Modulare Applikationsentwicklung für ARM Cortex-M

Schneller entwickeln mit Softwarekomponenten

Johannes Bauer, ARM Germany

Moderne, auf ARM Cortex-M basierende Mikrocontroller eignen sich aufgrund ihrer hohen Rechenleistung, vielseitiger Peripheriefunktionen und geringen Energieverbrauchs für immer mehr Anforderungen. Für diese komplexen Anwendungen werden fortgeschrittene Methoden in der Softwareentwicklung benötigt, um hochwertige Produkte und kurze Entwicklungszeiten sicherzustellen. Der Ansatz von ARM für modulare Entwicklung sind standardisierte Softwarekomponenten im CMSIS-Pack-Format.

Wenn es einen Trend bei Embedded-Systemen gibt, auf den sich wohl alle einigen können, dann ist es der zu mehr Funktionalität. Wo in der Vergangenheit viele Produkte mit einfachsten 4- oder 8-bit-Mikrocontrollern auskamen, geht die Entwicklung heutzutage immer mehr in Richtung der 32-bit-Architekturen, was sich auch deutlich in der Hinwendung immer mehr Hersteller zur ARM Cortex-M-Prozessoren zeigt. Diese Mikrocontroller bieten hohe Taktfrequenzen bis zu mehrere 100 MHz sowie vielseitige Peripheriefunktionen, und dass bei konsistent geringem Energieverbrauch. Diese Eigenschaften machen die Systeme nun für deutlich mehr Anwendungsszenarien interessant.

Eines davon sind die vielbeschworenen Applikationen des „Internet of Things“, welches eine große Menge an ständig mit dem Internet verbundenen Systemen bringen wird. Speziell für den Einsatz in typischen Consumer-Geräten hat ARM mbed OS angekündigt, ein vollständiges, nicht echtzeitfähiges Betriebssystem, welches Funktionalität wie Verschlüsselung, Gerätesicherheit, Kommunikationsprotokolle und Over-the-Air-Updates abdeckt und auch eine kommandozeilenorientierte Entwicklungsumgebung bereitstellt. Doch nicht alle Anforderungen an Embedded-Systeme lassen sich damit realisieren, so dass auch weiterhin eine große Vielfalt an Echtzeit-Betriebsystemen, Middleware-Komponenten und Entwicklungswerkzeugen zu erwarten ist.

Mit der zunehmenden Komplexität der Embedded-Applikationen ist es notwendig, auch die Entwicklungsmethodik zu überdenken. Wo es bisher oft üblich war, dass ganze Systeme vollständig von Einzelpersonen entwickelt wurden, ist dies bei immer umfangreicheren Anforderungen an die Funktionalität oft nicht mehr möglich. Vorgefertigte Software-Komponenten werden daher immer wichtiger, vom Betriebsystem über Schnittstellen-Bibliotheken bis hin zu Kommunikationsprotokollen. Um damit effektiv zu arbeiten, ist ein jedoch eine Standardisierung notwendig, die sicherstellt, dass die Kompatibilität von Softwarekomponenten untereinander, aber auch mit den unterschiedlichen Mikrocontroller-Plattformen und Entwicklungswerkzeugen, geklärt ist.

Für genau diesen Zweck hat ARM das CMSIS-Pack-Format[1] im Rahmen des umfassenden Cortex Microcontroller Software Interface Standard (CMSIS)[2] entwickelt. CMSIS-Pack definiert sowohl ein Paketformat als auch einen Verteilungsmechanismus für

Softwarekomponenten und legt dabei Wert darauf, dass existierende Software leicht als sogenanntes „software pack“ angeboten werden kann. Dazu dient eine XML-basierte Beschreibungsdatei, welche unter anderem folgende Informationen maschinenlesbar bereistellt:

- Mikrocontroller-Parameter wie Prozessorkern, Speicherbereiche und Debug-Schnittstellen
- Enthaltener Code, als Quellcode oder Bibliothek
- Kompatibilität mit Prozessorkernen und Compiler-Toolchains
- Anhängigkeiten zu anderen Softwarekomponenten
- Dokumentation und Lizenzinformationen

Diese „Pack Description“-Datei mit der Endung .pdsc wird zusammen mit den anderen Dateien der Softwarekomponente in ein Zip-Archiv gepackt, welches dann die Endung .pack erhält.

Die konkrete Funktionalität sei an zwei Beispielen erklärt. Zunächst soll CMSIS-Pack dazu verwendet werden, um ein sogenanntes „Device Family Pack“ zu erstellen, welches alle Informationen enthält, die benötigt werden, damit eine Entwicklungsumgebung einen bestimmten Mikrocontroller unterstützen kann. In der ARM-Cortex-M-Welt gehören dazu Startup- und Header-Dateien, die System View Description für die Registerbeschreibung im Debugger, der Flash-Algorithmus für das Beschreiben des On-Chip-Speichers und noch einige mehr Angaben, welche alle im CMSIS-Standard beschrieben sind. Diese Informationen werden von nahezu allen Anbietern von Cortex-M-Mikrocontrollern bereitgestellt. Sie müssen nun nur noch in einer .pdsc-Datei beschrieben werden, damit die Entwicklungsumgebung Bezeichung, Typ und Konfiguration des Mikrocontrollers kennt und für den Entwickler anzeigen kann. Dabei können Dateien, die Toolchain-spezifisch sind, entsprechend gekennzeichnet werden, z.B. die meist in Assembler geschriebenen Startup-Routinen. Dies erlaubt es, mit einem Device Family Pack alle gewünschten Entwicklungsumgebungen abzudecken.

Ein zweites Beispiel sei eine Softwarekomponente wie ein HTTP-Server. Der existierende Code kann ohne Veränderung übernommen werden. In der .pdsc-Datei des Software Packs werden nun wiederum Hersteller, Bezeichnung und Typ angegeben werden sowie Informationen, welche Dateien von Entwickler angepasst werden müssen und daher mit in das Projekt kopiert werden sollten, und welche Dateien, etwa vorkompilierte Bibliotheken, nur referenziert werden müssen. Dabei können auch unterschiedliche Varianten z.B. für unterschiedliche Cortex-M-Kerne definiert werden, die dann vom Entwicklungswerkzeug automatisch selektiert werden.

Neben der Beschreibung der Softwarekomponenten definiert CMSIS-Pack auch noch einen einfachen Mechanismus zur Bereitstellung der Pakete. Dafür kann in der pdsc-Datei eine URL mit angegeben werden, welche als Quelle für Updates dient. Für den Anbieter eines Packs ergeben sich mehrere Stufen für die Veröffentlichung:

- Ausschließlich private, offline verfügbare Packs
- Private, auf Intranet-Servern verfügbare Packs für einfache Updates
- Halböffentliche, auf Webservern verfügbare Packs, die an ausgewählte Kunden weitergegeben werden

- Öffentliche Packs, die für jeden verfügbar sind.

ARM unterhält einen Index[3] aller derzeit öffentlich verfügbaren Packs.

Die offen gestaltete Definition von CMSIS-Pack erlaubt eine flexible Verwendung der Software Packs für viele Anwendungsfälle. Derzeit werden Packs eingesetzt für die geschilderten Device Family Packs, für die Verbreitung des CMSIS-Pakets, für Middleware und Bibliotheken, für Board Support Packs und für die Weitergabe von selbsterstellten Softwarekomponenten innerhalb einer Firma. Der CMSIS Pack-Standard wurde schon von vielen Halbleiterherstellern angenommen, unter anderem von Atmel, Texas Instruments und Infineon. Viele weitere planen den Einsatz oder stehen kurz davor. Auch Software-Anbieter wir Micrium und Yogitech haben schon eigene Pakete veröffentlicht.

Bei den Entwicklungswerkzeugen gibt es derzeit die Referenzimplementierung von ARM im Keil Microcontroller Development Kit (MDK) sowie Atmel Studio 7, welches Device Family Packs sowohl für die ARM- als auch die AVR-basierten Microcontroller von Atmel verwendet. ARM hat auch eine Open-Source-Implementierung von CMSIS-Pack-Support für Eclipse-Umgebungen[4] veröffentlicht, welche die Unterstützung des Formats in Eclipse-basierenden Entwicklungsumgebungen beschleunigen sollte.

Damit sind gute Voraussetzungen für die weitere Verbreitung des CMSIS-Pack-Formats gegeben. ARM ist offen für die Mitwirkung bei der Weiterentwicklung des Standards und lädt alle interessierten Parteien, egal ob Halbleiterhersteller, Softwareanbieter oder Entwickler, ein, sich einzubringen.

Quellen

[1] http://www.keil.com/pack/doc/CMSIS/Pack/html/index.html
[2] http://www.keil.com/pack/doc/CMSIS/General/html/index.html
[3] http://www.keil.com/dd2/Pack/
[4] https://github.com/ARM-software/cmsis-pack-eclipse-prebuilt

Autor

Johannes Bauer ist Produktmanager für die Mikrocontroller-Entwicklungswerkzeuge von ARM und arbeitet in der Münchner Niederlassung des britischen Unternehmens. Nach dem Studium der Informationstechnik an der Technischen Universität Chemnitz und Stationen im Support und im technischen Marketing bei National Instruments beschäftigt er sich seit 2011 mit Tools für Mikrocontroller.