Arm&Eva Antworten

Zusammenfassung

Dieses Dokument soll einen schnellen Einstieg in die Entwicklung mit Arm&Eva ermöglichen. Es beantwortet häufig gestellte Fragen und gibt Hinweise zu weiterführenden Dokumenten.

Inhaltsverzeichnis

1 Konfiguration

1.1 Allgemeines

Das Arm-Modul besitzt einen Flash, welcher die Konfiguration des Systems enthält. Die Konfiguration gliedert sich in:

- Bootrelevante Daten (Bootloader, Linux-Loader), **Linux-Kernel**
- Low-Level Konfiguration
  - MAC-Addresse
  - Seriennummer
  - Netzwerk Konfiguration (IP-Addressen, ...)
  - **Anwenderkonfiguration**
- Firmware (**cramfs** oder **squashfs**)

Alle Daten liegen in einem rudimentären Dateisystem (TinyFs), welches sowohl vom BootLoader, als auch vom Linux System aus erreicht werden kann. Um Dateien in diesem TinyFs zu lesen oder zu schreiben wird das Programm “dfConfig” verwendet, welches auf dem Zielsystem installiert ist.

Auf das TinyFs Dateisystem kann eingeschränkt schreibend zugegriffen werden. Die Einschränkung betrifft die Dateigrösse einer zu schreibenden Datei.

Generell kann eine Datei problemlos überschrieben werden, solange die Datei nacher kleiner oder genausogroß ist wie die Datei vorher. Das vergrößern einer Datei ist nur in gewissem Rahmen möglich. (Siehe ebFlashSetup Projektdatei; Abschnitt **AllocateFlashSize**.)

1.2 Wie starte ich mein eigenes Programm / Skript beim Systemstart ?

1.2.1 Möglichkeit 1

Erstellen Sie eine Dos-Partition auf einer SD-Karte. (Die meisten SD-Karten sind so vorformatiert.)
Erstellen Sie auf dieser SD-Karte ein Verzeichnis mit dem Namen “autostart” und kopieren Sie dort Ihr Skript hinein.
Wenn die SD-Karte beim Systemstart im Slot liegt, werden alle Skripte / Programme in diesem Verzeichnis ausgeführt. (Siehe etc/init.d/rcInit.sdBoot)

Hinweis

Dieses Merkmal kann auch für Firmware-Updates verwendet werden.
In diesem Fall muss das Skript in diesem Verzeichnis den Befehl zum Auswechseln des Flashinhaltes beinhalten.

1.2.2 Möglichkeit 2

Ändern Sie das Skript “**app/init.d/rcNormal**” oder auch “**app/init.d/rcBackground**” in der Firmware und flashen Sie das System neu. (Siehe auch 2.1 und 2.2.)

1.3 Wie ändere ich die Ethernet / IP-Addresse ?

Die IP-Addresse wird beim Systemstart konfiguriert. (**/etc/init.d/rcInit.net**)
Dabei wird die IP-Addresse verwendet, die in der TinyFS Datei “eth0-address” steht. Sie können diese Datei leicht auslesen:

- dfConfig -rs eth0-address

Analog können Sie den Inhalt der Datei verändern:

- echo -n “192.168.1.99” > /tmp/n
- dfConfig -w eth0-address /tmp/h

Nach einem Neustart wird die neue IP-Addresse (192.168.1.99) verwendet.

Hinweis

Wenn Sie anstelle einer IP-Addresse das Wort "dhcp" hinschreiben, konfiguriert sich das System per dhcp.

1.4 Wie ändere ich die Webseiten ?

Das Arm&Eva verwendet per Default einen kleinen Web-Server (thttpd). Dessen Daten liegen unter “**/home/httpd/data**” im Dateisystem.

2 Firmware

Im Dateisystem befinden sich alle benötigten Programme, Bibliotheken und Konfigurationsdateien. Beispiele sind:

- web-server (thttpd)
- web-seiten
- ftp-server (bftpd)
- telnet-server (busybox)
- ...

Dieses Dateisystem wird auch als Firmware bezeichnet.

Auf dem Zielsystem liegt das Dateisystem als read-only Flashdateisystem vor, das beim Systemstart vom Kernel gemounted wird.
Ausgeliefert wird das Arm&Eva mit einem **SquashFS** 4.0-Rootdateisystem. Es kann je nach Kernel-Konfiguration auch **cramfs** verwendet werden. (squashfs komprimiert etwas besser.)

2.1 Erstellung der Firmware

Zur Erstellung (mastern) des Dateisystems werden die squashfs-tools benötigt. (Wenn cramfs verwendet werden soll, entsprechend die cramfs-tools.) Der Quellcode der squashfs-tools liegt unter "**sw/linux/tools**" im Arm&Eva Dateibaum.

2.1.1 Beispiel

Unter "**sw/linux/prepare.cfs**" finden Sie die Skripte "master" und "original", sowie zwei weitere Verzeichnisse:

- **resource.common**
  Hier liegt die Basisdistribution.
- **specialized.original**
  Hier liegen spezielle (anwendungsabhängige) Dateien und Skripte.

Für eine neues Projekt mit dem Arm&Eva sollten Sie zunächst specialized.original zu einem neuen Verzeichnis specialized.myProject kopieren. (Das Prefix "specialized" ist zwingend.)
Sie können nun Änderungen nach Ihren Wünschen in "specialized.myProject" durchführen.

Nachdem Sie die gewünschten Änderungen vorgenommen haben, müssen Sie ein neues Dateisystem erstellen:

- sudo ./master myProject

Das Skript "**master**" hilft Ihnen dabei alle Dateien an ihren Platz zu kopieren, bevor dann das Dateisystem gemastert wird.

Als Resultat folgt eine Datei mit der Endung .sqfs. Diese Datei kann jetzt auf das Zielsystem übertragen und in den Flash programmiert werden. (Siehe 3.2.)

Der Dateiname wird aus Produktname+Version gebildet. Siehe 2.2.

2.1.2 Hinweise

Hier nochmals der Hinweis, dass das squashfs (genauso wir cramfs) ein Read-Only Dateisystem ist. Wenn Sie auf dem Zielsystem Dateien erzeugen oder verändern wollen, machen Sie das in /tmp.

Es wird ein Verzeichnis mit dem Namen **Target** erstellt, welches ein exaktes Abbild des Dateisystems ist.

Hinweis

Achten Sie bei der Erstellung von Verzeichnissen und Dateien auch auf die Dateirechte (Zugriffsrechte). Sie werden unmittelbar in das SquashFs-übernommen.

Hinweis

2.2 Welche Änderungen kann ich durchführen ?

Es folgt eine Zusammenstellung interessanter Dateien, die Sie kennen sollten[1]:

- **/etc/cfg/product**
  Eine Textdatei, welche den Namen des Projektes / Produktes enthält.
- **/etc/cfg/version**
  Eine Textdatei, welche die Version ihrer Firmware enthält.
- **/app/init.d/rcBackground**
  Ein Skript, dass sehr früh (vor der Netzwerk Konfiguration), bei der Systeminitialisierung aufgerufen wird. (Siehe auch: **etc/init.d/rcInit**)
- **/app/init.d/rcNormal**
  Dieses Skript wird aufgerufen, wenn das System fehlerfrei und vollständig initialisiert wurde. Hier können Sie Ihre eigenen Programme starten.

Ihre Programme und Bibliotheken können Sie zum Beispiel unter **app/sw/bin** bzw. **app/sw/lib** kopieren. Unter **app/drv** liegen Gerätetreiber, die bei der Systemkonfiguration geladen werden.

Sie haben bei Änderungen in den genannten und anderen Verzeichnissen natürlich absolut freie Hand. Diese Struktur ist nur als Vorschlag zu verstehen, um ein System zu bekommen, welches möglichst einfach wartbar ist.

2.3 Linux-Kernel Update

Im Verzeichnis sw/linux-kernel befindet sich eine Datei mit dem Namen “ReadMe”. Hier ist beschrieben, wie Sie ein Kernelupdate durchführen.

[1] Absolute Pfade beziehen sich auf das Zielsystem. Auf Ihrem Entwicklungsrechner liegen die Daten unter “sw/linux/prepare.cfs/specialized.yourProject”.

3 Update des Systems

3.1 Wie verschaffe ich mir Überblick über freien Flash-Speicher ?

Bitte geben Sie auf dem Zielsystem folgendes ein:

- dfConfig -l

Zum Beispiel:

cramfs | 3252224 bytes | 54% growable (3695200 bytes left)
bedeutet, dass der Eintrag "**cramfs**" noch um 54% wachsen kann. Wenn Sie cramfs aktualisieren, kann die Datei also bis zu **3252224 + 3695200 ca. 6.6MByte** groß sein**.**

3.2 Wir aktualisiere ich die Firmware ?

Zur Erstellung der Firmware lesen Sie bitte das Kapitel 2.

3.2.1 Voraussetzungen

Voraussetzung ist, dass das neue Dateisystem nicht grösser ist als der Platz, den Sie ihm während des Setup-Vorgangs gegeben haben. (Siehe Datei linux/setup/recovery.prj Sektion "cramfs" / AllocateFlashSize = 0x....)
Per Default (Auslieferungszustand) darf das (komprimierte) Dateisystem ca. 6.6MByte groß sein.

3.2.2 Durchführung

Kopieren Sie die Firmware per ftp auf das Zielsystem oder machen Sie es per NFS-Share verfügbar.
Geben Sie folgendes ein:

- dfConfig -w cramfs /path/to/newfs.sqfs

Bei einem Neustart wird die neue Firmware aktiv.

Hinweis

Das programmieren des Flashs kann von einigen Sekunden bis zu mehreren Minuten dauern.

3.3 Wie aktualisiere ich den Kernel aus dem laufenden System ?

3.3.1 Voraussetzungen

Voraussetzung ist, dass der neue Kernel nicht grösser ist, als der Platz dem Sie dem Kernel im Setup-Vorgang gegeben haben. (Siehe Datei linux/setup/recovery.prj Sektion "Linux-Kernel" / AllocateFlashSize = 0x....)
Per Default (Auslieferungszustand) darf der komprimierte Kernel nicht größer als 1.4MByte sein.

3.3.2 Durchführung

Kopieren Sie den Kernel per ftp auf das Zielsystem oder machen Sie ihn per NFS-Share verfügbar.
Geben Sie folgendes ein:

- dfConfig -w TheOS /path/to/zImage

Der Befehl kehrt nach einigen Sekunden zurück. Bei einem Neustart wird der neue Kernel aktiv.

3.4 Wie entferne ich den Bootloader ?

3.4.1 Möglichkeit 1

- Trennen Sie alle Verbindungen vom Board (USB, RS232, Netzteil, usw.)
- Verbinden Sie das Eva-Board mit Hilfe der RS232 und starten Sie ein Terminalprogramm auf dem Entwicklungs-PC. (115200/8/N/1).
- Verbinden Sie das Netzteil mit der DC-Buchse
- Sobald Sie Meldungen im Terminal sehen, drücken Sie “r” und bestätigen Sie mit “y”

3.4.2 Möglichkeit 2

Diese Möglichkeit besteht, wenn Linux bereits läuft.

- dd if=/dev/zero of=/tmp/x bs=4000 count=1
- dfConfig -w LdDataFlash /tmp/x
- reboot

Dies überschreibt den Bootloader mit Nullen.

Hinweis

Nach einem Neustart können Sie mit Hilfe der ebTools eine Neuprogrammierung vornehmen.

Hinweis

Der Austausch des Bootloaders (LdDataFlash) ist derzeit nur mit Hilfe der ebTools möglich.

3.5 Das Übertragen der Firmware mit den ebTools

Frage: Das Übertragen der Firmware funktioniert nicht. Nach Eingabe von make setup bleibt die Übertragung stehen. Wo liegt das Problem ?

Antwort: Bei einigen Betriebssystemdistributionen wird das USB-Dateisystem **/proc/bus/usb** nicht automatisch gemounted. Fügen Sie folgende Zeile in die Datei /etc/fstab hinzu:

```
usb                 /proc/bus/usb    usbfs    defaults          0          0
```

Starten Sie Ihren Rechner neu oder geben Sie ein:
    sudo mount /proc/bus/usb.

Danach sollte das Setup funktionieren.

4 Hardware

Aufgrund der Tatsache, dass es sich bei Linux um ein Multitasking / Multiuser Bentriebssystem handelt, sind Zugriffe auf die Hardware nur aus dem Kernel-Kontext heraus gestattet. Dies führt dazu, dass für die Unterstützung von Peripherie Kerneltreiber notwendig sind. Für das LC-Display, sowie die LEDs sind bereits Kerneltreiber entwickelt worden.

4.1 Wie greife ich auf die SD-Karte zu ?

Die SD-Karte muss in das Dateisystem eingehangen (gemounted) werden.

- mount -t msdos /dev/mmcA1 /mnt/sd

Dies mounted die SD-Karte und unterstellt dabei ein FAT-Dateisystem.

Hinweis

Wenn Sie Unterstützung für lange Dateinamen wünschen müssen Sie stattdessen das Dateisystem vfat verwenden. Beachten Sie bitte, dass hierzu der Kernel entsprechend konfiguriert sein muss.

4.2 Das Display

4.2.1 Kontrast und Helligkeit

Der Display Gerätetreiber stellt zwei sysfs Nodes zur Verfügung:

- /sys/bus/platform/drivers/lcd-uc1611/backlight
  Für die Helligkeit (0..100%)
- /sys/bus/platform/drivers/lcd-uc1611/contrast
  Für den Kontrast (0..100%)

Beispiel: Folgendes Beispiel stellt die Helligkeit auf maximum und den Kontrast auf 40% ein.

- echo 40 > /sys/bus/platform/drivers/lcd-uc1611/contrast
- echo 100 > /sys/bus/platform/drivers/lcd-uc1611/backlight

4.2.2 Wie kann ich etwas auf dem Display darstellen ?

Möglichkeit 1 (Text)
Per Default stellt das Display eine normale Linux-Konsole dar. Sie können Sie über **/dev/tty1** auf dem Zielsystem erreichen:

- clear > /dev/tty1
- echo “Hello World” > /dev/tty1

Führt zur Darstellung von Text auf dem Display.

Möglichkeit 2 (Grafik)
Der Displaytreiber stellt zusätzlich zur Konsole-Funktionalität noch einen Device-Knoten (device node) zur Verfügung. Es handelt sich um "/dev/lcd". Hier können Sie direkt Bilder im PNM Format hinein laden. Sie werden unmittelbar auf dem Display dargestellt.

Um Beispielsweise innerhalb einer Anwendung Grafik auszugeben erstellen Sie einfach PNM-Grafiken und geben diese direkt an das Device aus.

4.3 LEDs

Die vier LEDs können durch sysfs Nodes gesteuert werden. Ein Beispiel:

- echo 1 > **/sys/class/leds/led10/brightness**
  Zum einschalten.
- echo 0 > **/sys/class/leds/led10/brightness**
  Zum ausschalten.

Der Treiber, welcher die LEDs steuert, implementiert die LED-Class Treiberschicht des Kernels.

4.4 Serielle Schnittstellen

Das MCU-Modul verfügt über bis zu fünf serielle Schnittstellen. (Das Evalboard "Eva" führt nur die Debug-Schnittstelle direkt heraus.)

Um die anderen Schnittstellen verfügbar zu machen, müssen Sie den Kernel anpassen und neu kompilieren. Detailierte Anweisungen dazu finden Sie im Dokument **"doc/de/serial.pdf"**.

4.5 USB-Device

Um den USB-Device Port zu verwenden, werden sog. USB-Gadget Treiber benötigt. Mit Hilfe dieser Gadget-Treiber lässt sich das MCU-Modul über USB als Netzwerkgerät oder als "Virtual COM" Gerät ansprechen. Entsprechende USB-Treiber für die Host-PC Seite (USB-Netzwerk oder USB-Comm) Treiber sind bereits vorhanden. Für nähere Informationen lesen Sie bitte "**doc/en/gadget/gadget.pdf**".

4.6 Andere Hardware

Das MCU-Modul bietet zahlreiche I/O-Ports, SPI Schnittstellen und andere Peripherie. Wenn Sie davon Gebrauch machen wollen benötigen Sie einen Kerneltreiber. Für einige Schnittstellen (z.B. SPI, UART oder I²C) existieren bereits fertige Implementierungen im Linux-Kernel. Hier sei auf die Kernelquellen verwiesen.

Hinweis

Gerne entwickeln wir auch für Ihre Peripherie entsprechende Gerätetreiber und Bibliotheken.

5 Software Entwicklung

5.1 Wie kompiliere ich Software für das MCU-Modul ?

Sie verwenden den Cross-Compiler, der unter "/usr/local/carmeva/bin/compiler/gcc-3.4.1-glibc-2.3.3" installiert ist.[2]

Sie können die Ihnen bekannten build-Werkzeuge (make, kdevelop, eclipse o.ä. verwenden.)
Alternativ können Sie das Buildsystem b++ verwenden, welches auch zum kompilieren der Bootloader verwendet wird. Ein Beispiel befindet sich unter "**sw/linux/examples/leds**". Dieses Beispiel lässt die LEDs blinken. Zum kompilieren geben Sie einfach b++ ein. (Das in diesem Verzeichnis vorhandene Makefile führt bei der Eingabe von "make" ebenfalls dazu, das b++ aufgerufen wird.)

5.2 Debugging

5.2.1 JTAG Debugging

Das MCU-Modul verfügt über eine JTAG Schnittstelle. Mit Hilfe dieser Schnittstelle können Sie das Betriebssystem oder auch den Bootloader (sowie die Betriebssystem unabhängigen Demos) debuggen. Voraussetzung dafür ist ein JTAG-Debugger (z.B. BDI2000 oder auch den Galep5 mit Hilfe der g5ocd Suite.) Lesen Sie bitte "**doc/en/debugging/debugging.pdf**".

Hinweis

Bitte beachten Sie, dass die JTAG Schnittstelle zum debuggen von Linux-ARM Programmen, die unter Linux laufen, nur bedingt geeignet ist. Hier eignet sich gdb-server besser.

5.2.2 gdb-server

Es ist möglich gdb auf dem MCU-Modul laufen zu lassen, um Anwendungen zu debuggen. Lesen Sie hierzu: "**toolchain/doc/crossdebug.pdf"**.

Weiterführend beachten Sie bitte die gdb-Dokumentation: z.B. "**http://www.gnu.org/software/gdb/documentation/**".

[2] Alternativ können Sie sich einen anderen Cross-Compiler z.B. mit Hilfe von "**crosstool**" oder "**crosstool-ng**" erzeugen.