BNG Bootloader Documentation

Release 0

Elias Medawar, Yves Peissard, Simon Honegger, Jonathan Stoppani

20. 04. 2010

Inhaltsverzeichnis

1 Abstract **1**

2 Architektur **3**
2.1 Überblick 3
2.2 Application Driver 4
2.3 Serial Interface Driver 4
2.4 Command Parser 5

3 Spezifikation **7**
3.1 Anforderungsspezifikation 7
3.2 Befehle 8

4 Anlagen **11**
4.1 Anlage 1: Planung 11
4.2 Anlage 2: Entwicklungs-prozess und -ressourcen 11
4.3 Anlage 3: TODOs 12

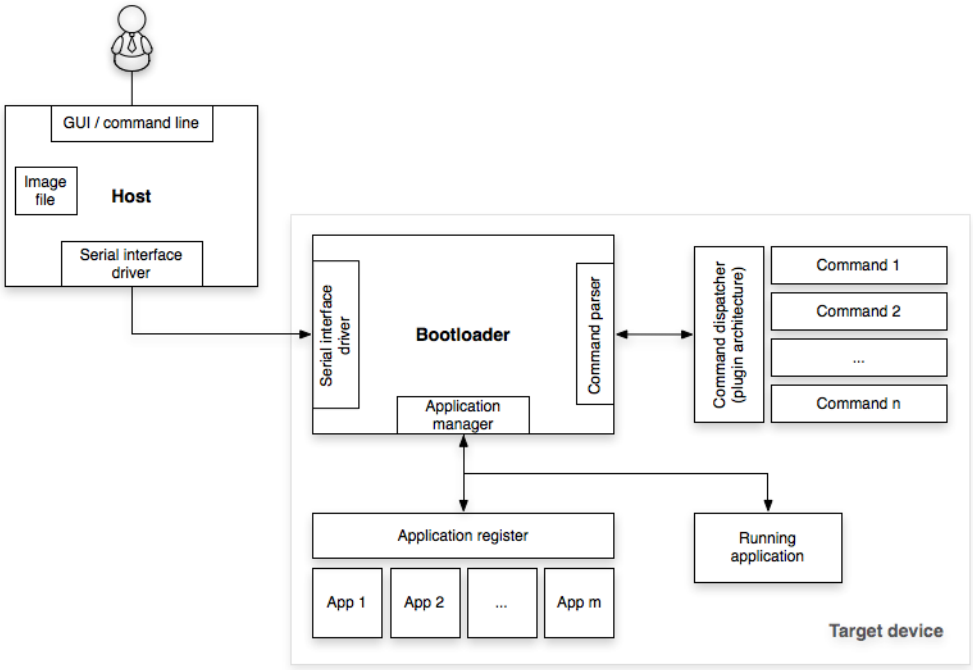
Abstract

Dieses Projekt hat zum Ziel im Ramen des Kurses “Mikroprozessoren 2” der Hochschule für Technik und Architektur einen Bootloader auf einem Mikrochip MC68332 zu implementieren. Dabei werden die behandelten Themen des Mikroprozessor Kurses *Programmieren in C* und *Grundlagen Assembler* angewendet. Nach Aufstarten des Mikrokontrollers kann eine externe Maschine (Terminal) mittels serieller Schnittstelle angeschlossen werden, mit der anschliessend Befehle eingegeben werden können um den Bootloader zu konfigurieren. Der Benutzer soll schliesslich unter zwei verschiedenen Anwendungen auswählen können, die er mit dem Bootloader starten will.

Was ist ein *Bootloader*? Definition von Wikipedia: Ein Bootloader ist eine spezielle Software, die gewöhnlich durch die Firmware (z. B. dem BIOS bei IBM-kompatiblen PCs) eines Rechners von einem bootfähigen Medium geladen und anschließend ausgeführt wird. Der Bootloader lädt dann weitere Teile des Betriebssystems, gewöhnlich einen Kernel.

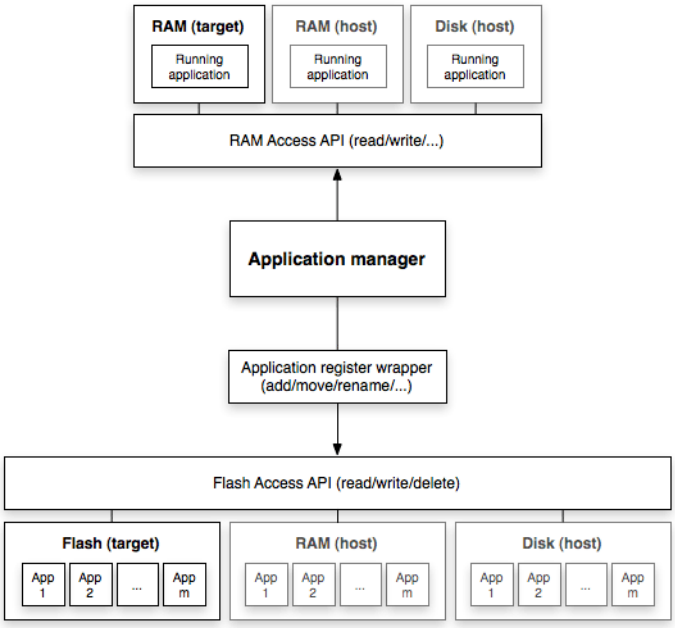
Architektur

2.1 Überblick



In der obenstehenden Gesammtübersicht sind die einzelnen Komponenten und deren Verknüpfung dargestellt. Im folgenden Kapitel wird die Architektur der jeweiligen Komponenten näher beschrieben.

2.2 Application Driver



Die Aufgabe der Application Driver ist es, das Applikationsverzeichnis zu verwalten und den aktuellen Ihnalt des Flashspeichers in den Arbeitsspeicher zu laden.

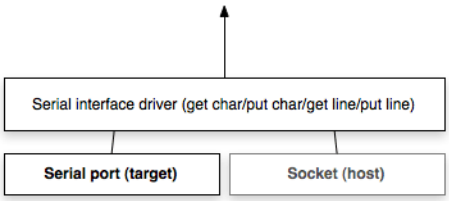
2.2.1 Flash Access API

Mit der Flash Access API können entweder Flashspeicher, Arbeitsspeicher oder Festplatte benutzt werden. Die drei Hauptfunktionen der API sind lesen, schreiben und löschen. Der Zugriff auf den Flashspeicher ist nur mittels Mikrokontroller MC68332 möglich (Zielplattform). Lesen und/oder Schreiben auf Arbeitsspeicher oder Festplatte sind für Testzwecke gedacht (Debugging).

2.2.2 Application Register Wrapper

Damit nicht direkt mit der Flash API gearbeitet werden muss, übernimmt der Application Register Wrapper die Ansteuerung der Flash Access API. Dies erleichtert das Entwickeln des Bootloaders und ermöglicht den Code des Application Driver übersichtlich zu gestalten.

2.3 Serial Interface Driver



Der Serial Interface Driver ermöglicht die Kommunikation zwischen Zielplattform (MC68332) und Benutzer des Bootloader. Mittels eines Keyboards und einem Terminal werden Zeichen an den Mikrokontroller geschickt und das Resultat vom Kontroller anschliessend wieder an den Benutzer zurück geschickt. Funktionen wie put_char oder get_char müssen hier implementiert werden. Damit wir die Serielle Schnittstelle in unserer Testumbebung emulieren können werden wir ein Socket benutzen.

2.4 Command Parser

Der Command Parser analysiert die einkommenden Befehle des Benutzerkeyboards und delegiert das Aufrufen der entsprechenden Methoden (siehe Commands).

Spezifikation

3.1 Anforderungsspezifikation

3.1.1 Funktional

Muss

Muss Ziele sind von der Projektspezifikation gegeben und müssen erfüllt werden.

1. Nach einschalten der Hardware wird der Bootloader automatisch gestartet
2. Der Bootloader muss eine Applikation aufnehmen und diese permanent speichern;
3. Der Bootloader muss eine permanente Applikation starten;
4. Der Benutzer muss eine exisitierende Applikation entfernen;
5. Binärdaten des Formats S-Record können behandelt werden;
6. Der Bootloader hat keine Auswirkung auf die funktionsweise der auszuführenden Applikation.

Kann

1. Der Bootloader zeigt eine Auswahl an zu verfügung stehenden Applikation an;
2. Der Bootloader started nach einer gewissen Zeit eine vom Benutzer als default markierte Applikation;
3. Eine Applikation kann auf das Gerät geladen werden ohne es permanent zu speichern; Dies auch wenn der Speicherplatz des permanenten Specihers belegt ist. Dieses kann ausgeführt werden; Bei Neustart des Geräts ist dieses nicht mehr vorhanden. Nach nicht permanenten laden einer Applikation; ist der vorherige Zustand nach dem neustarten wieder hergestellt;
4. Nach permanentem speichern einer Applikation und späterer Löschung ist der Zustand für den Anwender unverändert;
5. Der Speicherplatz wird insofern optimal genutzt, dass keine unnötigen, nicht verwendbaren Lücken enstehen;
6. Es kann eine default Applikation gewählt werden, welche nach einem gewissen Timeout automatisch gestartet wird;

7. Eine default Applikation wird nicht mehr als 5 mal hintereinander autmatisch gestartet, falls dessen verarbeitung fehlerhaft war;

3.1.2 Nicht-Funktional

1. Der Code sowie Kommentar sind in englisch gehalten;
2. Der Source code muss dokumentiert sein;
3. Ein Nutzer mit shell Kentnissen kommt mit der Benutzerschnittstelle zurecht. D. h., das Grundprinzip des Eingebens von verschiedenen Kommandos und deren Parameter, sowie die Benutzung des "Hilfe" Befehls sollte dem Benutzer vertraut sein.

3.2 Befehle

`help`

Zeigt eine Liste aller möglichen Befehle an.

`help command-name`

Zeigt die Hilfe für `command-name` an.

`download [-b app-name [-d]]`

Lädt die folgenden Daten in den Arbeitsspeicher als eine SRecord Datei.

-b app-id	Führt den Befehl `burn app-id` aus.
-d	Setzt die Applikation als default Startanwendung

`burn app-name [-d]`

Speichert das sich im Arbeitsspeicher befindende SRecordFile in den Flashspeicher unter dem Namen `app-name`.

-d	Setzt die zu brennende Applikation als default Startanwendung

---

`run [-r|app-id]`

Führt die default, oder die mit dem `app-id` identifizierte Applikation aus.

-r	Führt das sich im Arbeitsspeicher befindende SRecordFile aus.

`list`

Führt eine Liste der sich im Flashspeicher befindenden Applikationen auf.

`set [--default=app-id] [--timeout=seconds] [...]`

Setzt die Werte der gegebenen Konfigurationsoptionen.

-d app-id	Siehe `--default-app`
--default=app-id	Setzt die default Applikation
-t seconds	Siehe `--default-app`
--timeout=seconds	Setzt die Wartezeit nach der die default Applikation gestartet werden soll
-e num	Siehe `--default-app`
--errcount=num	Anzahl Neustarts im Falle eines fehlerhafen Starts einer Applikation. Sobald errorcount erreicht ist, wird das Timeout deaktiviert und den Bootloader wird in Terminal-Mode umschalten.

`bootconfig`

Führt die aktuelle Konfiguration (app-name, timeout, errorcount) des Bootloaders auf.

`erase app-id`

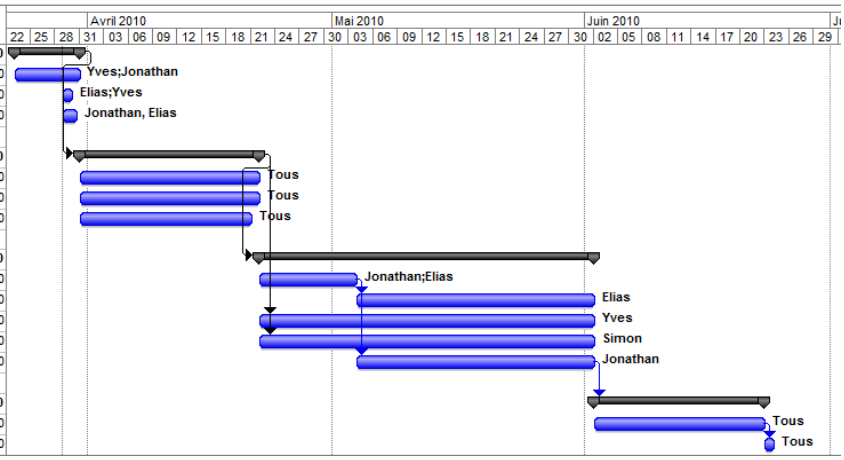
Entfernt die mit dem `app-id` identifizierte Applikation aus der Applikationsliste des Flashspeichers.

Anlagen

4.1 Anlage 1: Planung

Das folgende Bild zeigt die Großplanung für das Bootloader Projekt. Diese Planung wird während der Entwicklung kontinuierlich aktualisiert.

N°	Nom de la tâche	Début	Fin
1	**Architektur/ Spezifikation**	**Mar 23.03.10**	**Mar 30.03.10**
2	Architektur	Mar 23.03.10	Mar 30.03.10
3	Definieren von Befehlen	Lun 29.03.10	Lun 29.03.10
4	Planung	Lun 29.03.10	Mar 30.03.10
5			
6	**Design**	**Mer 31.03.10**	**Mer 21.04.10**
7	Ram access API	Mer 31.03.10	Mer 21.04.10
8	Flash access API	Mer 31.03.10	Mer 21.04.10
9	Serial Interface access API	Mer 31.03.10	Mar 20.04.10
10			
11	**Implementierung**	**Jeu 22.04.10**	**Mar 01.06.10**
12	Bootstrapper	Jeu 22.04.10	Lun 03.05.10
13	Applikation manger	Mar 04.05.10	Mar 01.06.10
14	Command Parser	Jeu 22.04.10	Mar 01.06.10
15	Flash Driver	Jeu 22.04.10	Mar 01.06.10
16	Rs32 Driver	Mar 04.05.10	Mar 01.06.10
17			
18	**Validation und Dokumentation**	**Mer 02.06.10**	**Mar 22.06.10**
19	UnitTests	Mer 02.06.10	Mar 22.06.10
20	Demonsrtation	Mer 23.06.10	Mer 23.06.10



Ein (Druck-geeignetes) PDF und das originale MS Project Dokument sind auch verfügbar:

- Planung PDF Dokument
- Planung MS Project Dokument

4.2 Anlage 2: Entwicklungs-prozess und -ressourcen

Das ganze Projekt wird auf github (http://github.com) verwaltet.

- Git Repository auf github: http://github.com/garetjax/Bootloader
- Letzer Build der Dokumentation: http://garetjax.github.com/Bootloader/

4.3 Anlage 3: TODOs

Sphinx ermöglicht das direkte Einfügen von *zu tun* Elemente in der Dokumentation. Diese Elemente können während des Builds durch eine Konfigurationsoption ausgeblendet werden.

Um ein neues *zu tun* Element hinzuzufügen, ist die folgende Syntax zu verwenden:

```
.. todo::
   A simple text describing what we have to do.
   * But it could be
   * a list too
   * if the points are many
```

Diese Elemente können durch die folgende Syntax in einer Auflistung zusammengefasst werden:

```
.. todolist::
```

Hier unten sind alle, in der Dokumentation gefundene, *zu tun* Element aufgelistet (falls sie aktiviert sind).