Einleitung

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

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

Hierzu möchte ich Ihnen ein paar Beispiele nennen:

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

des Bustakts heißt nun entsprechend `timerSelectTimer()`, die von UARTs/USARTs somit `usartSelectUsart()`.

- Bei den Timern wurden Input-Capture-/Output-Compare-Funktionen ursprünglich in einer gemeinsamen Funktion behandelt. Da sich die Anzahl der verfügbaren Input-Capture-/Output-Compare-Kanäle der einzelnen Timer unterscheidet (sie liegt zwischen null und vier), hätte alleine die Behandlung der möglichen Kanal-Kombinationen acht Fallunterscheidungen erfordert. Zusätzlich müsste die Funktion aber auch prüfen, ob der ausgewählte Timer überhaupt geeignet ist, was weitere Fallunterscheidungen erfordert hätte. Ich habe mich daher letztendlich dazu entschlossen, die Input-Capture-Funktionen völlig von den Output-Compare-Funktionen zu trennen. So entstanden aus ursprünglich einer `timerSetCapCompMode()`-Funktion im Verlauf beispielsweise die Funktionen `timerSetInputCaptureMode()` bzw. `timerSetOutputCompareMode()`.
- Die GPIO-Funktionen `gpioGetPinState()` und `gpioGetPortVal()` habe ich inzwischen stark überarbeitet. Haben sie ursprünglich die gewünschten Ergebnisse in einer Variablen gespeichert, auf die über einen Pointer zugegriffen werden musste, habe ich sie nun so geändert, dass sie die Ergebnisse unmittelbar zurückliefern.

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

Hinweis

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

Worum es geht

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

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

Hinweis

Dieses Buch befasst sich – wie bereits oben erwähnt – ausschließlich mit den Cortex-M4-Mikrocontrollern von STM. Aufgrund der sehr guten Skalierbarkeit der Cortex-M-Mikrocontroller von STM lassen sich die in diesem Buch beschriebenen Techniken aber auch mit den neuen STM32F7xx-Mikrocontrollern einsetzen! Auch Nutzer der Cortex-M0-, Cortex-M3- oder von Cortex-M23-Mikrocontrollern können von diesem Buch profitieren.

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

Warum STM32F4?

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

- In den meisten Unternehmen, in denen ich seit mehr als 30 Jahren als Softwareentwickler im Mikrocontrollerbereich arbeite oder gearbeitet habe, werden seit vielen Jahren Mikrocontroller von STM eingesetzt.
- In den einschlägigen Internetforen sind sehr viele Informationen und Hilfestellungen in Form von Tutorials zu finden. Im Anhang werde ich Ihnen einige Internetadressen nennen, die ich persönlich als besonders hilfreich empfinde.
- Die (englischsprachige) Dokumentation von STM empfinde ich als vorbildlich und klar strukturiert.
- Einer der wichtigsten Gründe besteht darin, dass STM sehr preisgünstige Evaluierungsboards vertreibt. Das in diesem Buch eingesetzte Evaluierungsboard

NUCLEO-F446RE ist bei einem weltbekannten Onlinehändler bereits zu einem Preis von weniger als 30 Euro erhältlich. Ein Debugger mit Vorrichtung zum Flashen der Software ist hier bereits enthalten!

Hinweis

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

Zielgruppe dieses Buches

Ich gehe davon aus, dass jeder Leser dieses Buches der englischen Sprache so weit mächtig ist, dass er die Originaldokumentation der Hersteller nachvollziehen kann. Dennoch erleichtert es die Entwicklungsarbeit häufig, wenn weitere Dokumentation oder Literatur auch in der eigenen Muttersprache verfügbar ist. Meines Wissens existiert derzeit nur noch ein weiteres deutschsprachiges Buch zu STMs Cortex-M-Mikrocontrollern, das sich hauptsächlich an Anwender richtet, die erste Erfahrungen in der Arduino-Welt gesammelt haben.

Tipp

Um Ihnen die Suche nach Informationen im Internet zu erleichtern, habe ich in Anhang A eine nach Themen sortierte Sammlung von derzeit gültigen Internetadressen (Stand: März 2022) zusammengestellt. Ich habe mich dabei – mit einer Ausnahme – auf sichere Webseiten (https://...) beschränkt.

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

Hinweis

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

Voraussetzungen

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

Hinweis

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

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

Hinweis

Die Beispiele in diesem Buch wurden alle mit dem STM32 NUCLEO-64 STM32F446RE getestet. Dies bedeutet auch, dass Peripheriekomponenten, die auf diesem Evaluierungsboard nicht vorhanden sind – hierzu zählen beispiels-

weise die Ethernet-Schnittstelle oder Komponenten zur Steuerung von Grafikdisplays –, in diesem Buch nicht behandelt werden: Entsprechende Beispiele will ich aber nach und nach auf meiner oben genannten Webseite nachreichen.

Aufbau des Buches

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

Teil I

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

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

Hinweis

Wenn Sie einen anderen Mikrocontroller dieser Familie verwenden, finden Sie die entsprechenden Informationen im jeweiligen Datenblatt (Datasheet). Besonders wichtig ist, dass Sie hier zusätzlich die entsprechenden Referenzhandbücher von `https://st.com` herunterladen!

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

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

Kapitel 2 befasst sich mit der Erstellung der *CMSIS*-Bibliothek (CMSIS = *Cortex Microcontroller Software Interface Standard*). Hierbei handelt es sich um eine Sammlung von Funktionen und Konstanten, die in diesem Buch verwendet wird und die die Basis für die im Buch gemeinsam entwickelte MCAL-Bibliothek darstellt. CMSIS ist dabei die einzige Fremdbibliothek, die hier verwendet wird. Darüber hinaus beschreibt Kapitel 2 den Bootvorgang sowie die Grundinitialisierung des

Mikrocontrollers und gibt einführende Hinweise zur Einstellung des Taktsignals. Auch eine Beschreibung, die das Verständnis von Linkerscripts erleichtern soll, finden Sie in Kapitel 2.

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

In **Kapitel 4** werden die Register vorgestellt, die für das Reset-Verhalten und die Steuerung von Taktsignalen (*Reset and Clock Control*, *RCC*) zuständig sind. Das hier Beschriebene ist in allen Projekten zu beachten, die Sie entwickeln! Besonders die Abschnitte zum RCC sind nur als Einstieg zu verstehen, da diese Komponente auch für die Einstellung der Systemtaktfrequenzen verwendet wird (siehe Kapitel 5).

Kapitel 5 wurde vollständig neu erstellt, was zur Folge hat, dass sich – im Vergleich zur ersten Auflage – alle Folgekapitel entsprechend verschieben. Hier wird die Einstellung der Taktfrequenzen des Prozessorkerns, der verschiedenen Busse sowie der Peripheriekomponenten beschrieben. In der ersten Auflage hatte ich zu diesem Thema noch auf ein besonderes Feature der STM32CubeIDE-Entwicklungsumgebung verwiesen: Die dort beschriebene Vorgehensweise verwende ich inzwischen selbst nicht mehr. Werksseitig sind die NUCLEO-Boards so konfiguriert, dass das System mit einer Frequenz in Höhe von 16 MHz getaktet wird. In diesem neuen Kapitel lernen Sie nun, wie Sie vorgehen müssen, um die Mikrocontroller mit ihrer maximalen Frequenz betreiben zu können. Dieses Kapitel ersetzt den alten Anhang B.

Wichtig

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

Hinweis

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

Teil II

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

Kapitel 6 befasst sich mit der Nutzung der *General Purpose Inputs/Outputs* (*GPIO*). Der Fokus liegt hier zunächst auf ihrer Nutzung als digitale Ausgänge zur Ansteuerung externer Komponenten. Sie bieten sehr vielfältige Konfigurationsmöglichkeiten, sodass ein großer Teil den Registern der GPIOs gewidmet ist. In zwei praktischen Beispielen werden wir die in Kapitel 2 erstellte CMSIS-Bibliothek einsetzen.

In **Kapitel 7** stelle ich Ihnen verschiedene Techniken zur Abfrage externer Komponenten vor. Hierbei handelt es sich um die Techniken *Polling*, *Interrupts* und *Exceptions*. Sie lernen Gründe dafür kennen, warum Polling keine gute Lösung darstellt und weshalb Interrupts zu bevorzugen sind. In diesem Kapitel werden Sie darüber hinaus lernen, wie Sie die GPIOs durch den Einsatz sogenannter externer Interrupts als Inputs für digitale Signale nutzen können.

In **Kapitel 8** lernen Sie abschließend die sogenannten alternativen Funktionen der GPIO-Pins kennen. Standardmäßig werden die GPIOs für die Ein- bzw. Ausgabe digitaler Größen verwendet. Es gibt aber auch Komponenten, die zur Verarbeitung analoger Größen dienen oder mittels serieller Schnittstellen für den Datenaustausch zwischen verschiedenen Geräten genutzt werden. Um die Anzahl der Anschlüsse des Mikrocontrollers – und damit die Produktionskosten – so gering wie möglich zu halten, können die GPIOs so konfiguriert werden, dass auch die Verarbeitung anderer digitaler und nichtdigitaler Signale möglich ist: Zu diesem Zweck verwenden alle Cortex-M-Hersteller die *alternativen Funktionen* (die ich im Verlauf des Buches auch als *AF* bezeichne).

Beginnend mit **Kapitel 9** werden Sie nach und nach die verschiedenen *Timer* und ihre Einsatzmöglichkeiten kennenlernen. Mit Timern sind besonders die STM-Mikrocontroller reichhaltig ausgestattet. Kapitel 9 befasst sich in verschiedenen Beispielen mit dem SysTick-Timer. Ich beginne hier ganz bewusst mit schlechten Beispielen, da Sie diese in vielen Onlinetutorials ebenfalls finden. In mehreren Schritten werden die schlechten Beispiele stetig verbessert, wobei ich Sie immer auf die besonderen Vorteile der neuen Techniken hinweise.

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

Kapitel 10 befasst sich im Anschluss mit den Funktionen und der Programmierung der *Basic Timer*. Neben dem SysTick-Timer sind sie die einfachsten Varianten der verfügbaren Timer.

In **Kapitel 11** werden Sie dann an die *General-Purpose Timer* (GP-Timer) herangeführt. Sie werden bereits hier feststellen, dass Sie Konzepte, die Sie in Kapitel 10 kennengelernt haben, sehr einfach wiederverwenden können. Kapitel 11 bietet zudem eine Einführung in die sogenannte *Pulsweitenmodulation* (*PWM*), da sie eine der wesentlichen Erweiterungen der GP-Timer im Vergleich zu den Basic Timern darstellt. Hier lernen Sie dann auch die sehr mächtigen und hilfreichen Input-Capture- bzw. Output-Compare-Funktionen kennen.

Kapitel 12 schließt mit der Vorstellung der *Advanced-Control-Timer* den Überblick über Timer ab. Alles, was Sie in den bisherigen Kapiteln über Timer gelernt haben, können Sie hier auf die gleiche Weise nutzen. Zusätzlich lernen Sie aber auch neue Dinge kennen, die besonders bei der Ansteuerung von elektrischen Antrieben unter Einsatz der sogenannten H-Brücken von elementarer Bedeutung sind.

Kapitel 13 und **14** befassen sich zum Abschluss von Teil II mit Komponenten, die es uns ermöglichen, die reale analoge Welt mit digitalen Geräten zu erfassen bzw. zu beeinflussen. Die Themen dieser beiden Kapitel sind daher *Digital-Analog-Wandler* (Kapitel 13) und *Analog-Digital-Wandler* (Kapitel 14).

Teil III

Ein wesentlicher Bestandteil des Einsatzes technischer Geräte besteht in der Übermittlung von Daten zwischen verschiedenen Geräten und/oder Komponenten. Während beispielsweise Drucker (und teilweise auch Scanner) früher oftmals mit parallelen Schnittstellen ausgestattet wurden, haben diese heutzutage praktisch keine Relevanz mehr: Sie wurden nahezu vollständig durch serielle Schnittstellen ersetzt. Sie glauben mir nicht? Dann möchte ich Sie auf zwei der wichtigsten seriellen Schnittstellen aufmerksam machen: USB und Ethernet! Waren serielle Schnitt-

stellen früher relativ langsam – Übertragungsraten von wenig mehr als 115 kBit waren eher die Ausnahme als die Regel –, so übertragen moderne serielle Schnittstellen Daten mit einer Übertragungsrate von mehreren Hundert MBit (USB) bis hin zu GBit in den neuesten Ethernet-Entwicklungen.

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

Kapitel 15 bietet zunächst eine grundlegende Einführung in das Thema *serielle Kommunikation*. Hier werden einige Grundlagen vermittelt, die zum Verständnis der technischen Umsetzung elementar sind. Der große Vorteil serieller Schnittstellen besteht darin, dass der Datenaustausch häufig über nur eine oder maximal zwei Datenleitungen erfolgt.

Mit der Besprechung und der Anwendung von *UARTs/USARTs* in **Kapitel 16** werden Schnittstellen behandelt, die überwiegend für die Kommunikation zwischen verschiedenen Geräten eingesetzt werden. Mit ihnen können Daten auch über größere Entfernungen, z.B. mehrere Hundert Meter, übertragen werden.

Über derart große Distanzen müssen aber längst nicht alle Daten transportiert werden: Sehr häufig ist es völlig ausreichend, wenn Komponenten Daten nur über Distanzen von wenigen Zentimetern austauschen. Auch zu diesem Zweck wurden serielle Schnittstellen entwickelt, von denen in **Kapitel 17** die Schnittstelle *Inter-Integrated Circuit* (*I²C*) vorgestellt wird.

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

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

Teil IV

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

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

In **Kapitel 19** werde ich Sie daher mit dem sogenannten *Direct Memory Access* (*DMA*) vertraut machen. Diese Technik wird besonders häufig beim Austausch größerer Datenmengen verwendet (z.B. beim Abspielen von Musikdateien), weil sie den Mikrocontrollerkern vom Laden, Bearbeiten und Transferieren der Daten entlastet: Der Mikrocontroller kann in der gleichen Zeit für andere wichtige Aufnahmen parallel weiter genutzt werden.

Im letzten Kapitel (**Kapitel 20**) gehe ich mit den sogenannten *Watchdog-Timern* (*WD*) noch auf eine besondere Timer-Klasse ein, die für spezielle Aufgaben beim sicheren Betrieb von Anwendungen verwendet werden. Sie werden dann eingesetzt, wenn ein Gerät – dies kann auch eine interne Komponente des Mikrocontrollers sein – nicht ausreichend schnell arbeitet (z.B. weil Daten nicht rechtzeitig zur Verfügung stehen oder weil ein solches Gerät defekt ist). Im ordnungsgemäßen Betrieb werden WDs ständig neu geladen und dürfen niemals ablaufen. Läuft ein WD dennoch ab, weil eine Komponente »den Betrieb aufhält«, kann sein Auslösen dazu genutzt werden, die Maschine in einen sicheren Zustand zu überführen.

Anhänge

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

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

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

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

Einsatz von Bibliotheken?

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

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

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

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

Hinweis

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

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

Hinweis

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

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

Hinweis

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

Entwicklungsumgebungen

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

Hinweis

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

Tipp

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

Hinweis zu den Prozessorregistern

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

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

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

Support

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

Anmerkungen des Autors

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

Hinweis

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

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

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