Skript Abarbeitung des Monitors

Der Monitor gestattet die automatisierte Abarbeitung von Befehlsfolgen. So müssen komplexe Befehlskombinationen nicht mühsam jedes Mal per Hand wiederholt werden. Die Befehlsausführung erfolgt über den Befehl

```
=> run Var
```

wobei die Variable selbst wieder mehrere Befehle enthalten darf. Exemplarisch sollen die Befehlsfolgen zum Test einer SD-Card in ein ausführbares Skript gebracht werden. Um eine SD-Card zu testen und sich den Inhalt ihres FAT-Filesystems anzeigen zu lassen, sind mehrere Befehle notwendig:

```
=> sd status 0

=> sd init 0

=> sd info 0

=> fatstat 0:

=> fatls 0:
```

Wir wollen nun alle SD Low-Level Befehle zu einer Gruppe mit dem Variablennamen **test_sd** zusammenfassen. Dazu wird die Environmentvariable **test_sd** mit dem Befehl

```
=> sentenv  test_sd 'sd status 0; sd init 0; sd info 0'
```

gefolgt vom Befehlsanweisungen angelegt. Eine Befehlskette wird in Hochkomma eingeschlossen und die einzelnen Befehle selbst durch Semikolon getrennt.

Über den Befehl

```
=> printenv
```

kann der Inhalt der Variable überprüft werden. Ist diese korrekt, erfolgt die Befehlsausführung über den Befehl

```
=> run test_sd
```

Hierbei werden nun alle drei Befehle hintereinander abgearbeitet. Den gleichen Weg beschreiben wir nun für den Test des FAT-Filesystems indem wir diese Befehle geeignet zusammenfassen.

```
=> setenv test_fat 'fatstat 0:; fatls 0:'
```

Auch diese Variable kann nun einzeln mit

```
=> run test_fat
```

getestet werden. Waren alle Konfigurationen erfolgreich, werden die neu angelegten Environmentvariablen mit dem Befehl

```
=> saveenv
```

dauerhaft im EEPROM gespeichert. Die komplette Überprüfung der SD-Card kann anschließend mit dem einfachen Befehl

```
=> run test_sd test_fat
```

ausgeführt werden.

Einige Befehle legen selbst eine Environmentvariable an. So lädt der Befehl

```
=> loadc
```

das CP/M 3 BIOS vom FAT-Filesystem der SD-Card und erzeugt gleichzeitig die Environmentvariable **startaddress** mit der Kaltstarteinsprungadresse des CP/M. Diese Adresse kann nun dem **go** Befehl als Startadresse mitgegeben werden:

```
=> go ${startaddress}
```

Die Variable **bootcmd** hat für den Monitor eine besondere Bedeutung. Sie wird während des Autobootvorganges des Monitors automatisch ausgeführt. Hier können also alle Befehlssequenzen für einen kompletten CP/M Neustart vereinbart und ausgeführt werden. Mit

```
=> setenv bootcmd 'reset; loadf; loadc; go ${startaddress}'
```

```
=> run bootcmd
```

startet also das CP/M auf der Default-Konsole. Soll die Konsolenausgabe direkt im Monitor erfolgen, so ist **bootcmd** wie folgt zu setzen:

```
=> setenv bootcmd 'reset; loadf; loadc; go ${startaddress}; connect'
```

Im CP/M 3 muss natürlich `PROFILE.SUB` entsprechend angepasst werden.

`PROFILE.SUB`

```
devive CON:=AVRCON
```