspeicher-Schnittstelle
	0x4002 3800–0x4002 3BFF	RCC (Reset and Clock Control)
	0x4002 3000–0x4002 33FF	CRC (Cyclic Redundancy Check)
	0x4002 1C00–0x4002 1FFF	GPIOH
	0x4002 1800–0x4002 1BFF	GPIOG
	0x4002 1400–0x4002 17FF	GPIOF
	0x4002 1000–0x4002 13FF	GPIOE
	0X4002 0C00–0x4002 0FFF	GPIOD
	0x4002 0800–0x4002 0BFF	GPIOC
	0x4002 0400–0x4002 07FF	GPIOB
	0x4002 0000–0x4002 03FF	GPIOA

Tabelle 1.5: Komponentenadressen, die über den AHB1-Bus angesprochen werden

Bus	Adressbereiche	Komponente(n)
APB2	0x4001 5C00–0x4001 5FFF	SAI1 (Serial Audio Interface)
	0x4001 5800–0x4001 5BFF	SAI2
	0x4001 4800–0x4001 4BFF	Timer TIM11
	0x4001 4400–0x4001 47FF	Timer TIM10
	0x4001 4000–0x4001 43FF	Timer TIM9
	0x4001 3C00–0x4001 3FFF	EXTI (External Interrupt Controller)
	0x4001 3800–0x4001 3BFF	SYSCFG (System Configuration)
	0x4001 3400–0x4001 37FF	SPI4 (Serial Peripheral Interface)
	0x4001 3000–0x4001 33FF	SPI1
	0x4001 2C00–0x4001 2FFF	SDIO (Secure Digital Input/Output Interface)
	0x4001 2000–0x4001 23FF	ADC1/2/3 (Analog-to-Digital Converter)
	0x4001 1400–0x4001 17FF	USART6 (Universal Synchronous/ Asynchronous Receiver/Transmitter)
	0x4001 1000–0x4001 13FF	USART1
	0x4001 0400–0x4001 07FF	Timer TIM8
	0x4001 0000–0x4001 03FF	Timer TIM1

Tabelle 1.6: Komponentenadressen, die über den APB2-Bus angesprochen werden

Bus	Adressbereiche	Komponente(n)
	0x4001 5C00–0x4001 5FFF	SAI1 (Serial Audio Interface)
	0x4001 5800–0x4001 5BFF	SAI2
	0x4001 4800–0x4001 4BFF	Timer TIM11
	0x4000 7400–0x4000 77FF	DAC (Digital-to-Analog Converter)
	0x4001 4400–0x4001 47FF	Timer TIM10
	0x4001 4000–0x4001 43FF	Timer TIM9
	0x4001 3C00–0x4001 3FFF	EXTI (External Interrupt Controller)
	0x4001 3800–0x4001 3BFF	SYSCFG (System Configuration)
	0x4001 3400–0x4001 37FF	SPI4 (Serial Peripheral Interface)
	0x4001 3000–0x4001 33FF	SPI1
	0x4001 2C00–0x4001 2FFF	SDIO (Secure Digital Input/Output Interface)
	0x4001 2000–0x4001 23FF	ADC1/2/3 (Analog-to-Digital Converter)
	0x4001 1400–0x4001 17FF	USART6 (Universal Synchronous/ Asynchronous Receiver/Transmitter)
	0x4001 1000–0x4001 13FF	USART1
	0x4001 0400–0x4001 07FF	Timer TIM8
	0x4001 0000–0x4001 03FF	Timer TIM1
APB1		
	0x4000 7000–0x4000 73FF	PWR (Power Control)
	0x4000 6C00–0x4000 6FFF	HDMI-CEC (High Definition Multimedia Interface, Consumer Electronics Control)
	0x4000 6800–0x4000 6BFF	CAN2 (Controller Area Network)
	0x4000 6400–0x4000 67FF	CAN1
	0x4000 6000–0x4000 63FF	FMPI2C1 (Fast Mode Plus Inter-integrated Circuit)
	0x4000 5C00–0x4000 5FFF	I2C3 (Inter-integrated Circuit)
	0x4000 5800–0x4000 5BFF	I2C2
	0x4000 5400–0x4000 57FF	I2C1
	0x4000 5000–0x4000 53FF	UART5 (Universal Asynchronous Receiver/Transmitter)
	0x4000 4C00–0x4000 4FFF	UART4
	0x4000 4800–0x4000 4BFF	USART3
	0x4000 4400–0x4000 47FF	USART2
	0x4000 4000–0x4000 43FF	SPDIFRX (Sony-Philips Digital Interface Receiver/Transmitter)

Tabelle 1.7: Komponentenadressen, die über den APB1-Bus angesprochen werden

Bus	Adressbereiche	Komponente(n)
	0x4000 3C00–0x4000 3FFF	SPI3/I2S3 (Serial Peripheral Interface/Inter-integrated Sound Interface)
	0x4000 3800–0x4000 3BFF	SPI2/I2S2
	0x4000 3000–0x4000 33FF	IWDG (Independent Watchdog)
	0x4000 2C00–0x4000 2FFF	WWDG (Window Watchdog)
	0x4000 2800–0x4000 2BFF	RTC- & BKP-Register (Real Time Clock, Backup-Register)
	0x4000 2000–0x4000 23FF	Timer TIM14
	0x4000 1C00–0x4000 1FFF	Timer TIM13
	0x4000 1800–0x4000 1BFF	Timer TIM12
	0x4000 1400–0x4000 17FF	Timer TIM7
	0x4000 1000–0x4000 13FF	Timer TIM6
	0x4000 0C00–0x4000 0FFF	Timer TIM5
	0x4000 0800–0x4000 0BFF	Timer TIM4
	0x4000 0400–0x4000 07FF	Timer TIM3
	0x4000 0000–0x4000 03FF	Timer TIM2

Tabelle 1.7: Komponentenadressen, die über den APB1-Bus angesprochen werden (Forts.)

1.2.3 Interner Aufbau des STM32F446

An dieser Stelle würde ich Ihnen gern das Blockschaltbild des STM32F446 zeigen. Damit es auf eine Buchseite passt, müsste ich es jedoch dermaßen verkleinern, dass Sie die Details nicht mehr erkennen könnten. Unter dem Link `https://www.st.com/resource/en/datasheet/stm32f446mc.pdf` finden Sie auf Seite 16 das vollständige Blockschaltbild dieses Mikrocontrollers.

Hinweis

Die anderen Mitglieder der STM32F4xx-Familie haben einen ähnlichen Aufbau: Die wesentlichen Unterschiede bestehen in der Anzahl der integrierten Peripheriekomponenten, der Taktfrequenz sowie der Ausstattung mit Flashspeicher und SRAM. Wichtig ist hier, dass die Komponenten, die in allen bzw. in mehreren Mikrocontrollern enthalten sind, überall gleich benannt sind und ihre Programmierung im Wesentlichen identisch ist. Es kann vereinzelt sein, dass bestimmte Komponenten etwas leistungsstärker sind als in den einfacheren Varianten. Dies äußert sich darin, dass manchmal Bits, die in Registern als reserviert markiert sind, in den besseren Varianten zusätzlich verwendet werden.

Der Cortex-M4-Kern

Das »Herz« der Prozessoren der STM32F4xx-Familie ist in der linken oberen Ecke des Blockschaltbilds zu sehen. Abbildung 1.4 zeigt es als Auszug aus dem oben erwähnten Blockschaltbild.

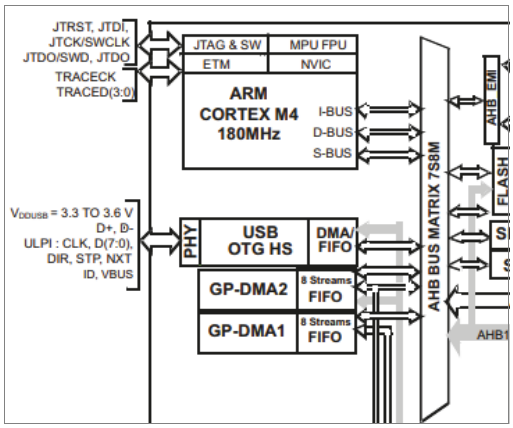


Abb. 1.4: Das »Herz« der STM32F4xx-Familie

Neben der *Central Processing Unit* (*CPU*) – hier als ARM CORTEX M4 180 MHz bezeichnet – verfügen alle Cortex-M-Mikrocontroller bereits über eine Vorrichtung zur Fehlersuche, die als *Embedded Trace Macrocell* (*ETM*) bezeichnet ist. Die ETM kommuniziert mit der Außenwelt über einen traditionellen *JTAG*-Port (*Joint Test Action Group*) sowie über eine einfache 2-Draht-Debug-Schnittstelle (*SWD, Serial Wire Debug*), die besonders platzsparend ist.

Ihre besondere Leistungsfähigkeit erreichen die Cortex-M-Mikrocontroller aller Hersteller durch den sogenannten *NVIC* (*Nested Vector Interrupt Controller*).

Hinweis

Auf die Bedeutung und die Aufgaben des NVIC gehe ich in Kapitel 7 »Interrupts« näher ein.

Optional können die Cortex-M-Mikrocontroller eine *Fließpunkteinheit* (*FPU, Floating Point Unit*) enthalten

Hinweis

Sämtliche Mikrocontroller der STM32F4xx-Familie sind mit einer FPU ausgestattet. Dies gilt nicht zwangsläufig für Cortex-M4-Controller anderer Hersteller!

Obwohl in Abbildung 1.4 nicht eingezeichnet, enthält der Cortex-M4-Kern noch weitere Komponenten. Hierbei handelt es sich um:

- *SCB* (*System Control Block*) – Dieser stellt die Schnittstelle des Programmierers zum Prozessor dar. Wie der Name bereits andeutet, dient er der Steuerung des Systems inklusive seiner Konfiguration. Darüber werden hier die sogenannten System-Exceptions verwaltet: Dies sind besondere Situationen, die üblicherweise nicht abgefangen werden können, wie beispielsweise eine Division durch null.
- *System Timer* – Jeder Cortex-M-Kern von Arm enthält einen solchen Timer, der allgemein als *SysTick-Timer* bezeichnet wird. Beim Einsatz von Betriebssystemen wie z.B. RTOS wird er zur Steuerung des Scheduler verwendet, der den Wechsel von einem Task zu einem anderen steuert. Wird, wie bei der sogenannten Bare-Metal-Programmierung, kein Betriebssystem eingesetzt, kann dieser Timer auch anders genutzt werden. Üblicherweise wird er so konfiguriert, dass er in einem 10-Millisekunden-Takt »tickt«. Andere Taktzeiten (auch kürzere) sind aber ebenfalls möglich: In den SysTick-Beispielen tickt der Timer im 1-ms-Takt.
- *MPU* (*Memory Protection Unit*) – Bestimmte Speicherregionen sollen vor einem Zugriff (Änderung) von außen – auch vor Programmierern – geschützt sein. Ein typisches Beispiel hierfür ist ein vom Hersteller implementierter Bootloader, der den Mikrocontroller auch dann noch »wiederbeleben« soll, wenn der Softwareentwickler durch grobe Fehler keinen Zugriff mehr auf ihn hat. Die MPU sorgt somit für eine höhere Gesamtsicherheit und Zuverlässigkeit des Systems.

Weitere Komponenten der STM32F4xx-Familie

Hier beginnen die Unterschiede zu anderen Mitgliedern dieser Familie und besonders zu Cortex-M4-Mikrocontrollern anderer Hersteller. Innerhalb der STM32F4xx-Familie bedeutet »Unterschiede« aber nicht, dass ihre Mitglieder zueinander inkompatibel sind, es soll vielmehr ausgedrückt werden, dass je nach Ausstattung nicht alle Peripheriekomponenten gleichermaßen verfügbar sind.

- Für die Steuerung bzw. das Abfragen externer Geräte (Motoren, Sensoren, Ventile, Relais, LEDs, Displays usw.) werden die sogenannten *GPIOs* (*General Purpose Inputs/Outputs*) verwendet. Je nach Anzahl der Anschlusspins der Variante (siehe hierzu auch den Buchstaben-Ziffern-Code weiter oben in Abbildung 1.2) stehen beim STM32F446 mit den Ports GPIOA bis GPIOH maximal acht dieser Ports zur Verfügung. Von GPIOH sind zwei Anschlüsse herausgeführt, alle anderen GPIO-Ports verfügen über bis zu 16 digitale Ein- bzw. Ausgänge.

Hinweis

Es gibt auch Varianten, die mit Port I einen weiteren Port enthalten! Beim STM32F76x existieren darüber hinaus noch die GPIO-Ports J und K!

- Der STM32F446 enthält insgesamt 14 *Timer*, die unabhängig voneinander für verschiedene Aufgaben genutzt werden können. Mögliche Anwendungen sind einfache Zeitschaltungen, die Erzeugung von PWM-Signalen (Betriebsart: Output Compare) oder das Messen (Zählen) von Impulsen, die von außen zugeführt werden (Betriebsart: Input Capture). Neben den »normalen« Timern gibt es mit der *RTC* (*Real Time Clock*) zusätzlich noch zwei *Watchdog-Timer*.

Hinweis

Auf die Details der Timer-Programmierung wird in den Kapiteln 9 bis 12 eingegangen.

- Besonders reichhaltig sind die STM32F4xx-Controller mit seriellen Schnittstellen für die Kommunikation mit der Außenwelt ausgestattet. Je nach Modell enthalten sie bis zu 20 serielle Schnittstellen (UART, USART, SPI, CAN, I^2C, I^2S usw.).
- Die Natur ist nicht digital, sondern analog. Dies bedeutet, dass sich physikalische Größen nicht sprunghaft, sondern stetig ändern. Mikrocontroller arbeiten intern aber rein digital. Mit den integrierten *DAC* (*Digital-to-Analog Converter*) und *ADC* (*Analog-to-Digital Converter*) ist es möglich, analoge Größen in Form einer elektrischen Spannung zu erzeugen (DAC) oder zu erfassen (ADC) und durch die digitale Software zu verarbeiten.
- Sollen große Datenmengen zwischen Speicher und Peripherie ausgetauscht werden (denken Sie hier beispielsweise an die Übertragung von Daten zwischen Speicher und einer SD-Karte), ist es ungünstig, dies über den CPU-Kern auszuführen: Er müsste die Daten zunächst lesen, anschließend verarbeiten und schließlich zum Zielgerät transportieren. Dieser Vorgang, den man allgemein als *Read-Modify-Write* bezeichnet, ist aus Sicht des Prozessorkerns sehr zeitaufwendig. Darüber hinaus kann es passieren, dass Interrupts während des Read-Modify-Write Daten verändern. Die Zeit, in der der Kern die Daten »hin und her schubst«, kann von der CPU besser genutzt werden, wenn es denn eine andere Lösung für diese Aufgabe gibt. Tatsächlich existiert eine solche Lösung, indem man *DMA-Controller* (*Direct Memory Access*) einsetzt, die diese Aufgabe übernehmen.

Hinweis

Der STM32F446 enthält sogar zwei DMA-Controller!

- Die Mitglieder der Advanced Line der STM32F4xx-Familie verfügen zudem noch über *Ethernet*-Schnittstellen sowie *TFT-/LCD-Interfaces* zum direkten Anschluss von Displays.

Kommandos und Daten werden zwischen dem CPU-Kern und der Peripherie über spezielle Leitungssysteme, die als Bus bezeichnet werden, transportiert. Die verschiedenen Bussysteme werden im nächsten Abschnitt vorgestellt.

1.2.4 Bussysteme des STM32F446

In diesem Abschnitt werden die verschiedenen Busse vorgestellt, die bereits in den Tabellen 1.2 bis 1.7 erwähnt wurden. Busse sind interne Verbindungen, über die der Prozessorkern mit dem Flashspeicher, dem SRAM sowie mit den Peripheriekomponenten Kommandos und Daten austauscht. Damit der Datenaustausch kollisionsfrei erfolgt, sind sämtliche Busse in einer Matrix miteinander verbunden.

Cortex-M4-interne Busse

Zunächst einmal soll gesagt werden, dass es **den** Cortex-M4-Bus in dieser Form gar nicht gibt! Stattdessen verbergen sich hinter dieser Bezeichnung mehrere Busse, die in ihrer Gesamtheit den Cortex-M4-Bus darstellen. Die hier genannten Busse sind unmittelbar an den Prozessorkern »angebunden«.

Hinweis

Werden in den folgenden Beschreibungen konkrete Geschwindigkeiten in Form von Taktfrequenzen genannt, so beziehen sie sich auf den STM32F446. Bei anderen Mikrocontrollern der Familie können die Werte abweichen.

- Der *I-Bus* (*Instruction Bus*) ist dafür zuständig, Instruktionen, die der Prozessorkern bearbeiten soll, aus dem Flashspeicher bzw. aus dem SRAM zu laden.
- Der *D-Bus* (*Data Bus*) ist für den Austausch von Nutzdaten zuständig. Darüber hinaus ermöglicht der D-Bus über einen Debugger für Diagnosezwecke auch den Zugriff auf Daten.
- Der *S-Bus* (*System Bus*) ermöglicht den Zugriff auf Daten, die entweder in den Peripheriekomponenten oder im SRAM gespeichert sind. Er kann auch dazu verwendet werden, Instruktionen zu transportieren, ist aber weniger effizient als der I-Bus.
- Ebenfalls an den CPU-Kern direkt angeschlossen sind spezielle Anschlüsse der *DMA*-Controller. Über diese Leitungen zeigen sie dem Prozessorkern an, dass sie die Datenleitungen zwischen Speicher↔Speicher- bzw. Speicher↔Peripherie exklusiv nutzen wollen.
- Das letzte Bussystem, das unmittelbar an der CPU-Seite der Matrix angeschlossen ist, wird für *USB OTG HS* (USB On-the-go, High Speed) genutzt.

D-, I- und S-Bus verbinden den CPU-Kern unmittelbar mit einer Seite der erwähnten Matrix. Hier sind ebenfalls die DMA-Controller und USB OTG HS angeschlossen.

Auf der anderen Seite der Matrix sind AHB1, AHB2 sowie Verbindungen zu Flashspeicher, SRAM und FMC bzw. QuadSPI angeschlossen.

- Für einen besonders schnellen Datenaustausch werden die beiden AHB-Busse *AHB1* und *AHB2* verwendet. AHB ist die Abkürzung von *Advanced High-Performance Bus*. AHB1 und AHB2 können Daten mit dem maximalen Systemtakt (also bis zu 180 MHz beim STM32F446) übertragen, sie werden daher für besonders schnelle Peripheriekomponenten (wie z.B. GPIOs, Cyclic Redundancy Check [CRC] oder Reset and Clock Control [RCC]), Flashspeicher und SRAM verwendet.
- Peripheriekomponenten, die nicht so schnell sind, sind hingegen mit *APB1* (*Advanced Peripheral Bus*) und *APB2* verbunden.

Um die AHB-Busse von der Datenflut zu entlasten, werden Daten über sogenannte *AHB-/APB-Brücken* (AHP/APB Bridges) übertragen. APB2 ist mit einer maximalen Taktrate von 90 MHz der schnellere Bus, APB1 wird mit maximal 45 MHz getaktet.

Hinweis

Auch die APB-Übertragungsraten in Höhe von 90 bzw. 45 MHz sind für die meisten Peripheriekomponenten noch viel zu schnell. Daher sind fast alle Peripheriekomponenten mit *Vorteilern* (*Prescaler*) ausgestattet, die den Peripherietakt weiter verringern. An welchen Bus die Peripheriekomponenten angeschlossen sind, können Sie dem Blockschaltbild (siehe den Link am Ende von Abschnitt 1.2.1) sehr einfach entnehmen.

Die Beschreibung des internen Aufbaus der STM32F4xx-Mikrocontroller soll hiermit abgeschlossen werden.