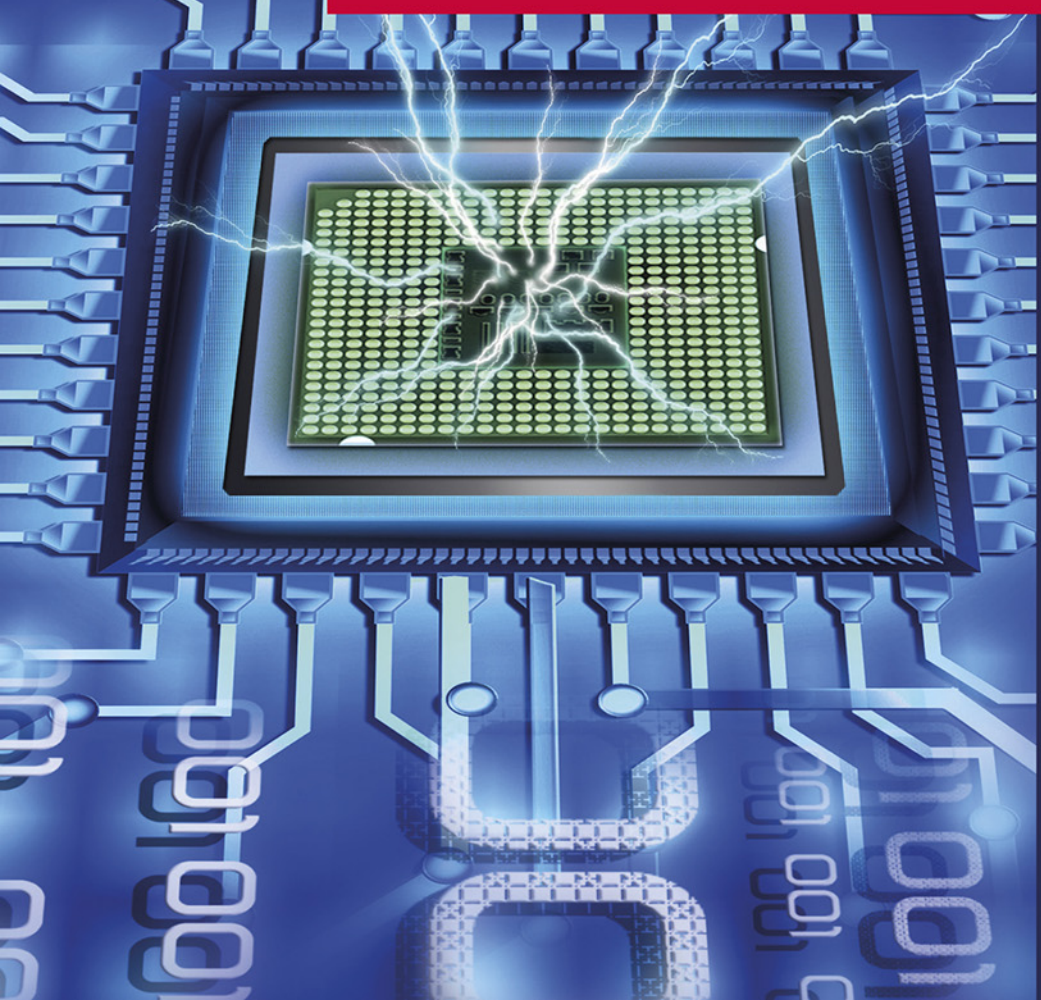
mitp

Ralf
Jesse

STM32

ARM-Mikrocontroller programmieren für Embedded Systems

Das umfassende Praxisbuch



Hinweis des Verlages zum Urheberrecht und Digitalen Rechtemanagement (DRM)

Liebe Leserinnen und Leser,

dieses E-Book, einschließlich aller seiner Teile, ist urheberrechtlich geschützt. Mit dem Kauf räumen wir Ihnen das Recht ein, die Inhalte im Rahmen des geltenden Urheberrechts zu nutzen. Jede Verwertung außerhalb dieser Grenzen ist ohne unsere Zustimmung unzulässig und strafbar. Das gilt besonders für Vervielfältigungen, Übersetzungen sowie Einspeicherung und Verarbeitung in elektronischen Systemen.

Je nachdem wo Sie Ihr E-Book gekauft haben, kann dieser Shop das E-Book vor Missbrauch durch ein digitales Rechtemanagement schützen. Häufig erfolgt dies in Form eines nicht sichtbaren digitalen Wasserzeichens, das dann individuell pro Nutzer signiert ist. Angaben zu diesem DRM finden Sie auf den Seiten der jeweiligen Anbieter.

Beim Kauf des E-Books in unserem Verlagsshop ist Ihr E-Book DRM-frei.

Viele Grüße und viel Spaß beim Lesen,

Ihr mitp-Verlagsteam

mitp

Neuerscheinungen, Praxistipps, Gratiskapitel,
Einblicke in den Verlagsalltag –
gibt es alles bei uns auf Instagram und Facebook

instagram.com/mitp_verlag

facebook.com/mitp.verlag

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 978-3-7475-0192-4
1. Auflage 2021

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

© 2021 mitp Verlags GmbH & Co. KG, Frechen

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

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

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

Inhaltsverzeichnis

Einleitung

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

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

Hinweis

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

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

Warum STM32F4?

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

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

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

Hinweis

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

Zielgruppe dieses Buchs

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

Tipp

Um Ihnen die Suche nach Informationen im Internet zu erleichtern, habe ich in Anhang A eine nach Themen sortierte Sammlung von derzeit gültigen Internetadressen (Stand: Dezember 2020) zusammengestellt. Ich habe mich dabei auf sichere Webseiten (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 `lpcopen` von NXP, beschrieben: Ihre Verwendung würde die Portierbarkeit von Software auf Mikrocontroller anderer Hersteller erheblich schwieriger gestalten.

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

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

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

Wichtig

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

Teil II

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

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

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

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

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

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

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

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

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

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

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

Teil III

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

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

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

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

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

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

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

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

Teil IV

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

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

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

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

Anhänge

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

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

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

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

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

Hinweis

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

Einsatz von Bibliotheken?

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 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

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 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. 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 µVision, 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.

Teil I

Grundlagen

In diesem Teil:

Dieser erste Teil umfasst vier Kapitel, in denen Sie viele grundlegende Informationen zu den Cortex-M4-Mikrocontrollern der Firma *STMicroelectronics* (kurz STM) erhalten.

Kapitel 1 liefert einen Überblick über die gesamte Cortex-M4-Familie von STM. Da die Softwareentwicklung von Embedded Systemen sehr hardwarenah erfolgt, finden Sie hier Informationen über den internen Aufbau der Mikrocontroller und ihre Bussysteme. Die Informationen reichen nicht aus, um eigene Hardware zu entwickeln, versetzen Sie aber in die Lage, sich mit den Hardwareentwicklern auf einem gemeinsamen Sprachniveau auszutauschen.

CMSIS, der *Cortex Microcontroller Software Interface Standard*, stellt eine Beschreibung des internen Aufbaus als Software dar. Die Firma Arm hat bereits früh erkannt, dass die Vielzahl der Halbleiterfirmen, die die Chips »in Silizium gießen«, mit Sicherheit eigene Softwarestandards in Form von Softwarebibliotheken entwickelt hat, die ihre Produkte voneinander abgrenzen. CMSIS stellt nun einen Standard dar, der einen Wechsel von einem Hersteller zu einem anderen erleichtert. In Kapitel 2 bauen Sie sich eine CMSIS-Bibliothek, die Sie in eigenen Projekten verwenden können. Im weiteren Verlauf des Buchs werden Sie mit der MCAL-Bibliothek eine eigene Funktionssammlung entwickeln, die ausschließlich auf der Programmierung der internen Register basiert. Die Umsetzung erfolgt auf die gleiche Weise, die bereits für CMSIS verwendet wurde.

Ich habe im Laufe meiner beruflichen Praxis als Ingenieur in der Entwicklung von Embedded Software sehr viel Code gesehen, der nur schwer verständlich war. Gerade die Programmiersprache C erlaubt den Softwareentwicklern sehr viele Freiheiten, wodurch der resultierende Code allerdings schwerer zu pflegen ist. In verschiedenen Branchen, wie z.B. der Automobilindustrie, der Luft- und Raumfahrt oder im Kraftwerkbau, werden aber besonders hohe Ansprüche an die Sicherheit und die Pflege der Software gestellt. Die Folge hiervon war die Entwicklung sogenannter Coding-Standards, wie z.B. MISRA-C in der Automobilindustrie. In Kapitel 3 stelle ich Ihnen bewährte Vorschriften zum Einsatz von Embedded C vor. Ihre Einhaltung wird in der Industrie zwingend vorausgesetzt und permanent überprüft.

In Kapitel 4 gehe ich auf Dinge ein, die Sie – unabhängig vom Mikrocontrollerhersteller – immer beachten müssen. Hierzu zählt die Basiskonfiguration jeder Embedded-C-Software für Cortex-M4-Mikrocontroller.

Kapitel 1

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 STM 32F446RE.

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														
• 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	✓	✓	✓	✓		✓		✓		
Foundation 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) • Temperature: -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/ 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	✓

*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^2S	4/3 (simplex)							
	I^2C	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.

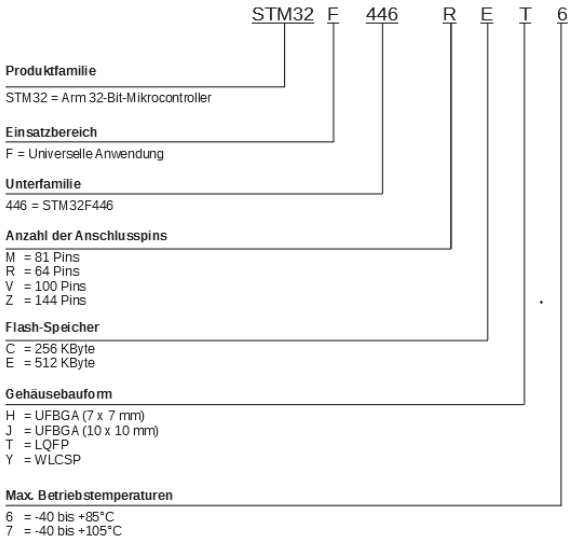


Abb. 1.2: Produktbezeichnungen der STM32F4xx-Mikrocontroller

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.