Sven Salzwedel

Entwurf und Implementierung eines Bootloader-Konzepts

Programmierung eines Embedded Systems auf Basis der Texas Instruments MSP430 Mikrocontroller Familie

Diplomarbeit

BACHELOR MASTER Publishing

Salzwedel, Sven: Entwurf und Implementierung eines Bootloader-Konzepts. Programmierung eines Embedded Systems auf Basis der Texas Instruments MSP430 Mikrocontroller Familie, Hamburg, Bachelor + Master Publishing 2016
Originaltitel der Abschlussarbeit: Entwurf und Implementierung eines Bootloader-Konzepts zur Programmierung eines Embedded Systems auf Basis der Texas Instruments MSP430 Mikrocontroller Familie

Buch-ISBN: 978-3-95993-015-4
PDF-eBook-ISBN: 978-3-95993-515-9
Druck/Herstellung: Bachelor + Master Publishing, Hamburg, 2016
Zugl. Hochschule für angewandte Wissenschaften, Coburg, Deutschland, Diplomarbeit, September 2008

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.

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

Die Informationen in diesem Werk wurden mit Sorgfalt erarbeitet. Dennoch können Fehler nicht vollständig ausgeschlossen werden und die Diplomica Verlag GmbH, die Autoren oder Übersetzer übernehmen keine juristische Verantwortung oder irgendeine Haftung für evtl. verbliebene fehlerhafte Angaben und deren Folgen.

Alle Rechte vorbehalten

© Bachelor + Master Publishing, Imprint der Diplomica Verlag GmbH
Hermannstal 119k, 22119 Hamburg
http://www.bachelor-master-publishing.de, Hamburg 2016
Printed in Germany

Kurzfassung

Ein in Zukunft an Bedeutung stark zunehmender Bereich der Computernetze ist die automatisierte mobile Datenerfassung mittels Sensorknoten, welche batteriebetrieben und völlig autark in einem drahtlosen Sensornetz zu verschiedenen Zwecken eingesetzt werden können. Das Fraunhofer Institut für Integrierte Schaltungen arbeitet selbst an einer proprietären Implementierung solcher Sensorknoten und der dazugehörigen Software. Wie in den meisten Bereichen der Informatik steigt mit zunehmenden Möglichkeiten die Komplexität der Software auf den Geräten, was zwangsläufig dazu führt, dass in der Software verstärkt Fehler auftreten können.

Im Rahmen dieser Diplomarbeit wurde ein Konzept und eine Implementierung eines Bootloaders entwickelt, welcher eine Aktualisierung der Software dieser Sensorknoten ermöglicht. Dabei wurde auf die Anforderung, größtmögliche Ausfallsicherheit beim Installationsvorgang eines Firmware-Images zu gewährleisten, eingegangen. Das Konzept sieht vor, dass verschiedene Sicherungsmechanismen einen unbenutzbaren Sensorknoten verhindern sollen und mehrere Firmware-Images auf dem Sensorknoten verwaltet werden können. Die prototypische Implementierung realisiert die Teile des Konzepts, welche ohne ohne Kenntnis des endgültigen Aktualisierungsmechanismus (in Bezug auf Teilaktualisierungsverfahren und Übertragungsmethodik) implementiert werden konnten.

Abstract

A steadily growing area of computer networking is the automated capturing of data with sensornodes. Those are small computer devices which can communicate over radio in a so called sensornetwork, work completely autarkic and are powered by batteries. The Fraunhofer Institute of Integrated Circuits works on an implementation of such devices and software. As with any technology in computer science the growing complexity of the software which runs on such devices causes a higher risk of software-bugs.

A bootloader concept and implementation to update software on sensornodes was developed within the work on this diploma thesis. The most important requirement was to assure that an update does not cause any harm on the device and that it can be booted into a known state everytime. The concept includes that several techniques have to be built to fulfill this requirement and that more than one firmware image can be stored and managed on the device. The implementation, which in fact is a prototype, only includes the basic parts of concept, as not everything could be implemented.

Inhaltsverzeichnis

1 Einleitung **1**

2 Allgemeine Grundlagen **3**

2.1 Drahtlose Sensornetze 3

2.2 Bootloader 6

2.2.1 PC Bootloader 6

2.2.2 Eingebettete Systeme 7

2.3 Bestehende Bootloader-Implementierungen 7

2.3.1 Atmel AVR Mikrocontroller 8

2.3.2 Deluge Softwareverteilungssystem 10

2.4 Dateiformate 12

2.4.1 TI TXT 13

2.4.2 Intel HEX 13

3 Sensorknoten Plattform **15**

3.1 Hardware 15

3.1.1 Adressraum des MSP430 16

3.1.2 Evaluation-Board 17

3.1.3 Speichermöglichkeiten 18

3.2 Software der Sensorknoten 21

3.2.1 MuTa Betriebssystem 21

3.2.2 Vernetzung der Sensorknoten 22

3.3 Entwicklungsumgebung 24

4 Konzeption **26**

4.1 Anforderungen an den Bootloader 26

4.2 Bootloader Fähigkeiten des MSP430 27

4.3 Aufteilung der Funktionalität 27

4.4 Ablauf eines Firmware Updates 28

4.5 Rollenverteilung . . . 30
4.5.1 Funktion des Bootloaders . . . 31
4.5.2 Aufgaben der Anwendung . . . 32
4.5.3 Rolle des Host-PC . . . 32
4.6 Sicherungsmechanismen . . . 33
4.6.1 Spannungsversorgung des MSP430 . . . 33
4.6.2 Resets des MSP430 . . . 33
4.6.3 Überschreiben des Bootloaders . . . 34
4.6.4 Defekte Daten . . . 34
4.7 Interrupt Weiterleitung . . . 35
4.7.1 Feste Interrupt Vektor Tabelle im Flash-Speicher . . . 35
4.7.2 Dynamische Interrupt Vektor Tabelle im RAM . . . 37
4.8 Platzierung des Bootloaders im Speicher . . . 38
4.8.1 Bootloader am Anfang ohne Interrupt Routing . . . 38
4.8.2 Bootloader am Anfang mit Interrupt Routing . . . 39
4.8.3 Bootloader am Ende mit Interrupt Routing . . . 40
4.8.4 Bootloader am Ende ohne Interrupt Routing . . . 41
4.8.5 Fazit . . . 42
4.9 EEPROM Speicheraufteilung . . . 42
4.9.1 Firmware-Image Verwaltung . . . 43

5 Implementierung **47**
5.1 Treiber . . . 47
5.1.1 USART . . . 48
5.1.2 SPI . . . 49
5.1.3 EEPROM . . . 52
5.1.4 Flash . . . 54
5.2 Bootloader . . . 55
5.2.1 Interrupt Handling . . . 56
5.2.2 Bootloader API . . . 59
5.3 Tool-Suite . . . 61

6 Test **64**
6.1 Probleme beim Test . . . 64
6.2 Flash-Speicher Überprüfung . . . 65
6.3 Test im Sensornetz . . . 66
6.3.1 Testaufbauten . . . 68

6.4 Testergebnisse 69

7 Zusammenfassung **70**

Abbildungsverzeichnis

2.1 Sterntopologie 4
2.2 Baumtopologie 5
2.3 Vermaschtes Netz 5
2.4 Speicheraufteilung des Atmel AVR 8

3.1 Sensorknoten S1, Originalmaße: 3,6cm x 8,5cm x 0,5cm 16
3.2 Adressraum des MSP430 F1611 17
3.3 Evaluation-Board für Sensorknoten 18
3.4 Beispiel einer Vernetzung mit Slotted-MAC 23
3.5 IAR Workbench 25

4.1 Ablauf eines Firmware Updates 30
4.2 Ablauf der Interrupt Verarbeitung mit fixem Interrupt-Routing 37
4.3 Bootloader am Anfang ohne Interrupt Routing 39
4.4 Bootloader am Anfang mit Interrupt Routing 40
4.5 Bootloader am Ende mit Interrupt Routing 41
4.6 Bootloader am Ende ohne Interrupt Routing 42
4.7 Speicheraufteilung des EEPROM 43
4.8 Verweis auf ein komplettes Firmware Image 45
4.9 Verweis auf Segmente für Teilaktualisierung 46

5.1 Kontrollfluss in `spi_write` 51
5.2 Kontrollfluss beim Start des Bootloaders 56
5.3 Implementierte Speicheraufteilung des EEPROM 61
5.4 KOMBooterSuite 62

6.1 Debugger mit Flash-Speicher Ansicht 66
6.2 Testaufbauten 69

Tabellenverzeichnis

2.1 Mögliche Größen des AVR Bootloaders 9
2.2 Mögliche Platzierungen der Interrupts beim AVR 9
2.3 Intel Hex Format Zeilenaufbau 14

4.1 Einfacher Image Entry Aufbau 44
4.2 Erweiterter Image Entry Aufbau 44

6.1 Befehle der SMMART-Testanwendung 68