Bausteininitialisierung – Mögliche Methoden und Tools

Ingo Pohle, MicroConsult

Dieser Vortrag erläutert folgende **Aspekte**, die bei der **Initialisierung von Mikrocontroller-Architekturen** wichtig sind:

Welche Komponenten einer Mikrocontroller Architektur müssen initialisiert werden?
Welche Initialisierungs-Methoden gibt es für Mikrocontroller?
Welche Initialisierungs-Tools für Mikrocontroller können verwendet werden?
Resümee

Folgende Komponenten eines Mikrocontrollers müssen initialisiert werden:

- Core Register: z. B. Core Control Register, Stack Pointer, etc.
- Core Peripherals: z. B. Cache, SPRAM, etc.
- System-Takt (z. B. PLL, Clock-Tree)
- Interrupt System
- DMA Controller
- Memory Management Unit MMU
- Memory Protection Unit MPU
- Peripherie:
  - Ports
  - Timer
  - Serielle Schnittstellen
  - Capture/Compare Einheiten
  - PWM Einheiten
  - A/D-Wandler
  - Externer Buscontroller
  - etc.

Hinweis:
Die **Initialisierung** von **CPU**, **System-Komponenten** (Clock-System), der **Memory-Interfaces** und RAM-Speicher Setup (Laden von Initialisierungs-Werten in die RAM-Speicher) wird in der Regel im **Start-Up Programmi** durchgeführt. Alternativ wird eine Initialisierungsfunktion „**System Init**“, die vom Bausteinhersteller zur Verfügung gestellt wird, eingesetzt.

Methoden für die Initialisierung von Mikrocontrollern:

Individuelle Initialisierung eines Mikrocontrollers

Einsatz von Initialisierungs-Bibliothek(en)

Einsatz von Initialisierungs-Tools

Individuelle Initialisierung eines Mikrocontrollers

Die Methode der individuellen Initialisierung hat folgende Aspekte für ein Projekt:
Für jedes Projekt wird die Initialisierung neu erstellt.

Die Wiederverwendbarkeit der Initialisierungssoftware ist nicht, oder nur bedingt möglich.

Eine nicht vollständige Initialisierung kann zu Problemen, einem Sicherheitsrisiko,
bzw. Fehlverhalten des Systems führen.

Vorteile:

Ressourcensparend

Exakt auf die Erfüllung der funktionalen und nicht-funktionalen Anforderungen abgestimmt.

Nachteile:

Höherer Aufwand bei der Entwicklung und dem Test der Software.

Der Initialisierungs-Code ist nicht wiederverwendbar (nicht portierbar).

Diese Methode ist nicht geeignet für sicherheitsrelevante Applikationen.

Einsatz von Initialisierungs-Bibliothek(en) für die Initialisierung eines Mikrocontrollers

Die Methode der Initialisierung über eine Bibliothek hat folgende Aspekte für ein Projekt:

Der Initialisierungscode wird z. B. von einem Baustein- oder Tool-Hersteller erstellt.

Für die Initialisierung eines Mikrocontrollers, bzw. dessen Komponenten werden Bibliotheksroutinen verwendet:

Diese werden aus einer Bibliothek über ein standardisiertes Interface (z. B. für Cortex-M:

CMSIS Abstraktion Layer für die Initialisierung von Core und Core Peripherie und die Herstellerspezifischen Peripherien) angesprochen.

Der Code ist pre-compiliert und vom Hersteller getestet.

Die einzelnen (System-)relevanten Parameteroptionen sind dokumentiert und werden meist in Form von Symbolen verwendet. Diese „verbalen“ Parameter (Symbol-Defines in Header Files – „datei.h“) sind leichter verständlich, fehlerhafte, oder fälschlicherweise verwendete Parameter werden vermieden.

Eine für die richtige Funktion der Hardwarekomponenten **erforderliche Parameterreihenfolge** wird stets eingehalten.

Hinweis: Meist bieten die Hersteller der Initialisierungsbibliotheken zusätzlich noch **hardwarenahe Treiber** (Low-Level Treiber Software) und einen **Hardware-Abstraktions-Support** für den Betrieb der Peripherie an. Durch die Hardware-Abstraktion kann die Initialisierung von Hardware aus der Hardware-unabhängigen Softwareschicht erfolgen.

Vorteile:

Eine schnellere Erstellung des Initialisierungs-Codes ist möglich, durch den Einsatz von wiederverwendbarem Code (portierbarer Code). Dadurch wird Zeit bei der Entwicklung und beim Test eingespart.

Der Code ist: übersichtlicher, leichter lesbar, erweiterbar und änderbar,

Fehlende Anwendung von Parametern wird vermieden. Es werden die möglichen Wertebereiche der Parameter geprüft. Alle relevanten Parameter in der Schnittstelle werden von einer Initialisierungsfunktion angefordert.

Eine fehlerhafte Reihenfolge bei der Initialisierung wird vermieden.

Die Funktionalität der Bibliotheksroutinen muss nur einmal getestet werden.

Nachteile:

Der Aufwand bezüglich Code-Größe und RAM-Speicherbedarf steigt durch den Einsatz von Initialisierungsstrukturen.

Höhere Laufzeit bei der Abarbeitung wird benötigt.

Es können Kosten für den Erwerb der Initialisierungs-Software entstehen.Bibliotheken erfüllen die eigenen Programmierrichtlinien/Qualitätsstandard unter Umständen nicht.

Beispiele für die CMSIS Initialisierungs-Bibliotheken (ARM® **C**ortex™ Microcontroller Software Interface Standard - vendor independent library standard) für die ARM Cortex-M Architekturen:

STM CMSIS Bibliothek für Bausteine der ARM® Cortex™ M-Serie von STMicroelectronics:
http://www.st.com/stonline/products/support/micro/files/stm32f10x_stdperiph_lib.zip

Bibliothek für Bausteine der ARM® Cortex™ M-Serie von:

NXP CMSIS Bibliothek für:
LPC11xx ; siehe: http://ics.nxp.com/support/lpcxpresso/

LPC175x/LPC176x; siehe: http://www.lpcware.com/content/nxpfile/lpc175x6x-cmsis- compliant-standard-peripheral-firmware-driver-library-keil-iar-gnu

LPC43xx; siehe: http://www.lpcware.com/content/nxpfile/lpc4350apdlzip

TI CMSIS Bibliothek für:
TI Stellaris® MCUs CMSIS_LM3S: http://www.ti.com/tool/cmsis_lm3s

Hinweis: Generelle Informationen zum ARM® Cortex™ Microcontroller Software Interface Standard **CMSIS** unter:
http://www.arm.com/products/processors/cortex-m/cortex-microcontroller-software-interface-standard.php

Einsatz von Initialisierungs-Tools für die Initialisierung eines Mikrocontrollers

Die Initialisierung über spezielle Initialisierungs-Tools wird in zwei unterschiedlichen Arten angeboten:

Peripheriemodul Initialisierung: jedes Modul (z.B. Timer) wird einzeln initialisiert

Applikations-Initialisierung (Initialisierung einer Funktionalität, bestehend aus mehreren Modulen, z. B. einer Motoransteuerung)

Aspekte der Initialisierungsmethode für ein Projekt:

Die Core Register, die Core Peripherie und die Baustein-spezifischen Peripherien stehen für die Initialisierung interaktiv zur Verfügung (z. B. graphische Bedienoberfläche).

Als Ergebnis der Anwender Auswahl von Modes und Steuerungsvorgaben werden Initialisierungsdateien generiert. Dieser Generierungsprozess ist reproduzierbar, d. h. bei gleicher Mode- und Parameterauswahl wird immer der gleiche Code generiert.

Vorteile:

Es können typischerweise neben den Initialisierungsdateien auch noch Low Level Treiber Dateien aus einer Baustein-spezifischen Bibliothek generiert werden.

Der generierte Code ist ANSI-C konform.

Das Tool überwacht:

verfügbare Ressourcen wie z. B. verfügbare Ports (Doppelbelegung wird vermieden)

maximale und minimale Systemparameter (z. B. mögliche Übertragungsraten)

das ein Peripherie-Modul erst initialisiert werden darf, wenn es aktiviert worden ist (z. B. der Takt für die Peripheriefunktionalität auch freigeschalten ist).

Nachteile:

Das Ergebnis von Initialisierungs-Tools kann in einer anderen Programmiersprache (z.B. C anstatt C++) generiert sein.

Die neueste Generation von Initialisierungs-Tools bietet die Möglichkeit der Applikations-Initialisierung.

Der generierte Code entspricht auch hier möglicherweise nicht den firmeninternen Programmierrichtlinien bzw. Qualitätsvorgaben.

Es ergibt sich eine weitere Toolabhängigkeit (Toolintegration, Tool-Zertifizierung für Entwicklungen im sicherheitskritischen Bereich).

Keine Gewährleistung des Anbieters bei nicht zertifizierten Initialisierungs-Tools.

Beispiele für Initialisierungs-Tools:

DAvE™ 2.2
Initialisierungs-Tool für System und Peripherie-Module in Infineon 8-/16- und 32-Bit Mikrocontrollern; geeignet für Bausteine vom Typ: XC800, XE16x, XC2000 und TriCore Derivate

Processor Expert
Initialisierungs-Tool für System und Peripherie-Module in Freescale Mikrocontrollern; dieses Tool ist geeignet für die Bausteinfamilien HCS12X, ColdFire/ColdFire+, Kinetis ARM® Cortex Mikrocontroller, Qorivva

DAvE™ 3.0
Initialisierung von Applikationen - DAVE™ Apps; geeignet für Bausteine vom Typ XMC4000 (Typ: ARM™ Cortex M-4); Aktuelle Version: 3.1.2 (Beta –Version). Hierbei wird Code generiert der auf vordefinierten und getesteten SW Komponenten (DAVE Apps) basiert.
Die Konfiguration erfolgt über ein Graphisches User Interface. Für die Dokumentation und den Test des generierten Codes, bzw. die verwendeten Funktionen stehen Interfacebeschreibungen (APIs) zur Verfügung.

Resümee:

Letzen Endes ist die Auswahl der Initialisierungs-Methode sehr stark abhängig von der konkreten Applikation bzw. dem Umfeld, in dem diese eingesetzt wird (z. B.: wird überhaupt eine Bibliothek, bzw. ein Initialisierungs-Tool für den gewählten Mikrocontroller zur Verfügung gestellt). Die Methode, die sich am besten eignet hängt insbesondere von den zu erfüllenden Anforderungen, wie z. B. den nicht-funktionalen Qualitätsmerkmalen wieder:

Änderbarkeit: Analysierbarkeit, Modifizierbarkeit, Stabilität

Benutzbarkeit: Bedienbarkeit, Erlernbarkeit, Verständlichkeit

Effizienz: Zeitverhalten, Verbrauchsverhalten

Funktionalität: Angemessenheit, Ordnungsmäßigkeit, Richtigkeit, Sicherheit

Übertragbarkeit: Anpassbarkeit, Austauschbarkeit, Installierbarkeit, Konformität

Zuverlässigkeit: Fehlertoleranz, Reife, Wiederherstellbarkeit

Der schnelle Weg zum Evaluierungs-Test
Die Toolgestützte Initialisierung eignet sich hervorragend um in sehr kurzer Zeit:
Peripherien zu konfigurieren und erste Tests von Applikationsteilen durchzuführen.
Damit lassen sich schnell z.B. Prototypen evaluieren und Machbarkeitsstudien durchführen.

Die Ergebnisse wie z. B. Initialisierungs- und Treiber-Routinen sind möglicherweise als Basis für eine eigene Treiberentwicklung und Reviews verwendbar. Ist die geforderte Qualität dieser Routinen ausreichend, können sie direkt in das Produkt einfließen.

Korrekte Initialisierungswerte und richtige Initialisierungsreihenfolge

Neben den richtigen Initialisierungswerten kommt es für das fehlerfreie Funktionieren von einzelnen Peripherien und vernetzt operierenden Peripherien häufig auch auf die richtige Initialisierungsreihenfolge an. Der Code aus Bibliotheken, bzw. der generierte Code berücksichtigt in der Regel genau diese

Erfordernisse. Die funktional richtige Reihenfolge bei der Einstellung von Modes und der Freigabe der Einzel- bzw. vernetzten Operation/Funktion ist maßgebend für die einwandfreie Funktion der Peripherien.

Minimierung des Entwicklungsaufwandes ermöglicht Verkürzung der Entwicklungszeit

Die Einbindung von fertigen Bibliotheken und die Anwendung von Initialisierungs-Tools minimieren signifikant den Entwicklungsaufwand. Dies aber nur unter der Bedingung, dass die verwendete Bibliothek, bzw. der generierte Code auch den firmeninternen bzw. den produktspezifischen Qualitätsansprüchen gerecht werden. Unser Rat: Der Gewinn an Entwicklungszeit kann unter Umständen der oft zu kurz bemessenen Testphase eines Projektes zugeschlagen werden.

Für den Fall, dass die Initialisierungs- und Treiber-Routinen dennoch selbst entwickelt werden müssen, z.B. für den Einsatz in sicherheitskritischen Systemen, bieten die Bibliotheken dafür eine hervorragende technische Grundlage.

Zum Autor:

Dipl.-Ing. Ingo Pohle: Studium Mikroelektronik mit dem Schwerpunkt Nachrichtentechnik , 5 Jahre Entwicklungsingenieur , über 25 Jahre Erfahrung als Trainer und Coach für Mikroprozessoren, Bussysteme und Embedded-Betriebssysteme.

Kontakt:
Email: info@microconsult.de
Tel. 089 46061746