Flashen des Entwicklungsboards TechnoTrend MDVBDM Rev. 1.0 mittels BDI2000

Dipl.-Inf. (FH) Arkadius Nowakowski <anowa@freenet.de>

13. Juni 2005

Inhaltsverzeichnis

1 Konfiguration

Das BDI2000 benötigt unterschiedliche Informationen über den verwendeten Flash-Baustein. Diese sind in der Konfigurationsdatei im Bereich `[FLASH]` einzutragen.

Auf dem Entwicklungsboard befindet sich ein Flash-Chip vom Typ *AM29LV320DB* der Firma AMD. Genaue Informationen sind unter http://www.amd.com/us-en/FlashMemory/ProductInformation/0,,37_1447_1623_1468%5E1543,00.html zu finden und das Datasheet befindet sich unter http://www.amd.com/us-en/assets/content_type/white_papers_and_tech_docs/23579c6.pdf. Der Chip hat eine Datenkapazität von 32 Megabit bzw. 4 Megabyte, die wahlweise mit 8 Bit (8x4 Megabit) oder 16 Bit (16x2 Megabit) adressiert werden kann. Auf Seite 11 des Datasheets ist das Einstellen Adressierungsmodus beschrieben:

> Word/Byte Configuration
>
> The BYTE# pin controls whether the device data I/O pins operate in the byte or word configuration. If the BYTE# pin is set at logic 1, the device is in word configuration, DQ0-DQ15 are active and controlled by CE# and OE#.
>
> If the BYTE# pin is set at logic 0, the device is in byte configuration, and only data I/O pins DQ0-DQ7 are active and controlled by CE# and OE#. The data I/O pins DQ8 DQ14 are tri-stated, and the DQ15 pin is used as an input for the LSB (A-1) address function.

Laut Dokument `Schaltplan_MDVBDM_1.0.pdf`, Seite 13, das sich im Archiv `indmp.tgz` im Unterverzeichnis `INDIMP/BuildEnv` befindet, ist der Pin 53 (`BYTE#`) des Flash-Chips an die 3V3 Leitung angeschlossen. Somit entspricht dies einer *logic 1* und der Baustein befindet sich in einer *word configuration*. Ferner stellt der BDI2000-Hersteller Abatron eine Liste mit den unterstützten Flash-Chips und den dazugehörigen Werten für die Konfiguration des BDI2000 zur Verfügung, die von http://www.abatron.ch/Files/Flashsupp5.pdf bezogen werden kann. Anhand den dort vorhandenen Informationen und der Tatsache, dass der Chip mit 16 Bit adressiert wird, ergibt sich für die Konfigurationsvariable `CHIPTYPE`, die den Flash-Algorithmus beschreibt, der Wert

Byte 7 == 0xff	Slot 0	Slot 1
ja	System	Daten
nein	Daten	System

Tabelle 1: Belegung der Slots auf dem Flash-Baustein abhängig von Byte 7 des EEPROMs

`AM29BX16`. Des Weiteren benötigt man die Option `BUSWIDTH`, die zur Spezifizierung der Datenbusbreite aller Flash-Chips dient. Da auf dem Entwicklungsboard nur ein Chip mit 16 Bit verwendet wird, ist hier der Wert `16` einzustellen.

2 Flash-Aufteilung

Für den laufenden Betrieb wurde ein Mechanismus entwickelt, der eine sichere Aktualisierung der System- und Anwendungsdaten auf der Set-Top-Box ermöglicht. Der Flash-Adressraum beträgt, wie oben beschrieben, 4 Megabyte, die für den Mechanismus in vier logische Bereiche aufgeteilt wurden - einen Bootloader-, einen System- und zwei Datenbereiche. Der Systembereich und ein Datenbereich, auch Slot 0 bzw. Slot 1 genannt, sind gleich groß und nehmen den Großteil der Datenkapazität ein. Welcher Bereich sich in welchem Slot befindet, hängt von Byte 7 des EEPROMs ab. Ist dessen Wert 255, so befindet sich das System innerhalb des Slot 0 und die Daten innerhalb von Slot 1. Weicht dieser Wert von 255 ab, ist die Slot-Belegung vertauscht. Tabelle 1 veranschaulicht diesen Zusammenhang.

Der Bootloader befindet sich am Anfang des Flash-Speicherbereichs und entscheidet beim Systemstart, abhängig vom Byte 7 des EEPROMs, ob das System von Slot 0 oder Slot 1 geladen werden soll. Direkt hinter dem Bootloader liegen Slot 0 und Slot 1, gefolgt von dem wesentlich kleineren zweiten Datenbereich.

Bei einem System-Update im laufenden Betrieb, erfolgt ein Tausch der Slot-Belegung und eine dementsprechende Änderung des Byte 7. Befindet sich das System vor Beginn des Update-Vorgangs innerhalb von Slot 0, so wird das aktuelle System in den Slot 1 geschrieben. Anschließend erfolgt die Aktualisierung des Datenbereichs in Slot 0 und Byte 7 erhält einen anderen Wert als 255. Somit wird gewährleistet, dass bei einer Unterbrechung des Update-Vorgangs das alte, aber funktionsfähige, System immer noch in Slot 0 vorhanden ist.

Die Slot-Größen können über die Kernelkonfiguration eingestellt werden. Hierzu erfolgt der Aufruf `make menuconfig` im Verzeichnis `/av72demo/linux` innerhalb der Sandbox. In erster Linie ist die Option `Support for slot mechanism` notwendig. Die dazugehörende Kerneloption befindet sich unter:

```
System Type -> AV7200 Options -> Support for slot mechanism
```

Ist diese aktiviert, ist dort die Größe des Bootloaderbereichs (`Size of Bootloader`) und die Größe für Slot 0 und Slot 1 (`Slotsize`) einzustellen. Wie oben bereits erwähnt, sind Slot 0 und Slot 1 von ihrer Dimensionierung her identisch. Aus diesem Grund ist auch nur eine Option für die Slot-Größe vorhanden. Beide Werte sind im Hex-Zahlenformat einzutragen. Derzeit ist der Bootloader auf eine Größe von `0x20000` und die Slots auf eine Größe von `0x200000` Byte eingestellt, was 128 kByte bzw. 2048 kByte entspricht. Diese Konfiguration stimmt nicht mit der Datenkapazität des vorhandenen Flash-Bausteins überein, da die eingestellten Werte einen verfügbaren Datenraum von 4224 kByte vorraussetzen, auf dem Baustein jedoch nur 4096 kByte vorhanden sind. Tabelle 2 veranschaulicht die Positionen der unterschiedlichen Bereiche auf dem Flash-Baustein.

3 Flash-Inhalt

Sämtliche Daten, die auf das Flash kommen sollen, befinden sich innerhlab der Sandbox in sog. CRAMFS-Images. CRAMFS ist die Abkürzung für *Compressed ROM Filesystem*, ein Dateisystem für ausschließlich lesenden Zugriff. Tabelle 3 zeigt für welchen Flash-Bereich, welche Dateien

Adresse	Bereich
0x000000	Bootloader
0x020000	Slot 0
0x220000	Slot 1
0x420000	Daten

Tabelle 2: Adressen der Bereiche auf dem Flash-Baustein

Bereich	Datei
Bootloader	`linux/arch/arm/boot/zImage.flash`
Slot 0	`slot.av72demo.v1.cram`
Slot 1	`fs/usr2.av72demo.v1.cram`
Daten	`fs/usr3.av72demo.cram`

Tabelle 3: Flash-Bereiche und die dazugehörenden Dateien

vorgesehen sind, für den Fall, dass Byte 7 im EEPROM den Wert 255 enthält.. Die Dateipfade sind relativ zum Verzeichnis `/av72demo`.

Das Erstellen der Dateien kann auf mehreren Wegen erfolgen. Die Vorraussetzung für ein lauffähiges System ist der Kernel, welcher im Sandbox-Verzeichnis `/av72demo/linux` per `make zImage` erstellt werden muss. Dieser Aufruf kompiliert u.A. den Bootloader und legt diesen in der Datei `zImage.flash` ab (siehe Tabelle 3). Diese Datei besteht im wesentlichen aus einem Header, der von der größe des SDRAM abhängt, und aus dem Bootloader. Die Header liegen in binärer Form im Verzeichnis `linux/arch/arm/boot` und sind an ihren Dateinamen `kern.head.size0x...` zu erkennen.

Für das Erstellen des System-Images `slot.av72demo.v1.cram` ist das Skript `/av72demo/linux/slotmachine` zuständig. Es baut diese Datei aus vier Teilen zusammen. Der erste Teil ist nur 8 Byte lang und enthält zwei Hexwerte - die Adresse des Kernels im gemappten Adressraum (`0x10268000`) und die Größe des komprimierten Kernels in der Datei `arch/arm/boot/compressed/kern.gz`, welcher sich gleich hinter den beiden Werten im Image befindet. An dritter Stelle stehen, abhängig von der Kernelgröße, bis zu drei Null-Bytes. Diese werden eingefügt, da das, an vierter Stelle stehende, root-Filesystem ausschließlich an 4-Byte-Grenzen liegen darf. Das Image für das Dateisystem ist zuvor vom Skript `/a72demo/build_rootfs` zu erstellen. Dabei wird aus dem Verzeichnis `/av72demo/rootfs_files` ein CRAMFS-Image generiert. Alternativ kann auch per Skript `/av72demo/make_loader_n_slot0` der Bootloader zusammen mit dem System in ein gemeinsames Image geschrieben werden, welches an den Anfang des Flash-Speichers zu schreiben ist. Die erstellte Image-Datei hat dabei ebenfalls den Namen `slot.av72demo.v1.cram`.

Die Images `fs/usr2.av72demo.v1.cram` und `fs/usr3.av72demo.cram` können mit dem Skript `fsos <fsname> <version>` im Verzeichnis `/av72demo/fs` erstellt werden. Der erste Parameter gibt die Bezeichnung des zu erstellenden Dateisystems an, in dem Fall `usr2` oder `usr3`. Der zweite Parameter steht für die Versionsnummer, die nach dem Aufruf im Dateinamen hinter dem kleinen `v` zu finden ist. Für die erste Datei lautet der Aufruf daher `./fsos usr2 1` für die Zweite `./fsos usr3 1`.

Alle drei CRAMFS-Images können auch mit dem Skript `projector` auf ein mal erstellt werden. Hierbei enthält die Datei `slot.av72demo.v1.cram` bereits den Bootloader und ist somit an den Flashanfang zu schreiben.

4 Flashen von Daten

Bevor ein ein Flash-Chip beschrieben werden kann, ist vor dem Schreibvorgang ein Löschen der zu beschreibenden Sektoren zwingend notwendig. Das BDI2000 stellt hierfür den Befehl `ERASE <address> [<mode>]` zur Verfügung. Das erste Argument gibt die Adresse an, ab der die Daten gelöscht werden sollen. Das zweite Argument ist optional und gibt an, ob der gesamte Chip, ein Block oder ein Sektor an der zuvor angegebenen Adresse gelöscht werden soll. Die möglichen Werte für `<mode>` lauten daher dementsprechend `CHIP`, `BLOCK` und `SEKTOR`, wobei der letzte standardmäßig beim Auslassen des Arguments gesetzt wird. Für das Löschen des gesamten Flash-Inhalts auf dem Board ist somit der Aufruf `ERASE 0x20000000 CHIP` erforderlich. Alternativ dient der Befehl `ERASE <address> <size> <count>` zum Löschen mehrerer Sektoren gleichzeitig. Der erste Parameter hat dabei die gleiche Bedeutung wie vorher. Der Parameter `<size>` gibt die Sektorgröße an und mit `<count>` wird die Anzahl der zu löschenden Sektoren festgelegt. Möchte man nur den Bootsektor löschen, genügt es den Befehl `ERASE 0x20000000 0x20000 1` abzusetzen.

Das Bespielen des Flash-Speichers erfolgt mit dem BDI2000-Befehl `PROG [<offset>] [<file> [<format>]]`. Der Parameter `<offset>` steht wie gewohnt für die Adresse, ab der der Schreibvorgang durchzuführen ist. Der zweite Parameter `<file>` steht für den Pfad zur schreibenden Datei, die sich auf dem TFTP-Server befindet, d.h. es ist vor dem Aufruf des Befehls ein Kopieren der gewünschten Datei in das TFTP-Verzeichnis erforderlich. Das Format der Datei kann im dritten Parameter `<format>` angegeben werden. Hier besteht die Auswahl zwischen `AOUT`, `BIN`, `COFF`, `ELF` und `SREC`. Befindet sich das System-Image mit Bootloader innerhalb des TFTP-Verzeichnisses in der Datei `system`, ist der Befehl `PROG 0x20000000 system BIN` aufzurufen. Die Adresse ergibt sich aus dem Mapping des Flash-Speichers an die Adresse `0x20000000`. Soll das System-Image ohne Bootloader geschrieben werden, so ist die Adresse mit `0x20200000` zu ersetzen. Der Bereich `[FLASH]`, in der Konfigurationsdatei für das BDI2000, kann die Parameter für den Befehl `PROG` ebenfalls beinhalten. Hier gibt die Variable `FILE <filename>` den Dateinamen und die Variable `FORMAT <format> [<offset>]` das Format und die Startadresse für den Flash-Vorgang an. Eine, zu dem soeben genantnen Beispiel, äquivalente Konfiguration sieht wie folgt aus:

```
[FLASH]
...
FILE bdi2000/system
FORMAT BIN 0x20000000
...
```

Mit Hilfe dieser Einstellungen kann der Flash-Vorgang mit Absetzen des BDI2000-Befehls `PROG` gestartet werden.