MXE11: Unix auf dem Mikrocontroller

V1.71 (c) 2017-2018 Jörg Wolfram

1 Start und Stop

Insgesamt enthält die Imagedatei 3 Diskimages. Alle Images passen in 8 Megabytes und damit auf den AT45DB642 Dataflash. Das erste Disk-Image (/dev/rk0) wird immer nach / gemounted, von den anderen 2 Images kann immer nur eines gemounted werden, vorzugsweise nach /mnt. Das „mount" Binary befindet sich im /etc Ordner, folglich muss der Aufruf dann heißen:

```
/etc/mount /dev/rk1 /mnt
```

und zum unmounten:

```
/etc/umount /dev/rk1
```

Auf Controller-Ebene gibt es einen Cache, der insbesondere häufige Schreibaktionen verhindern soll. Der Write-Cache ist teilweise relativ klein und puffert bei 64K Controller-RAM gerade einmal 4 Sektoren. Das reicht aber aus, um die periodischen Schreibzugriffe im Ruhezustand vom Speichermedium fernzuhalten.

Nach dem Start erscheint kurz eine Meldung, während der man ndurch 2-maliges Drücken der Leertaste in das Setup-Menü gelangt. Momentan lässt sich hier nur die (evtl. vorhandene) RTC stellen. Später können hier noch andere Dinge wie z.B. Image-Transfer über die serielle Schnittstelle dzukommen. Danach wird die aktuelle Version angezeigt und darunter ein Prompt (@). Hier muss jetzt der zu startende Kernel angegeben werden. Beim verwendeten Mini-Unix kann zwischen 2 Kerneln gewählt werden. „rkmx" startet den unveränderten ursprünglichen Kernel, „mx" hingegen einen leicht angepassten. Genaugenommen ändert sich nur die Art, wie Datum und Uhrzeit beim Start bestimmt werden. Während „mx" die aktuelle Zeit der RTC nimmt, liest der „rkmx" Kernel die letzte Änderungszeit des Superblocks auf Disk1. Außerdem kann der modifizierte Kernel mit mehreren Terminals umgehen.

Während des Starts leuchtet die Drive-LED ständig, um danach auszugehen. Falls das Medium nicht gefunden wurde, beginnt die Drive-LED zu blinken und im Terminal wird **MXE11: NO DRIVE** angezeigt. In diesem Fall muss kontrolliert werden, ob das Flashmedium richtig angeschlossen bzw. die SD-Karte kontaktiert ist. Dabei wird nur die physikalische Anwesenheit des Mediums überprüft, nicht der Inhalt.
Danach wird die Sys-LED aktiviert und versucht, vom Medium zu booten. dabei wird der erste Sektor in das RAM eingelesen und an den Anfang des gelesenen Speicherbereiche gesprungen. Ist kein gültiger Bootsektor vorhanden, hält in den meisten Fällen der Emulator mit einem **User Halt** und die Status-LED beginnt zu blinken.

Nach erfolgreichem Start folgt der Login-Prompt. Einloggen kann man sich als **root** oder als User **mxe**, in beiden Fällen ist kein Passwort gesetzt.

Zum Beenden muss die PDP11 heruntergefahren werden. Zu Herunterfahren dient das Kommando bzw. gleichnamige Programm **halt**, der zuerst ein sync() ausführt und danach den Write-Cache im Controller auf das Speichermedium zurückschreibt. Abschließend verlischt die Status-LED und die Stromversorgung kann abgeschaltet werden.

Achtung: Entfernen Sie **NIE** die Stromquelle oder resetten den Controller, solange das System nicht heruntergefahren ist. Geschieht dies trotzdem, ist mit Sicherheit mit Datenverlusten oder inkonsistenten Dateisystemen zu rechnen.

```
#halt
Cleanup cache ......
MXE11: SYSTEM HALTED
```

2 Geänderte und Zusätzliche Programme

2.1 Die Shell (sh)

Die Shell (**sh**) enthält momentan 2 Besonderheiten:

1. Ausser dem unter Unix V6 üblichen **chdir** lässt sich auch **cd** verwenden
2. Mit dem Kommando **prompt** lässt sich (experimentell) der aktuelle Pfad im Prompt einblenden

2.2 System anhalten (halt)

Dieses Programm stammt von mir und läuft nur auf dem MXE11. Es dient zum Herunterfahren des Systems. Zuerst wird ein sync() ausgeführt, dann der Magic-Wert 1234d in die Speicherzelle 0xFABC geschrieben. Dies veranlasst den Emulator, den Cache zu leeren und das System anzuhalten. Nach erfolgtem Halt verlöscht die Status-LED.

2.3 MIPS Benchmark (mips)

Dieser Benchmark führt sehr oft hintereinander den Befehl **INC R4** aus. Als Parameter wird angegeben, wieviele Millionen mal der Befehl ausgeführt werden soll. Gibt man hier z.B. 100 an, lässt sich leicht aus der verstrichenen Zeit der MIPS-Wert bestimmen. Für genauere Resultate sollte der Befehl mittels **time** gestartet werden.

2.4 Türme von Hanoi Benchmark (hanoi)

Dieser Benchmark sortiert einen 10-er Stapel Scheiben um. Der Parameter gibt die Anzahl der Sortierungen an, ein guter wert ist 1000. Für genaue Resultate sollte der Befehl mittels **time** gestartet werden, die Anzahl der Loops pro Sekunde ist dann
1000 / gemessene Zeit.