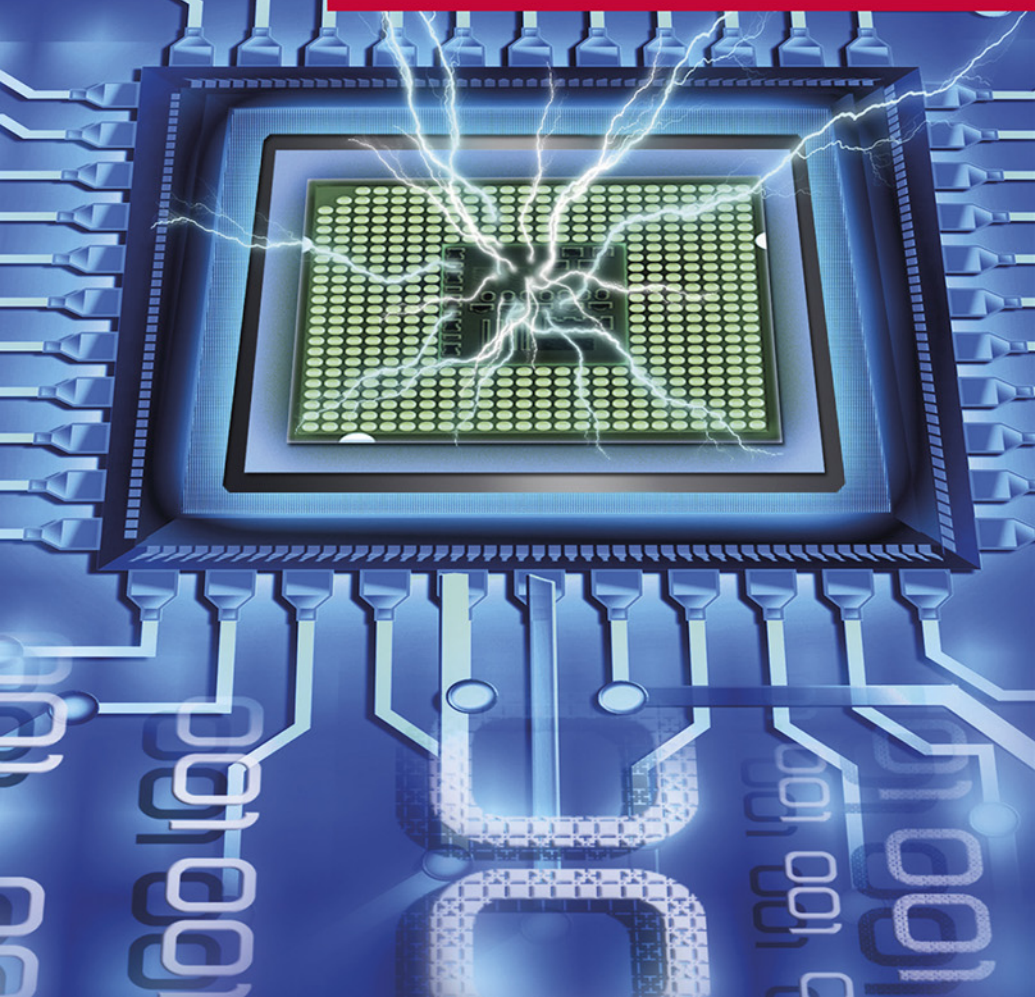


STM32

ARM-Mikrocontroller programmieren
für Embedded Systems

Das umfassende Praxisbuch



Inhaltsverzeichnis

Einleitung.....	11
Warum STM32F4?	11
Zielgruppe dieses Buchs.....	12
Voraussetzungen.....	13
Aufbau des Buchs	14
 Teil I Grundlagen	 23
 1 Die Hardware des STM32F4xx-Mikrocontrollers	 25
1.1 Überblick über die STM32F4xx-Familie	25
1.2 Der STM32F446	27
1.2.1 Varianten des STM32F446	27
1.2.2 Speicherbelegung/Memory-Mapping.....	29
1.2.3 Interner Aufbau des STM32F446.....	33
1.2.4 Bussysteme des STM32F446	36
 2 CMSIS und MCAL	 39
2.1 CMSIS-Bibliothek beschaffen und erstellen	39
2.2 Aufräumarbeiten	40
2.3 Erstellen der Bibliothek.....	43
2.4 Fertigstellen der CMSIS-Bibliothek.....	45
2.4.1 Build-Variable für das CMSIS-Verzeichnis	45
2.4.2 Einstellen von Suchpfaden	46
2.4.3 Abschlussarbeiten an der CMSIS-Bibliothek.....	47
2.4.4 Verwenden von CMSIS in eigenen Projekten	48
2.4.5 Bezeichnungstechniken in CMSIS von STM.....	51
2.5 Der Bootvorgang	53
2.5.1 Der Reset-Handler	54
2.5.2 SystemInit.....	56
2.5.3 Die Erzeugung des Taktsignals	57
2.5.4 Linkerscripts	59

2.6	Aufbau der Headerdatei(en)	62
2.6.1	Orientierung in stm32f446xx.h	63
2.7	MCAL	66
3	Embedded C vs. Standard C	71
4	RCC, SYSCFG und SCB	77
4.1	Reset and Clock Control (RCC)	77
4.1.1	Reset	77
4.1.2	Clock Control	78
4.1.3	Ein erstes einfaches Projekt	79
4.1.4	Weitere wichtige Aufgaben des RCC	81
4.2	System Configuration Controller (SYSCFG)	85
4.3	System Control Block (SCB)	86
 Teil II Kernkomponenten der STM32F4xx-Mikrocontroller		87
5	GPIO: General Purpose Input/Output.	91
5.1	Features und Grenzdaten	92
5.2	GPIO-Register	93
5.3	Zwei einfache Beispiele	102
5.3.1	Rechtecksignal mit dem ODR-Register	102
5.3.2	Rechtecksignal mit BSRR	103
6	Polling, Interrupts und Exceptions.	107
6.1	Allgemeines zu Polling und Interrupts	108
6.1.1	Polling	108
6.1.2	Interrupts	109
6.2	Interrupts beherrschen	110
6.2.1	Maskierbare und nicht-maskierbare Interrupts	110
6.2.2	Globale Interrupts	112
6.2.3	Der Nested Vector Interrupt Controller NVIC	112
6.3	Externe Interrupts	118
6.3.1	External Interrupt/Event Controller (EXTI)	118
6.3.2	Beispiel zu externen Interrupts	123
6.3.3	Erkennung mehrerer externer Interrupts	128
6.4	Exceptions	133

7	Alternative GPIO-Funktionen	135
7.1	Wiederholung	135
7.2	Aktivieren alternativer Funktionen	137
7.3	Auswahl alternativer Funktionen	137
8	System Tick Timer	141
8.1	Verwendung des SysTick-Timers	143
8.2	Steuerung mehrerer Funktionen	146
8.3	Steuerung mehrerer Funktionen: Ein verbesserter Ansatz	152
8.4	Die MCAL-Version des Projekts Kap08-SysTick-04	162
9	Timer/Counter (SysTick und GP-Timer)	165
9.1	Der SysTick-Timer	168
9.1.1	Das SysTick Control and Status Register (CTRL)	169
9.1.2	Das SysTick Value Register (VAL)	169
9.1.3	Das SysTick Load Register (LOAD)	169
9.1.4	Das SysTick Calibration Register (CALIB)	169
9.2	Allgemeines zu Timern und Countern	171
9.3	Basic Timer TIM6 und TIM7	173
9.3.1	TIM6/TIM7 Control Register 1/2 (TIMx-CR1 und TIMxCR2)	174
9.3.2	TIM6/TIM7 DMA/Interrupt Enable Register (TIMx_DIER)	175
9.3.3	TIM6/TIM7 Status Register (TIMx_SR)	176
9.3.4	TIM6/7 Event Generation Register (TIMx_EGR)	176
9.3.5	TIM6/TIM7 Counter (TIMx_CNT)	176
9.3.6	TIM6/TIM7 Prescaler (TIMx_PSC)	176
9.3.7	TIM6/TIM7 Auto-Reload Register (TIMx_ARR)	177
9.3.8	Beispiel mit Basic Timer TIM6	177
9.4	Prescaler (Vorteiler) der Busse	181
10	General-Purpose Timer (GP-Timer)	185
10.1	GP-Timer, Teil 1: TIM9 bis TIM14	186
10.1.1	Register der GP-Timer TIM9 bis TIM14	187
10.1.2	Beispiel 1 zum Einsatz von TIM12	192
10.1.3	Beispiel 2 zum Einsatz von TIM12	197
10.2	GP-Timer, Teil 2: TIM2 bis TIM5	200
10.2.1	Einschub: Pulsweitenmodulation (PWM)	201
10.2.2	Beispiel: Dimmen einer LED mittels PWM	207

11	Advanced-Control Timer	211
11.1	Neue Register der Advanced-Control Timer	212
11.2	Einschub: Schalten induktiver Lasten	217
11.3	Beispiel zur Totzeitgenerierung mit dem STM32F446	219
11.3.1	Das Projekt Kap11-ACTIM-DeadTime.	219
12	Digital-Analog-Konverter	223
12.1	Technische Verfahren der D/A-Wandlung	223
12.1.1	Die Parallelwandlung	224
12.1.2	Das Zählverfahren	225
12.1.3	Das R-2R-Verfahren	225
12.2	DACs in der STM32F4xx-Familie.	226
12.2.1	Datenhaltereregister.	227
12.2.2	Datenformate	228
12.3	Die Register des DAC	229
12.4	Ein einfaches Anwendungsbeispiel	231
12.5	Tipps für eigene Anwendungen	235
13	Analog-Digital-Wandlung	237
13.1	ADCs in der STM-Familie.	238
13.2	Register in den STM-ADCs.	242
13.3	Anwendungsbeispiel	247

Teil III Serielle Schnittstellen 251

14	Serielle Kommunikation	253
14.1	Grundlegende Begriffe	254
14.1.1	Kommunikationsrichtungen	254
14.1.2	Aufbau der Daten	255
14.1.3	Datenpegel.	257
14.1.4	Übertragungsgeschwindigkeit.	257
14.1.5	Übertragungsprotokolle	258
14.1.6	Asynchrone vs. synchrone Datenübertragung	259
14.2	Ausführungsformen einfacher RS-232-Schnittstellen	259
15	UARTs und USARTs	261
15.1	Was sind UARTs und USARTs?.	262
15.1.1	Die USART-/UART-Register	263
15.1.2	Empfangen und Senden von Daten	268

16	Inter-Integrated Circuit I²C	279
16.1	Die ursprüngliche Idee hinter I ² C	279
16.2	Prinzipieller Aufbau einer I ² C-Schaltung	280
16.3	Betriebsarten/Protokoll von I ² C	283
16.3.1	Vier Betriebsarten.	283
16.3.2	Das I ² C-Protokoll	284
16.4	I ² C in der STM32F4xx-Familie	287
16.5	Ein Beispiel mit dem PCF8574	294
16.5.1	Der PCF8574	296
16.5.2	Das Programmlisting	297
16.6	Neue MCAL-Funktionen.	304
16.7	Weitere Beispiele	305
16.7.1	Daten lesen von I ² C-Komponenten.	306
16.7.2	Anmerkungen zu den Beispielprojekten	309
17	Serial Peripheral Interface SPI	313
17.1	Datentransfer in SPI-Interfaces	316
17.1.1	CPHA = 1	317
17.1.2	CPHA = 0	317
17.1.3	Anwendung von SPI	318
17.2	SPI-Register der STM32F4xx-Familie	318
17.2.1	SPI Control Register 1 und 2 (SPI_CR1, SPI_CR2)	319
17.2.2	Das SPI Status Register (SPI_SR)	321
17.2.3	Die Checksummen-Register (CRC)	322
17.2.4	Das SPI Data Register (SPI_DR)	324
17.3	Ein einfaches Beispiel mit dem MAX7219	324
17.3.1	Kurzbeschreibung des MAX7219	332
17.4	Eine kleine Übung.	334
Teil IV Weitere Komponenten		337
18	Direct Memory Access (DMA)	339
18.1	Funktionsweise	340
18.2	DMAC(s) in der STM32F4xx-Familie	341
18.2.1	Register der DMACs in der STM32F4xx-Familie	344
18.3	Beispiel: Memory → USART2 → PC mit DMA	349
18.3.1	Erläuterung der Funktionsweise	353
18.4	Beispiel: USART2 → PC mit DMA → USART2.	355
18.4.1	»Probleme« von DMA	361

19	Watchdog	365
19.1	Independent Watchdog (IWDG)	365
19.2	Window Watchdog (WWDG)	367
19.2.1	Funktionsweise	368
19.3	Debuggen und Watchdogs	370
 Teil V Anhang		 371
A	Internetadressen und Literaturnachweise	373
A.1	Literaturnachweise	373
A.2	Infos zur STM32F4xx-Familie	374
A.3	Programmierung in C	374
A.4	Tastaturkürzel von STM32CubeIDE	374
A.5	Internet-Tutorials	374
A.6	Support	375
B	Takteinstellung mit STM32CubeMX	377
B.1	Möglichkeit 1	377
B.2	Möglichkeit 2	381
C	Einführung in das Debuggen	385
C.1	Debugger einrichten	386
C.2	Variablen »beobachten«	390
C.3	Anzeige von Prozessorregistern	392
C.4	Anzeige von SRAM-Inhalten	393
C.5	Vorsicht!	393
D	Funktionen der MCAL-Bibliothek	397
D.1	General-Purpose Input/Output GPIO	397
D.2	Externe Interrupts EXTI	398
D.3	SysTick-Timer	398
D.4	Basic-/GP-/Advanced-Control Timer	399
D.5	UART/USART	400
D.6	Systemfunktionen	401
D.7	Inter-Integrated Circuit I ² C	402
D.8	Serial Peripheral Interface SPI	403
D.9	Direct Memory Access DMA	403
	Stichwortverzeichnis	405

Einleitung

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

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

Hinweis

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

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

Warum STM32F4?

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

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

- In den einschlägigen Internetforen sind sehr viele Informationen und Hilfestellungen in Form von Tutorials zu finden. Im Anhang werde ich Ihnen einige Internetadressen nennen, die ich persönlich als besonders hilfreich empfinde.
- Die (englischsprachige) Dokumentation von STM empfinde ich als vorbildlich und klar strukturiert.
- Einer der wichtigsten Gründe besteht darin, dass STM sehr preisgünstige Evaluierungsboards vertreibt. Das in diesem Buch überwiegend eingesetzte Evaluierungsboard *NUCLEO-F446RE* ist bei einem weltbekannten Onlinehändler bereits zu einem Preis von weniger als 20 Euro erhältlich. Ein Debugger mit Vorrichtung zum Flashen der Software ist hier bereits enthalten!

Hinweis

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

Zielgruppe dieses Buchs

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

Tipp

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

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

Hinweis

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

Voraussetzungen

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

Hinweis

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

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

Hinweis

Die Beispiele in diesem Buch wurden alle mit dem STM32 NUCLEO-64 STM32F446RE getestet. Dies bedeutet auch, dass Peripheriekomponenten, die auf diesem Evaluierungsboard nicht vorhanden sind – hierzu zählen beispielsweise die Ethernet-Schnittstelle oder Komponenten zur Steuerung von Grafik-

displays –, in diesem Buch nicht behandelt werden: Entsprechende Beispiele will ich aber nach und nach auf meiner oben genannten Webseite nachreichen.

Aufbau des Buchs

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

Teil I

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

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

Hinweis

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

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

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

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

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

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

Wichtig

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

Teil II

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

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

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

In **Kapitel 7** lernen Sie abschließend die sogenannten alternativen Funktionen der GPIO-Pins kennen. Standardmäßig werden die GPIOs für die Ein- bzw. Ausgabe digitaler Größen verwendet. Es gibt aber auch Komponenten, die zur Verarbeitung

analoger Größen dienen. Um die Anzahl der Anschlüsse des Mikrocontrollers – und damit die Produktionskosten – so gering wie möglich zu halten, können die GPIOs so konfiguriert werden, dass auch die Verarbeitung nicht-digitaler Signale möglich ist: Zu diesem Zweck verwenden alle Cortex-M-Hersteller die *alternativen Funktionen* (die ich im Verlauf des Buchs auch als *AF* bezeichne).

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

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

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

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

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

Kapitel 12 und 13 befassen sich zum Abschluss von Teil II mit Komponenten, die es uns ermöglichen, die reale analoge Welt mit digitalen Geräten zu erfassen bzw. zu beeinflussen. Die Themen dieser beiden Kapitel sind daher *Digital-Analog-Wandler* (Kapitel 12) und *Analog-Digital-Wandler* (Kapitel 13).

Teil III

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

Schnittstellen Daten mit einer Übertragungsrate von mehreren Hundert MBit (USB) bis hin zu GBit in den neuesten Ethernet-Entwicklungen.

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

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

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

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

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

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

Teil IV

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

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

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

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

Anhänge

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

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

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

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

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

Hinweis

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

Einsatz von Bibliotheken?

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 gefunden).
- 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 für legitim: Schließlich wollen die Hersteller Kunden binden! Es ist aber auch vorstellbar, dass aus bestimmten Gründen der Umstieg auf Mikrocontroller anderer Hersteller erforderlich wird. Die Portierung bereits vorhandener Software würde dann 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 – weitere 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 Prozessorkernen 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

Eine weitere Ausnahme besteht darin, dass wir im weiteren Verlauf selbst eine Bibliothek mit dem Namen *MCAL* entwickeln. *MCAL* ist eine Abkürzung und steht für *Mikrocontroller Abstraction Layer*, ein Begriff, der von vielen Softwareentwicklern verwendet wird. *MCAL* wird schrittweise weiterentwickelt. Neue Funktionalitäten werden zunächst in einem einfachen Beispiel eingeführt, angewendet und erläutert. »Einfach« bedeutet hierbei, dass sie zunächst für eine konkrete Komponente, z.B. *GPIOA* oder den Timer *TIM2*, geschrieben werden. Die gleiche Funktionalität kann aber auch auf ähnliche Komponenten angewendet werden: Funktionen, die im Anschluss in *MCAL* aufgenommen werden, berücksichtigen diesen Umstand, sodass diese Funktionen dann universell eingesetzt werden können. Um die Anzahl der Buchseiten – und hiermit den Preis des Buchs – so niedrig wie möglich zu halten, werden die *MCAL*-Funktionen nicht als Listing abgedruckt. Den Quelltext der Bibliothek stelle ich Ihnen unter <https://www.ralf-jesse.de> kostenlos 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 Buchs 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 Beispielpunkte 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 μ Vision* 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. Die genannten kommerziellen Entwicklungsumgebungen sind auch als kostenlose Evaluierungsversionen verfügbar (siehe den Kasten »Tipp« weiter unten).

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 μ Vision, die IAR Workbench oder auch die Entwicklungsumgebung von Segger werden als Evaluierungsversionen kostenlos angeboten: Dann müssen Sie allerdings einige 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 Buchs 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 Buchs 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.

Hinweis zu den Abbildungen

Im Referenzhandbuch sind in vielen Registern einige Bits mit »Res.« (reserviert) beschriftet, was sich während der Vorbereitung des Drucks als schlecht lesbar herausgestellt hat. Um Ihnen das Lesen zu erleichtern, habe ich mich dazu entschlossen, »Res.« durch einen hellgrauen Hintergrund zu ersetzen.

Ein paar wenige Abbildungen im Buch sind leicht unscharf, dies war leider nicht zu vermeiden. Sie finden diese Abbildungen unter den Downloaddateien zum Buch unter der weiter unten angegebenen Webadresse.

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

An dieser Stelle bleibt mir nur noch, Ihnen viel Spaß und Erfolg beim Durcharbeiten dieses Buchs zu wünschen. Ich erneuere hier nochmals mein Angebot, Sie bei der Lösung von Schwierigkeiten während der Umsetzung eigener Anwendungen zeitnah zu unterstützen: Nutzen Sie hierfür meine E-Mail-Adresse embedded@ralf-jesse.de. Ich freue mich auch über Anregungen und Kritik, um diese in einer weiteren Auflage des Buches berücksichtigen zu können.

Zum Zeitpunkt der Veröffentlichung dieses Buchs werden Sie auf meiner Webseite <https://www.ralf-jesse.de> auch die Beispielprojekte zu diesem Buch herunterladen können. Wenn es meine Zeit zulässt, werde ich dort aber auch weitere Projekte veröffentlichen. Hier plane ich beispielsweise die Ansteuerung von Grafikdisplays und später auch die Implementierung einer Ethernet-Anbindung (dann allerdings mit einem anderen STM32F4-Controller, da der STM32F446 eine solche Schnittstelle nicht enthält). Auch die MCAL-Bibliothek wird weitergepflegt und immer wieder erweitert.

Die Hardware des STM32F4xx-Mikrocontrollers

Will man Software für einen Mikrocontroller entwickeln, ist es hilfreich, wenn grundlegende Kenntnisse zur Hardware vorhanden sind. Die erforderlichen Kenntnisse müssen nicht so tiefgehend 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 einer exemplarischen Beschreibung des STM32F446RE.

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 sowohl die Verfügbarkeit von Flashspeicher (maximal 1.536 KByte) und SRAM (bis zu 320 KByte) als auch die Verfügbarkeit serieller Schnittstellen. Die maximale Taktfrequenz beträgt hier zwischen 84 und 100 MHz.

Abbildung 1.1 zeigt die derzeit verfügbaren Mikrocontroller dieser Familie (Stand: Dezember 2020). 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													
STM32F469	180	512k bis 2056k	384	✓	✓	✓	✓	✓	✓		✓	✓	✓
STM32F429	180	512k bis 2056k	256	✓	✓	✓	✓		✓		✓	✓	
STM32F427	180	1024k bis 1024k	256	✓	✓	✓	✓		✓		✓		
Foundation Lines													
STM32F446	180	256k bis 512k	128		✓	✓	✓	✓	✓	✓			
STM32F407	168	512k bis 1024k	192	✓	✓	✓							
STM32F405	168	512k bis 1024k	192		✓								
Access Lines													
Produktlinie	FCPU (MHz)	Flash (kBytes)	RAM (KB)	Stromaufw. µA/MHz (Min.)	Stromaufw. (Stopp) (Min.)	Abm. (min.) in mm	FSMC (NOR/PSRAM/ LCD-Support)	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	✓

- ART Accelerator™
- SDIO
- USART, SPI, I²C
- I²S + Audio-PLL
- 16- und 32-Bit Timer
- 12-Bit ADC (0.41 µs)
- True Number Random Generator
- Batch Acquisition Mode
- Low Voltage (1.7 - 3.6V)
- Temperature: -40°C - 125°C

*I/F = Interface (Schnittstelle)

Abb. 1.1: Überblick über die STM32F4-Familie

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/I ² S	4/3 (simplex)								
	I ² C	4/1 davon als FMP+ (Fast mode Plus)								
	USART/UART	4/2								
	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								
	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		LQFP144 UFBGA144		

Tabelle 1.1: Ausstattung des STM32F446

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.

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.

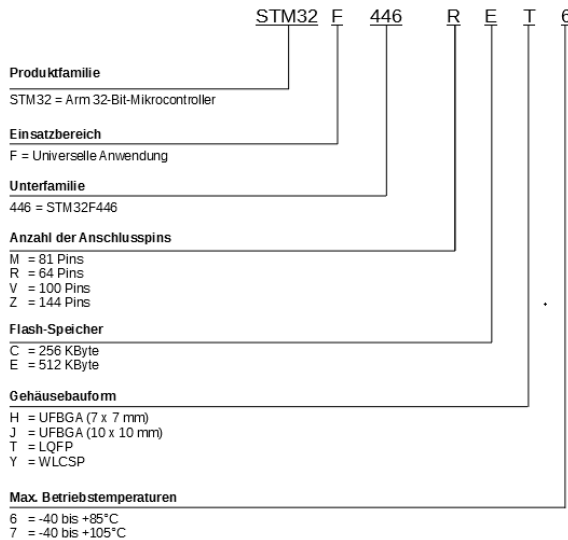


Abb. 1.2: Produktbezeichnungen der STM32F4xx-Mikrocontroller

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 lernt.

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 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 erläutert.

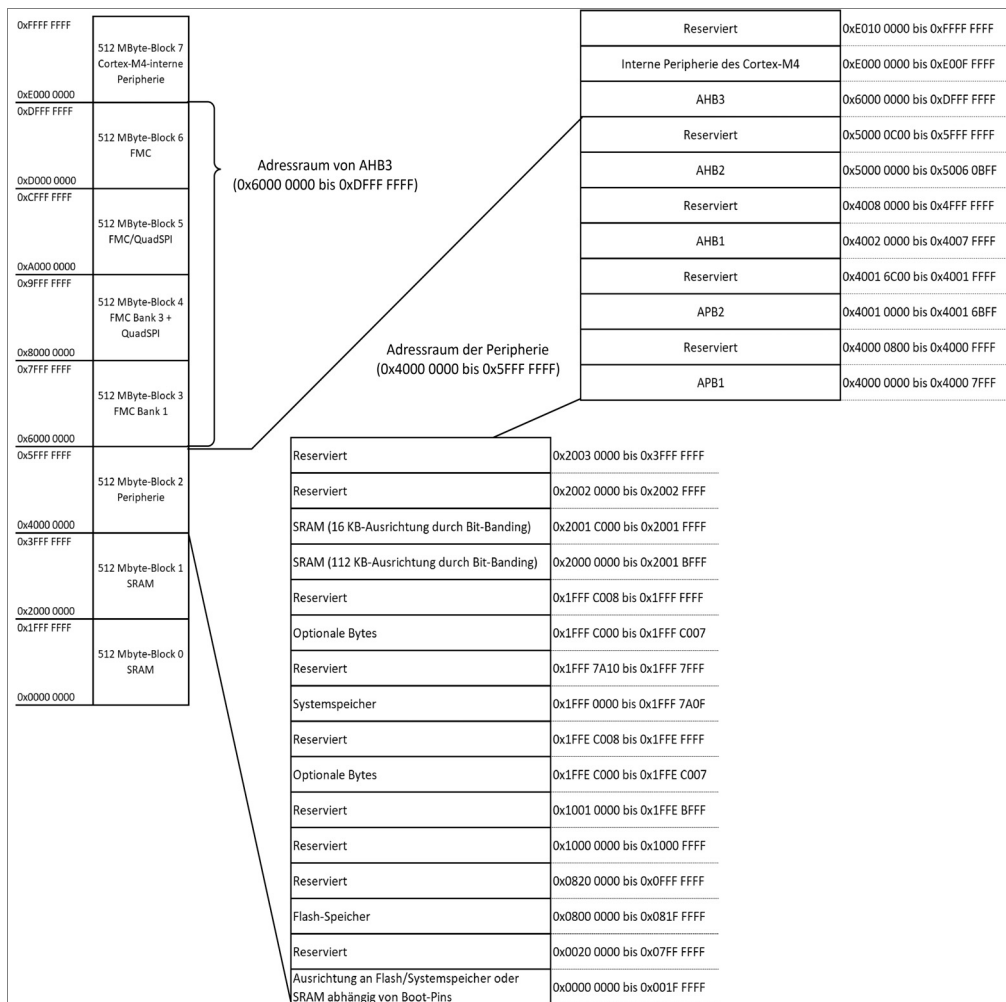


Abb. 1.3: Speicherbereiche (Memory-Mapping), aus dem Datenblatt entnommen

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). Ich habe bewusst auf die Angabe der jeweiligen Endadresse verzichtet, da sich die Größe der beiden Bereiche innerhalb der STM32F4xx-Familie unterscheidet.

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 Tabellen weggelassen.

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)
APB1	0x4000 7400–0x4000 77FF	DAC (Digital-to-Analog Converter)
	0x4000 7000–0x4000 73FF	PWR (Power Control)
	0x4000 6C00–0x4000 6FFF	HDMI-CEC (High Definition Multi-media 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)
	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

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 leistungsfähiger 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.

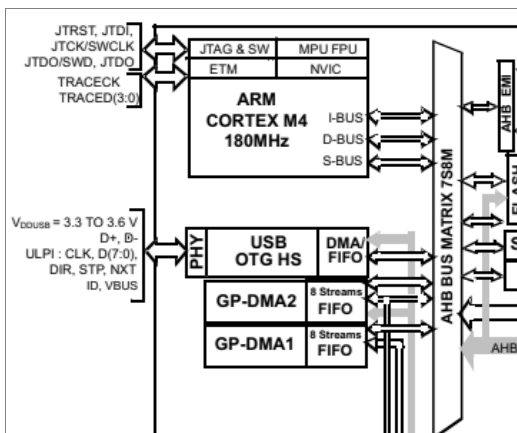


Abb. 1.4: Das »Herz« der STM32F4xx-Familie

Neben der *Central Processing Unit (CPU)* – hier als ARM CORTEX M4 180MHz 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 6 »Polling, Interrupts und Exceptions« 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 »tickt« er in einem 10-Millisekunden-Takt. Andere Taktzeiten (auch kürzere) sind aber ebenfalls möglich.
- **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 bedeuten »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) stehen 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 oder das Messen (Zählen) von Impulsen, die von außen zugeführt werden. 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 8 bis 11 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, I²C, I²S usw.).
- Die Natur ist nicht digital, sondern analog. Dies bedeutet, dass sich physikalische Größen niemals 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 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 erst 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 1.3 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«.

- 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) 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 alle Peripheriekomponenten mit *Vorteilern (Prescaler)* ausgestattet, die den Peripherietakt weiter verringern. An welchen Bus die Peripheriekomponenten angeschlossen sind, können Sie dem Blockschaltbild (siehe den oben genannten Link) sehr einfach entnehmen.

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

Stichwortverzeichnis

A

- Abfrage externer Komponenten
 - Exceptions 15
 - Interrupts 15
- Analog-Digital-Umsetzer (ADC, Analog-to-Digital Converter) 237
 - ADC Control Register 1 (ADC_CR1) 242
 - ADC Control Register 2 (ADC_CR2) 243
 - ADC Regular Data Register (ADC_DR) 244
 - ADC Regular Sequence Register 1/2/3 (ADC_SQR1, ADC_SQR2, ADC_SQR3) 246
 - ADC Sample Time Register 1 und 2 (ADC_SMPR1, ADC_SMPR2) 245
 - ADC Status Register (ADC_SR) 244
 - Conversion Time 245
 - injizierte Gruppen (Injected Groups) 242
 - reguläre Gruppen 242
 - reguläre Interrupts 241
 - Sample Time 245
 - sukzessive Approximation 237
- Aufbau der STM32F4xx-Familie
 - ADC (Analog-to-Digital Converter) 35
 - Central Processing Unit (CPU) 33
 - DAC (Digital-to-Analog Converter) 35
 - DMA-Controller (Direct Memory Access) 36
 - Embedded Trace Macrocell (ETM) 34
 - Ethernet 36
 - GPIOs (General Purpose Inputs/Outputs) 35
 - JTAG (Joint Test Action Group) 34
 - MPU (Memory Protection Unit) 34
 - NVIC (Nested Vector Interrupt Controller) 34
 - RTC (Real Time Clock) 35
 - SCB (System Control Block) 34
 - SWD, Serial Wire Debug 34
 - System Timer 34
 - SysTick-Timer 34
 - TFT-/LCD-Interfaces 36
 - Timer 35
 - Watchdog-Timer 35

B

- Backup Space 78
- Bare-Metal-Programmierung 40
- Behandlung von Ereignissen
 - Interrupt Request (Interrupt-Anforderung) 109
 - Interrupt-Handler 110
 - Interrupts 108
 - ISR, Interrupt-Service-Routine 110
 - Nested Vector Interrupt Controller (NVIC) 110, 112
 - Non-maskable Interrupt (NMI) 111
 - Polling 108
 - Segmentation Fault 111
 - Tail Chaining 113
- Bibliotheken
 - CMSIS (Cortex Microcontroller Software Interface Standard) 14, 19
 - HAL 18
 - MCAL (Microcontroller Abstraction Layer) 20
 - Mikrocontroller Abstraction Layer 20
 - SPL (Standard Peripheral Library) 18
- Blocking Function 142
- Bootvorgang 53
 - Bare-Metal (ohne Betriebssystem) 53
 - BIOS (Basic Input/Output System) 53
 - UEFI (Unified Extended Firmware Interface) 53
- Bussysteme
 - AHB1 (Advanced High-Performance Bus) 37
 - AHB2 37
 - AHB-/APB-Brücken 37
 - APB1 (Advanced Peripheral Bus) 37
 - APB2 37
 - D-Bus (Data Bus) 36
 - DMA 37
 - I-Bus (Instruction Bus) 36
 - S-Bus (System Bus) 37
 - USB OTG HS 37
 - Vorteiler (Prescaler) 37

C

Cortex-M4-Controller 11

D

Datensegmente

BSS (Block Started by Symbol) 61

Heap 62

Stack + Stackpointer 61

Digital-Analog-Konverter (DAC, Digital-to-Analog Converter) 223

Ausrichtung linksbündig 229

Ausrichtung rechtsbündig 229

DAC Channel x 12-bit Left-Aligned Data Holding Register (DAC_DHL12Rx) 231

DAC Channel x 12-bit Right-Aligned Data Holding Register (DAC_DHR12Rx) 231

DAC Channel x 8-bit Right-Aligned Data Hold Register (DAC_DHR8Rx) 231

DAC Channel x Data Output Register (DAC_DORx) 231

DAC Control Register (DAC_CR) 229

Dual DAC Channel 12-bit Left-Aligned Data Holding Register (DAC_DHL12LD) 231

Dual DAC Channel 8-bit Right-Aligned Data Hold Register (DAC_DHR8RD) 231

Dual DAC Channel x 12-bit Right-Aligned Data Holding Register (DAC_DHR12RD) 231

Dual-Channel-Betrieb 229

Parallelwandlung 224

Single-Channel-Betrieb 229

Digital-Analog-Wandler (D/A-Wandler) 202

Direct Memory Access (DMA) 339

Arbiter 342

Channels 342

DACK (DMA Acknowledge) 340

DMA High Interrupt Flag Clear Register (DMA_HIFCR) 347

DMA High Interrupt Status Register (DMA_HISR) 347

DMA Low Interrupt Flag Clear Register (DMA_LIFCR) 347

DMA Low Interrupt Status Register (DMA_LISR) 347

DMA Stream x Configuration Register (DMA_SxCR) 344

DMA Stream x FIFO Control Register (DMA_SxFCR) 347

DMA Stream x Memory 0 Register (DMA_SxM0AR) 348

DMA Stream x Memory 1 Register (DMA_SxM1AR) 348

DMA Stream x Number of Data Register (DMA_SxNDTR) 342, 347

DMA Stream x Peripheral Address Register (DMA_SxPAR) 348

DMAC (DMA Controller) 340

FIFO (First In, First Out) 342

HLDA (HOLD Acknowledge) 340

HOLD Request (HRQ) 340

Read-Modify-Write 339

Ringbuffer 342

Schwellwert (Threshold) 342

Streams 342

Direction Bit (DIR) 186

DMA (Direct Memory Access) 262

E

Echtzeituhr

RTC (Real Time Clock) 57

Entwicklungsumgebung 20

Embedded Studio 21

IAR Workbench 21

IDE, Integrated Development Environment 20

Keil µVision 21

STM32CubeIDE 20

Target Selector 48

Exceptions

BusFault_Handler 133

HardFault_Handler 133

MemManage_Handler 133

F

Flexible Memory Controller

Memory Bank Switching 85

Flexible Memory Controller (FMC)

Speicherbänke 85

Floating Point Unit (FPU) 122

G

General Purpose Inputs/Outputs (GPIO) 15

GNU Compiler Collection (GCC) 54

GPIO

Aktoren 91

General Purpose Input/Output 91

Sensoren 91

GPIO-Register

- atomarer Zugriff 99
- GPIO Port Alternate Function Low Register (GPIOx_AFR_L) 99
- GPIO Port Alternative Function High Register (GPIOx_AFR_H) 102
- GPIO Port Bit Set/Reset Register (GPIOx_BSRR) 97
- GPIO Port Configuration Lock Register (GPIOx_LCKR) 99
- GPIO Port Input Data Register (GPIOx_IDR) 96
- GPIO Port Mode Register (GPIOx_MODER) 93
- GPIO Port Output Data Register (GPIOx_ODR) 97
- GPIO Port Output Speed Register (GPIOx_OSPEEDR) 95
- GPIO Port Output Type Register (GPIOx_OTYPER) 94
- GPIO Port Pull-up/Pull-Down Register (GPIOx_PUPDR) 96
- GPIO_AFRH (GPIO Alternate Function Register High) 135
- GPIO_AFR_L (GPIO Alternate Function Register Low) 135
- GPIO_MODER (GPIO Mode Register) 137
- GPIOx Alternate Function Register High 137
- GPIOx Alternate Function Register Low 137
- Lock Key 99

GPIOs

- alternative Funktionen 136

I**Inter-Integrated Circuit I2C**

- I²C TRISE Register (I2C_TRISE) 293

Inter-Integrated Circuit I²C 279

- Clock Stretching 283
- Fast Speed I²C 280
- Fast Speed I²C Plus 280
- I squared C 279
- I2C 279
- I2C Control Register 1 (I2C_CR1) 288
- I²C Clock Control Register (I2C_CCR) 292
- I²C Data Register (I2C_DR) 291
- I²C FLTR Register (I2C_FLTR) 294
- I²C Own Address Register (I2C_OAR1, I2C_OAR2) 293

- I²C Status Register (I2C_SR1 und I2C_SR2) 291

- IIC 279

- Multi-Byte Burst Write 287

- Multi-Master-/Multi-Slave-System 280

- Repeated Start 285

- SCL (Serial Clock) 279

- SDA (Serial Data) 279

- Startsignal 285

- Stoppsignal 285

- System Management Bus (SMB) 279

- TWI (Two-Wire Interface) 279

Interrupts

- Interrupt-Service-Routine (ISR) 145

L

- Linker 59

- Linkerscript 56, 60

M

- Majerle, Tilen (<https://majerle.eu>) 167

- MCU-Betriebsarten

- Privileged Mode 122

- Unprivileged Mode 122

- Memory Mapping 14

N

- Not-Aus 109

- Not-Halt 109

- NUCLEO-F446RE 12

- Nyquist-Shannon-Abtasttheorem 188

P

- Peripheriekomponenten

- RCC (Reset and Control Register) 57

- SCB (System Control Block) 57

- Produktlinien 25

- Access Line 25

- Advanced Line 25

- Foundation Line 25

- Programmierstandard

- MISRA-C-Standard 71

- Statische Codeanalyse 74

- Prozessorregister

- CPACR (Co-Processor Access Register) 57

- EXTI Event Mask Register

- (EXTI_EMR) 121

- EXTI Falling Trigger Selection

- (EXTI_FTSR) 122

- EXTI Interrupt Mask Register (EXTI_IMR) 121
- EXTI Pending Register (EXTI_PR) 122
- EXTI Rising Trigger Selection (EXT_RTSTR) 122
- EXTI Software Interrupt Event (EXTI_SWIER) 122
- Flexible Memory Controller (FMC) 85
- RCC AHB1 Peripheral Clock Enable in Low Power Mode (RCC_AHB1LPENR) 83
- RCC AHB1 Peripheral Clock Enable Register (RCC_AHB1ENR) 81
- RCC AHB1 Peripheral Reset Register (RCC_AHB1RSTR) 83
- RCC AHB2 Peripheral Clock Enable in Low Power Mode (RCC_AHB2LPENR) 83
- RCC AHB2 Peripheral Reset Register (RCC_AHB2RSTR) 83
- RCC AHB3 Peripheral Clock Enable in Low Power Mode (RCC_AHB3LPENR) 83
- RCC AHB3 Peripheral Reset Register (RCC_AHB3RSTR) 83
- RCC APB1 Peripheral Clock Enable in Low Power Mode (RCC_APB1LPENR) 83
- RCC APB1 Peripheral Reset Register (RCC_APB1RSTR) 83
- RCC APB2 Peripheral Reset Register (RCC_APB2RSTR) 83
- RCC Clock Configuration Register (RCC_CFGR) 182
- RCC Clock Control & Status Register (RCC_CSR) 84
- RCC Configuration Register (RCC_CFGR) 84
- RCC_AHB2 Peripheral Clock Enable Register (RCC_AHB2ENR) 81
- RCC_AHB3 Peripheral Clock Enable Register (RCC_AHB3ENR) 81
- RCC_APB1 Peripheral Clock Enable Register (RCC_APB1ENR) 81
- RCC_APB2 Peripheral Clock Enable Register (RCC_APB2ENR) 81
- Reset and Clock Control (RCC) 63, 77
- Reset and Control Register RCC 57
- SYSCFG_EXTICRn 120
- System Configuration Controller (SYSCFG) 77, 85
- System Control Block (SCB) 77
- SysTick Calibration (CALIB) 169
- SysTick Control and Status Register (CTRL) 169
- SysTick LOAD Register (LOAD) 169
- SysTick Value Register (VAL) 169
- SysTick_Type 168
- USART Status Register (USART_SR) 112
- Vector Table Offset Register (VTOR) 57, 86, 117
- PWM**
 - Center Alignment 205
 - Duty-Cycle 204
 - Duty-Zeit 203
 - Edge Alignment (linksbündig) 205
 - getaktete Spannung 204
 - Left Edge Alignment 205
 - Pulsweitenmodulation (PWM) 201
 - Right Edge Alignment 205
 - Tastverhältnis 204
 - Vorwärtsspannung (Forward Voltage) 204
 - Vorwärtsstrom (IF) 204
- PWM-Erzeugung (PWM = Pulse Width Modulation, Pulsweitensteuerung) 200
- R**
 - Read-Modify-Write 36
 - Register der STM32F4xx-Familie
 - RCC Clock Control & Status Register, RCC_CSR 78
 - RCC Clock Control Register, RCC_CR 80
 - Reset and Clock Control, RCC 77
 - Reset_Handler 54
 - Reset-Arten
 - Backup Domain Reset 77
 - Brown Out Detection, BOD, BOR 78
 - Low-Power Management Reset 78
 - Power Down Reset, PDR 78
 - Power On Reset, POR 78
 - Power Reset 77
 - System Reset 77
- S**
 - Segmente
 - Datensegment(e) 55
 - Textsegment(e) 55
 - Serial Peripheral Interface (SPI) 313
 - Cyclic Redundancy Check (CRC) 322
 - Generator-Polynom 322

Halb-Duplex 313
 I²S (Inter-Integrated Circuit Sound) 316
 Polynom-Grad 322
 SCK 313
 SDI (MISO) 313
 SDO (MOSI) 313
 Simplex 314
 SPI Control Register 1 (SPI_CR1) 319
 SPI Control Register 2 (SPI_CR2) 319
 SPI CRC Polynomial Register
 (SPI_CRCPR) 322, 323
 SPI Data Register (SPI_DR) 324
 SPI RX CRC Register
 (SPI_RXCRCR) 322, 324
 SPI Status Register (SPI_SR) 321
 SPI TX CRC Register
 (SPI_TXCRCR) 322, 323
 SS (Slave Select) 313
 Voll-Duplex-Bus 313
 Serielle Schnittstellen
 asynchron 259
 Datenrahmen (Frame) 255
 DMA-Controller 275
 Duplexer 255
 EIA232 259
 even parity 256
 Halb-Duplex 254
 Handshaking 258
 no parity 256
 Nutzdaten 255
 odd parity 256
 Paritätsbit 256
 Pegelwandler 257
 Protokoll 258
 Ringbuffer (Circular Buffer) 273
 RS232 259
 Sicherungsdaten 255
 Simplex 254
 Startbit 256
 synchron 259
 UARTs 262
 USARTs 262
 virtuelle COM-Ports 260
 Voll-Duplex 255
 SPL 18
 Stackpointer 56
 Stages 54
 System Configuration Controller (SYSCFG)
 119, 120

T

Takterzeugung
 High Speed External (HSE) 57, 78
 High Speed Internal (HSI) 57, 78
 HSE (High Speed External) 57, 78
 HSI RC (High Speed Internal) 57, 78
 LSE (Low Speed External) 57
 PLL (Phased Lock Loop) 58, 78
 Timer
 Advanced-Control Timer 211
 Auto-Reload Register (ARR) 174
 Basic Timer 173
 TIM6 173
 TIM7 173
 General-Purpose Timer (GP-Timer) 185
 H-Brücke 215
 One-Shot-Timer 173
 Repetition Counter (Wiederholungszähler) 211
 skalierbare Architektur 166
 SysTick 143
 TIM6/TIM7 Count Register
 (TIMx_CNT) 176
 TIM6/TIM7 Event Generation Register
 (TIMx_EGR) 176
 TIM6/TIM7 Prescaler (TIMx_PSC) 176
 TIM6/TIM7 Status Register
 (TIMx_SR) 176
 TIMx Auto-Reload Register
 (TIMx_ARR) 177
 TIMx Break and Dead-Time Register
 (TIMx_BDTR) 215
 TIMx Capture/Compare Enable Register
 (TIMx_CCER) 212
 TIMx Capture/Compare Register
 (TIMx_CCRn) 205
 TIMx Control Register (TIMx_CR1 und
 TIMx_CR2) 174
 TIMx Counter (TIMx_CNT) 174
 TIMx DMA/Interrupt Enable Register
 (TIMx_DIER) 175
 TIMx Repetition Counter Register
 (TIMx_RCR) 214
 Totzeiten (Dead Time) 211
 Timer-Funktionen
 Input Capture 185
 One-Shot 185
 Output Compare 185
 Timer-Register
 TIMn Capture/Compare Enable Register
 (TIMn_CCER) 191

TIMn Capture/Compare Mode Register
n (TIMn_CCMRn) 189
TIMn Capture/Compare Register
(TIMn_CCRx) 190
TIMn Slave Mode Control Register
(TIMn_SMCR) 189
TIMx Control Register n (TIMx_CRn)
200
TIMx Option Register (TIMx_OR) 187

U

UART/USART

fraktionale Baudratengeneratoren 266
IrDA = Infrared Data Association 262
LIN-Bus (LIN = Local Interconnect
Bus) 262
Oversampling (Überabtastung) 267
Universal Asynchronous Receiver/
Transmitter 262
Universal Synchronous and Asynchro-
nous Receiver/Transmitter 262
USART Baud Rate Register
(USART_BRR) 266
USART Control Register 1
(USART_CR1) 265

USART Control Register 2
(USART_CR2) 265
USART Data Register (USART_DR) 264
USART Status Register (USART_SR)
263

Überladung von Funktionen 67

V

Vector Table 56

Vector Table Offset Register 56

W

Watchdog 365

Early-Wakeup-Interrupt 369
Independent Watchdog (IWDG) 365
IWDG Key Register (IWDG_KR) 367
IWDG Prescaler Register (IWDG_PR)
366
IWDG Reload Register (IWDG_RLR)
366
IWDG Status Register (IWDG_SR) 367
Window Watchdog (WWDG) 365, 367
WWDG Configuration Register
(WWDG_CFR) 369
WWDG Control Register (WWDG_CR)
368