Erstellen der CMSIS-Bibliothek

Im Vergleich zur gedruckten Version meines Buches „STM32 Arm-Mikrocontroller programmieren für Embedded Systeme“ haben sich im Laufe der Zeit – seit der Erstellung des Manuskripts sind immerhin bereits ca. 18 Monate vergangen – einige Änderungen ergeben. Zunächst hat STMicroelectronics die Entwicklungsumgebung (IDE) aktualisiert: Wurde für das Buch noch mit der Version 1.6 gearbeitet, so liegt inzwischen die Version 1.7.0 vor. Obwohl der Versionssprung gar nicht so groß erscheint, können mit der aktuellen Version der IDE erstellte Projekte nicht mit älteren Versionen bearbeitet werden. Ich habe zudem – noch auf Basis der 1.6er-Version – meine eigene Vorgehensweise optimiert, sodass die im Buch beschriebene Vorgehensweise inzwischen veraltet ist: Die neue Vorgehensweise erscheint mir etwas leichter nachvollziehbar.

Vorbedingungen

- Download und Installation der Entwicklungsumgebung von der STM-Webseite.
- Download des MCU-Packages und eventuell verfügbarer Patches.

Vorbereiten der MCU-Packages

Nach erfolgtem Herunterladen der MCU-Packages (Stand 21.11.2021: STM32 Cube MCU Package for STM32F4 series 1.26.0 sowie dem Patch 1.26.2) gehen Sie bitte folgendermaßen vor:

- Entpacken Sie die Version 1.26.0 in ein temporäres Verzeichnis, z.B. C:\temp.
- Aktualisieren Sie den Inhalt des temporären Verzeichnisses durch Anwenden des Patches 1.26.2 (d.h. entpacken Sie die Patch-Datei in das eben erstellte Verzeichnis und bestätigen Sie das Überschreiben aller Dateien).
- Für die Beispiele des Buches wird nur der geringste Teil des so erzeugten temporären Firmware-Pakets benötigt.

Erzeugen des CMSIS-Projekts

- Der erste Schritt, der beim Erstellen eines neuen Projekts – prinzipiell können Sie zwischen den beiden Optionen *Statische Bibliothek* (Static Library) oder *Ausführbares Projekt* (Executable) wählen – besteht in der Auswahl des verwendeten Mikrocontrollers. Klicken Sie daher zunächst im *File*-Menü den Eintrag *New* und dann *STM32 Project* an. Im hierauf angezeigten sogenannten *Target Selector* können Sie den zu verwendenden Mikrocontroller auswählen. Auf eine Abbildung des Target Selectors verzichte ich, da dieser Dialog selbsterklärend ist.
- Der eben angesprochene *Target Selector* dient zur Auswahl der Ziel-MCU (MCU = Microcontroller Unit). Hierbei handelt es sich um einen integrierten Bestandteil der Entwicklungsumgebung STM32CubeIDE. Die Vielzahl der verfügbaren ST-Mikrocontroller, die zudem in vielen verschiedenen Ausprägungen erhältlich sind, kann etwas verwirrend sein. Um die Auswahl zu vereinfachen, reicht es im ersten Schritt aus, im Editierfeld *Commercial Part Number* (oben links) einfach die Nummer der MCU anzugeben: Beispielhaft habe ich hier einfach nur ‚446‘ eingegeben, im rechten Teil des Dialogs wird dann eine Liste mit derzeit 33 Varianten der STM32F446-MCU angezeigt. Hier müssen Sie natürlich die korrekte MCU auswählen. Die Auswahl schließen Sie dann einfach durch das Anklicken der mit *Finish* beschrifteten Schaltfläche.

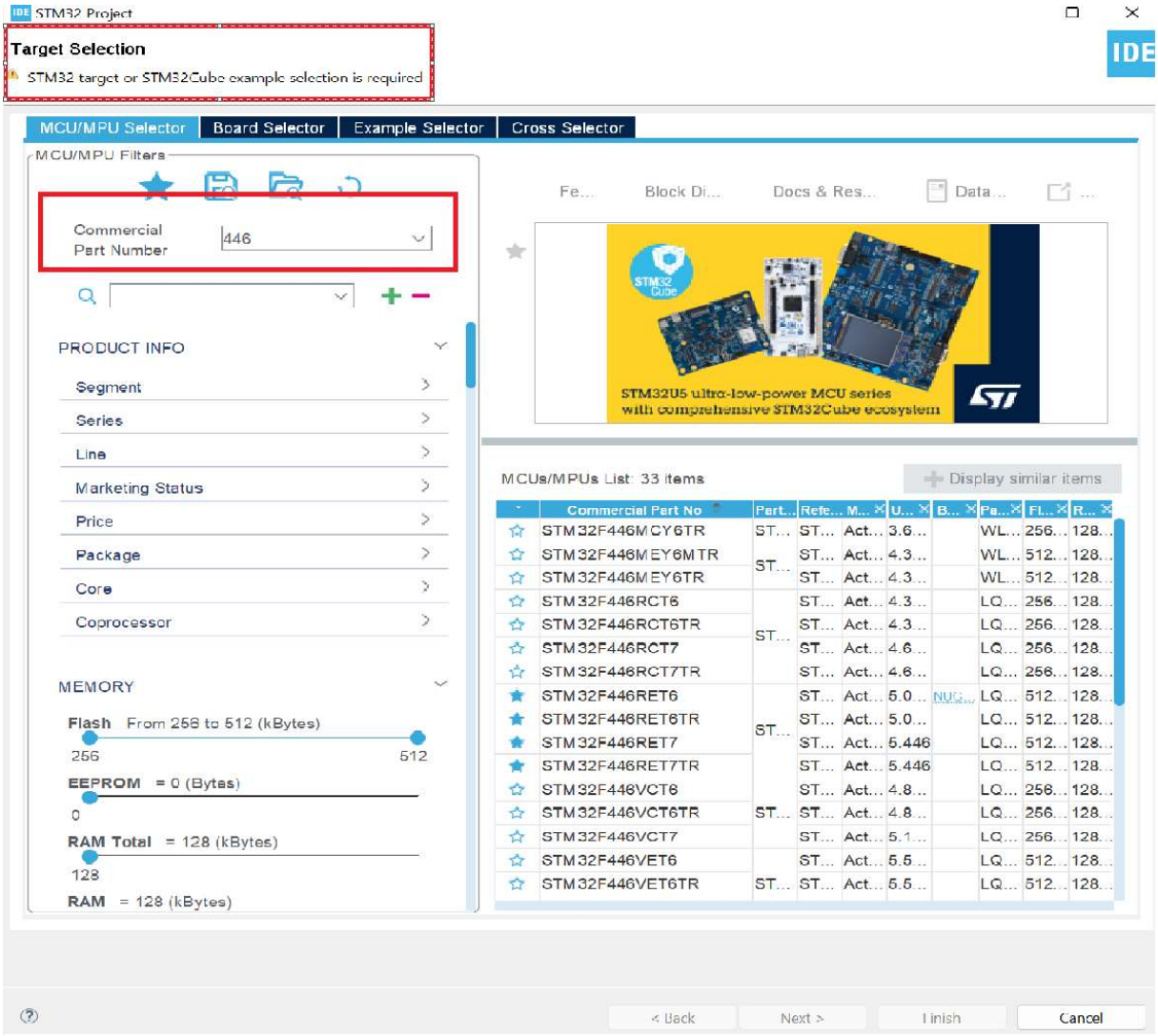

Abbildung 1: Auswahl der Ziel-MCU mit dem Target Selector

- Nach der Auswahl des Mikrocontrollers benennen Sie im nächsten Dialog das neue Projekt. Da in diesem Text die Erstellung der CMSIS-Bibliothek beschrieben wird, sollte der Projektname *CMSIS* lauten. Geben Sie ihn ein und beachten Sie auch die weiteren Einstellungen, die Abbildung 1 zeigt.

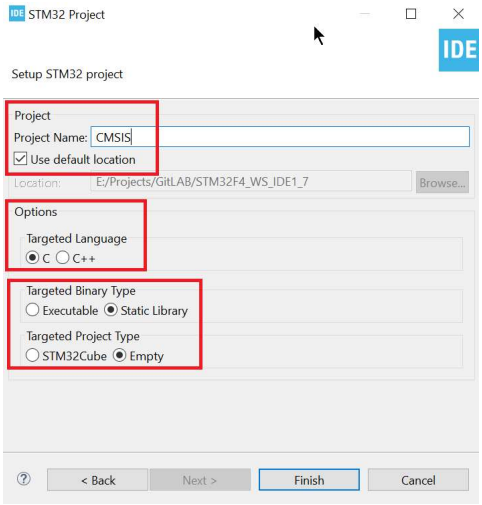

Abbildung 2: Erstellen einer statischen Bibliothek

Nach Betätigen der Schaltfläche Finish sollte die IDE Folgendes anzeigen (Abbildung 2):

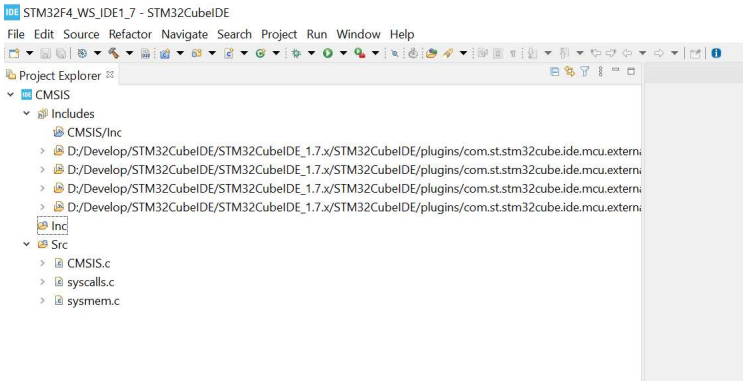

Abbildung 3: Ansicht des neuen (Bibliotheks-)Projekts im *Project Explorer* der IDE.

Die gezeigten Pfade zu den verschiedenen Dateien werden bei Ihnen vermutlich anders lauten. Wichtig ist hier nur der grundsätzliche Aufbau!

- Die Verzeichnisse *Inc* und *Src* werden im weiteren Verlauf durch andere Dateien ersetzt und können aus dem Projekt gelöscht werden.
- Kopieren Sie nun aus dem vorab erstellten temporären Verzeichnis die Verzeichnisse **Include** (es enthält Headerdateien, die von ARM erstellt wurden) und **Device** (der Inhalt wurde von STM erstellt) in das Projekt-Verzeichnis in Ihrem Workspace und aktualisieren Sie in der IDE die Ansicht durch Drücken der Taste F5. Die Projektstruktur sollte im Anschluss folgendermaßen aussehen (Abbildung 3):

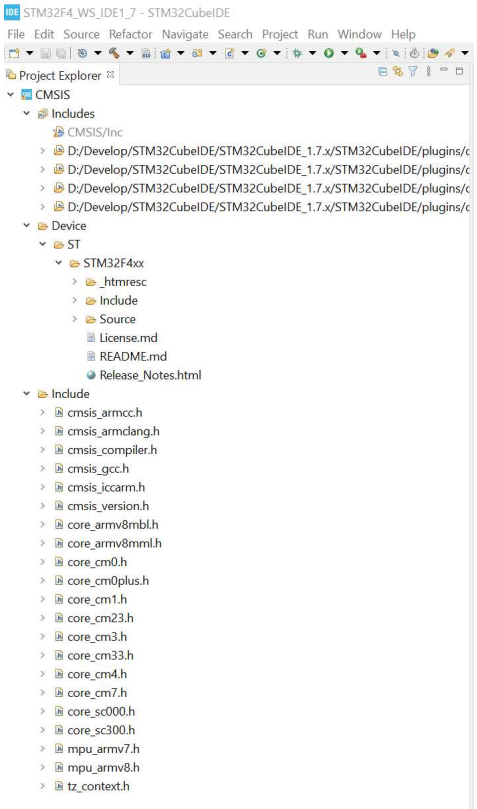

Abbildung 4: Projektstruktur nach dem Kopieren

- Suchen Sie im Verzeichnis **Device\ST\STM32F4xx\Include** nun die Datei **stm32f4xx.h** und öffnen Sie sie durch einen Doppelklick im Editor der Entwicklungsumgebung.

- Im Bereich der Zeilen 59 bis 94 dieser Datei finden Sie eine Menge auskommentierte mit #define beginnende Zeilen. Entfernen Sie hier die Kommentarzeichen um den von Ihnen verwendeten Mikrocontroller. Abbildung 4 zeigt, wie dies für den STM32F446 aussieht:

```
/* Uncomment the line below according to the target STM32 device used in your
   application
 */
#if !defined (STM32F405xx) && !defined (STM32F415xx) && !defined (STM32F407xx) && !defined (STM32F417xx) && \
    !defined (STM32F427xx) && !defined (STM32F437xx) && !defined (STM32F429xx) && !defined (STM32F439xx) && \
    !defined (STM32F401xC) && !defined (STM32F401xE) && !defined (STM32F410Tx) && !defined (STM32F410Cx) && \
    !defined (STM32F410Rx) && !defined (STM32F411xE) && !defined (STM32F446xx) && !defined (STM32F469xx) && \
    !defined (STM32F479xx) && !defined (STM32F412Cx) && !defined (STM32F412Rx) && !defined (STM32F412Vx) && \
    !defined (STM32F412Zx) && !defined (STM32F413xx) && !defined (STM32F423xx)
  /* #define STM32F405xx */   /*!< STM32F405RG, STM32F405VG and STM32F405ZG Devices */
  /* #define STM32F415xx */   /*!< STM32F415RG, STM32F415VG and STM32F415ZG Devices */
  /* #define STM32F407xx */   /*!< STM32F407VG, STM32F407VE, STM32F407ZG, STM32F407ZE, STM32F407IG  and STM32F407IE Devices */
  /* #define STM32F417xx */   /*!< STM32F417VG, STM32F417VE, STM32F417ZG, STM32F417ZE, STM32F417IG and STM32F417IE Devices */
  /* #define STM32F427xx */   /*!< STM32F427VG, STM32F427VI, STM32F427ZG, STM32F427ZI, STM32F427IG and STM32F427II Devices */
  /* #define STM32F437xx */   /*!< STM32F437VG, STM32F437VI, STM32F437ZG, STM32F437ZI, STM32F437IG and STM32F437II Devices */
  /* #define STM32F429xx */   /*!< STM32F429VG, STM32F429VI, STM32F429ZG, STM32F429ZI, STM32F429BG, STM32F429BI, STM32F429NG,
                                     STM32F439NI, STM32F429IG  and STM32F429II Devices */
  /* #define STM32F439xx */   /*!< STM32F439VG, STM32F439VI, STM32F439ZG, STM32F439ZI, STM32F439BG, STM32F439BI, STM32F439NG,
                                     STM32F439NI, STM32F439IG and STM32F439II Devices */
  /* #define STM32F401xC */   /*!< STM32F401CB, STM32F401CC, STM32F401RB, STM32F401RC, STM32F401VB and STM32F401VC Devices */
  /* #define STM32F401xE */   /*!< STM32F401CD, STM32F401RD, STM32F401VD, STM32F401CE, STM32F401RE and STM32F401VE Devices */
  /* #define STM32F410Tx */   /*!< STM32F410T8 and STM32F410TB Devices */
  /* #define STM32F410Cx */   /*!< STM32F410C8 and STM32F410CB Devices */
  /* #define STM32F410Rx */   /*!< STM32F410R8 and STM32F410RB Devices */
  /* #define STM32F411xE */   /*!< STM32F411CC, STM32F411RC, STM32F411VC, STM32F411CE, STM32F411RE and STM32F411VE Devices */
  #define STM32F446xx          /*!< STM32F446MC, STM32F446ME, STM32F446RC, STM32F446RE, STM32F446VC, STM32F446VE, STM32F446ZC,
                                     and STM32F446ZE Devices */
  /* #define STM32F469xx */   /*!< STM32F469AI, STM32F469II, STM32F469BI, STM32F469NI, STM32F469AG, STM32F469IG, STM32F469BG,
                                     STM32F469NG, STM32F469AE, STM32F469IE, STM32F469BE and STM32F469NE Devices */
  /* #define STM32F479xx */   /*!< STM32F479AI, STM32F479II, STM32F479BI, STM32F479NI, STM32F479AG, STM32F479IG, STM32F479BG
                                     and STM32F479NG Devices */
  /* #define STM32F412Cx */   /*!< STM32F412CEU and STM32F412CGU Devices */
  /* #define STM32F412Zx */   /*!< STM32F412ZET, STM32F412ZGT, STM32F412ZEJ and STM32F412ZGJ Devices */
  /* #define STM32F412Vx */   /*!< STM32F412VET, STM32F412VGT, STM32F412VEH and STM32F412VGH Devices */
  /* #define STM32F412Rx */   /*!< STM32F412RET, STM32F412RGT, STM32F412REY and STM32F412RGY Devices */
  /* #define STM32F413xx */   /*!< STM32F413CH, STM32F413MH, STM32F413RH, STM32F413VH, STM32F413ZH, STM32F413CG, STM32F413MG,
                                     STM32F413RG, STM32F413VG and STM32F413ZG Devices */
  /* #define STM32F423xx */   /*!< STM32F423CH, STM32F423RH, STM32F423VH and STM32F423ZH Devices */
#endif
```

Abbildung 5: Entfernen der Kommentarzeichen für den gewünschten Mikrocontroller

Erweitern Sie nun die Anzeige des Verzeichnisses **Device\ST\STM32F4xx\Source** und löschen Sie hier die Verzeichnisse **arm** und **iar** (die Verzeichnisse enthalten Dateien, die von Fremd-Compilern benötigt werden. Wir verwenden aber nur **gcc**).

Projektspezifische Einstellungen

Mit den bisher durchgeführten Arbeiten lässt sich der Sourcecode der Bibliothek aber noch nicht kompilieren. Zunächst muss dem CMSIS-Projekt noch mitgeteilt werden, wo es die Headerdateien findet. Klicken Sie nun mit der rechten Maustaste auf den Projektnamen (hier also CMSIS) und anschließend auf **Properties**. Es öffnet sich der in Abbildung 5 gezeigte Dialog:

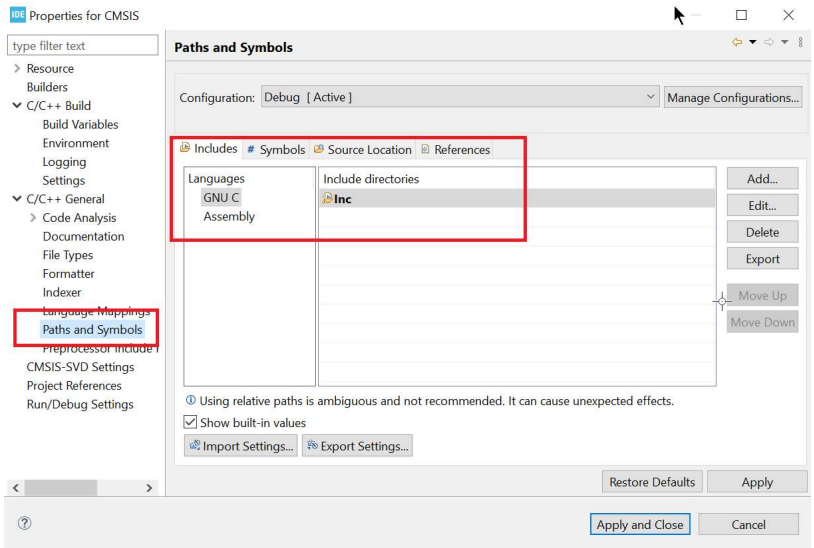

Abbildung 6: Properties-Einstellungen des CMSIS-Projekts

Unter **C/C++ General** finden Sie den Eintrag **Paths and Symbols**. Im rechten Teil des Dialogs sehen Sie für den Compiler (GNU C), dass hier standardmäßig als Pfad das nicht mehr vorhandene Verzeichnis **Inc** gezeigt wird. Markieren Sie dieses Verzeichnis durch einen Klick mit der linken Maustaste und löschen Sie es durch Anklicken der Schaltfläche **Delete**.

Nun müssen Sie allerdings die Pfade zu den verwendeten Headerdateien eintragen. Im gleichen Dialog klicken Sie daher auf die Schaltfläche **Add…**. Erweitern Sie hier die Anzeige des Projektnamens (also CMSIS): Wichtig sind hier die Einträge **Device** und **Include** (dies sind die Namen der Verzeichnisse, die die Quelldateien des MCU-Packages enthalten).

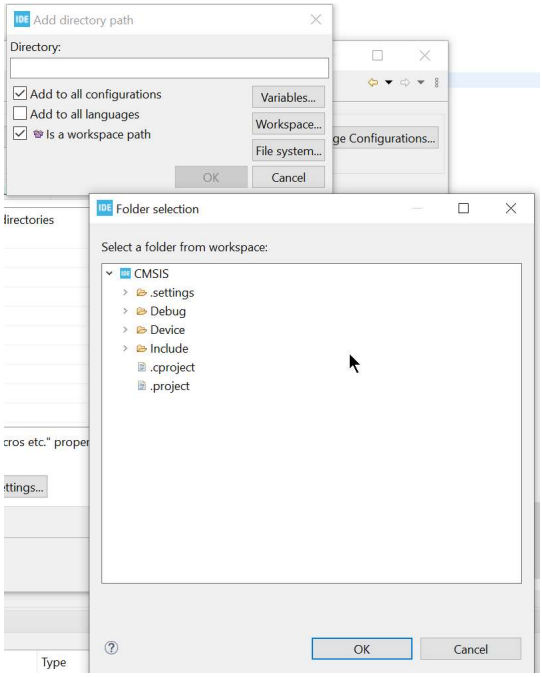

Abbildung 7: Hinzufügen der Headerdateien

Klicken Sie hier zunächst den Eintrag **Include** an und bestätigen Sie durch Anklicken der Schaltfläche **OK.** Ein weiterer Klick auf OK im darüberliegenden Dialog fügt das Verzeichnis CMSIS\Include zu den Projekteinstellungen hinzu. Wiederholen Sie diese Aktion für das Projektverzeichnis **CMSIS\Device\ST\STM32F4xx\Include**. Haben Sie dies erledigt, klicken Sie im Ausgangsdialog die Schaltfläche **Apply and Close** an.

Die IDE weist Sie darauf hin, dass die Änderungen erst dann im Projekt verwendet werden können, wenn der Index neu aufgebaut wird. Dieser Index ermöglicht später den Zugriff auf die Einträge der Headerdateien durch Drücken der Tastenkombination **STRG** + **LEER**. Der Vorteil dieser sogenannten ‚intelligenten Worterweiterung' besteht darin, dass nur Fragmente eines Bezeichners eingegeben werden müssen: Er wird dann automatisch durch die IDE erkannt und in den Quelltext eingefügt. Beginnen mehrere Bezeichner mit der gleichen Zeichenfolge, so werden alle Bezeichner angezeigt, und Sie können dann durch einen Doppelklick den gewünschten Eintrag automatisch in den Quelltext einfügen.

Generieren der Bibliothek

Wenn Sie alle Schritte durchgeführt haben, können Sie das Projekt kompilieren. Ich verwende hierzu immer die Tastenkombination **STRG** + **B**. Die fertige Bibliothek finden Sie anschließend im Verzeichnis **Debug** bzw. **Release** des Projektverzeichnisses.

Noch ein Hinweis

In der Datei **system_stm32f4xx.c** (sie wird von STM mitgeliefert) finden Sie in den Zeilen 51 bis 57 den folgenden Sourcecode:

```
51 #if !defined  (HSE_VALUE)
52   #define HSE_VALUE    ((uint32_t)25000000) /*!< Default value of the External oscillator in Hz */
53 #endif /* HSE_VALUE */
54
55 #if !defined  (HSI_VALUE)
56   #define HSI_VALUE    ((uint32_t)16000000) /*!< Value of the Internal oscillator in Hz*/
57 #endif /* HSI_VALUE */
58
```

Abbildung 7: Einstellung der HSE-Frequenz (HSE = High Speed External)

Standardmäßig werden die NUCLEO-Boards von STM mit einer Frequenz in Höhe von 16 MHz betrieben, was auch vollkommen in Ordnung ist. Dieser Wert wird auch als HSI_VALUE (HSI = High Speed Internal) bezeichnet. HSE_VALUE ist hingegen die Frequenz, wenn Sie einen externen Quarz verwenden: Er darf mit maximal 25 MHz schwingen. Die NUCLEO-Boards enthalten aber auf dem Debug-Teil der Platine „nur“ einen 8MHz-Quarz. Wenn Sie diesen verwenden, so müssen Sie in Zeile 52 den Wert von 25000000 nach 8000000 ändern. Vergessen Sie dies, so stimmen die intern erzeugten Frequenzen nicht, was zu einem seltsamen Verhalten der Software führen kann.

Beachten Sie, dass der Wert von HSE_VALUE bei jedem Update der Firmware wieder auf 25000000 gesetzt wird. Überprüfen Sie den Wert von HSE_VALUE nach jedem Firmware-Update.