halec
Herrnröther Str. 54
63303 Dreieich
Germany

www.halec.de

Handbuch

roloFlash AVR Basisversion

Dokumentenversion 1.0.1 vom 2012-02-24
(Stand der Software: 1.7)

Inhaltsverzeichnis

I Vorwort

- Mit roloFlash können Sie mobil und unabhängig vom PC Ihre Produkte mit Atmel-AVR-Mikrocontroller flashen.
- Sie können ungelerntes Personal oder Kunden Ihre Produkte flashen lassen, da Fehlbedienungen prinzipbedingt ausgeschlossen sind.
- Dazu sind kein PC und keine spezifischen Tool-Chains (z. B. von Atmel) nötig.
- Nutzen Sie roloFlash im Feldeinsatz, in Ihrem Kundenumfeld bzw. zur Serien- und Kleinserienfertigung.
- Gewinnen Sie Freiräume, indem Sie auf einen einheitlichen Prozeß für alle unterstützten Mikrocontroller-Familien* zurückgreifen.

Begriff „Target“

Unter „Target“ verstehen wir Ihre zu flashenden Produkte. Die Produkte enthalten den zu flashenden Mikrocontroller. Diesen Begriff verwenden wir von nun an regelmäßig in diesem Dokument.

* Zur Zeit Atmel-AVR-Serie, weitere Mikrocontroller-Familien auf Anfrage.

II VERPACKUNGSINHALT

Bitte überprüfen Sie sorgfältig den Lieferumfang:

- roloFlash AVR
- microSD-Karte
  - vorbereitet für den Einsatz in Ihrem roloFlash, mit Dokumentation, Beispielen, Firmware und roloBasic-Compiler
  - zum Einlegen in roloFlash
- Adapter für microSD-Karte
  - zur Verwendung an PCs, die nur über einen SD-Karten- Anschluß verfügen.

Hinweis: Die microSD-Karte befindet sich entweder im roloFlash oder Adapter eingesteckt oder liegt nebenbei.

III BESCHREIBUNG

ISP-Buchse:

Diese Buchse wird auf den ISP-Stecker des zu programmierenden Targets gesteckt. Die Spezifikation des ISP-Steckers finden Sie in den Dokumenten von Atmel (z. B. Beschreibung für STK500).

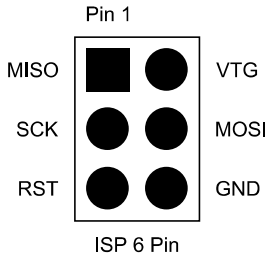

***Abb.:** Draufsicht Pinbelegung **ISP-Stecker** auf der Target-Platine*

LEDs:

Fünf programmierbare zweifarbige (rot und grün) LEDs. Mit den LEDs können Sie z. B.

- ein grünes Lauflicht laufen lassen, das den Flashvorgang darstellt.
- mit rot Fehlermeldungen ausgeben.

microSD-Kartenslot:

Für eine microSD- oder microSDHC-Karte, die das abzuarbeitende Skript (RUN.BIN) sowie die zu flashenden Images enthält.

ACHTUNG: Bitte verwenden Sie ausschließlich microSD-Karten, die vom Hersteller aus mit einem Adapter auf SD-Karte geliefert werden. Nur bei diesen microSD-Karten ist gewährleistet, daß sie das benötigte Protokoll beherrschen. Des weiteren wird empfohlen, ausschließlich microSD-Karten von Kingston oder SanDisk zu verwenden.

IV Ablauf/Verwendung

Der Ablauf gliedert sich in zwei Teile:

- Vorbereitung der microSD-Karte am PC (z. B. in der Entwicklung)
- Flashen der Targets (z. B. ungeschultes Personal in der Produktion, Kunde oder Techniker im Feldeinsatz)

1 Vorbereitung der microSD-Karte am PC (z. B. in der Entwicklung)

Maßgeblich ist immer die Datei „RUN.BIN“, die der roloFlash zum Flashen abarbeitet.

- Falls Sie eine microSD-Karte formatieren wollen, benutzen Sie dazu Windows 7 (Windows XP ist nicht geeignet).
- Sie erstellen den gewünschten Ablauf in roloBasic. Dazu können Sie ein Beispielskript verwenden oder anpassen. Im Kapitel VII „Liste der Prozeduren und Funktionen“ finden Sie die entsprechende Liste, die Ihnen roloFlash zur Verfügung stellt. Ihre erzeugte Datei sollte die Dateiendung „.BAS“ haben.
- Ihr Skript kann dabei auf übliche „.HEX“-Dateien (Intel-Hex-Format: "INTEL 16" / "I16HEX") verweisen, die auf das Target geflasht werden sollen.
- Sie rufen den Compiler „rbc.exe“ auf. Der Compiler erzeugt eine gleichnamige kompilierte Datei mit der Endung „.BIN“.
- Sie benennen die Datei um in „RUN.BIN“ oder rufen statt „rbc.exe“ die Batchdatei `compile.bat` auf, welche aus „RUN.BAS“ ein „RUN.BIN“ erzeugt. Danach legen Sie die Datei „RUN.BIN“ zusammen mit den vom Skript aus benötigten Dateien (in der Regel eine „.HEX“-Datei) auf der microSD-Karte ab.

Sie können die Dateien mit den Skripten („.BAS“), die kompilierten Dateien („.BIN“) und den Compiler nach eigenem Ermessen auf dem PC und/oder auf der microSD-Karte speichern. Für den roloFlash ist lediglich die Datei „RUN.BIN“ (sowie die durch den Code referenzierten Dateien) relevant.

2 Flashen der Targets (z. B. ungeschultes Personal in der Produktion)

Hier ist der Ablauf denkbar einfach:

- Target mit Energie versorgen.
- roloFlash auf den 6-poligen ISP-Stecker (gemäß Atmel-Beschreibung) aufstecken.
- roloFlash wird vom Target mit Energie versorgt und beginnt automatisch mit der Abarbeitung der Datei „RUN.BIN“. Hierdurch wird üblicherweise das Flashen vorgenommen. Währenddessen kann z. B. ein grünes Lauflicht den Flashvorgang symbolisieren.
- roloFlash abziehen – fertig.

v AKTUALISIEREN VON ROLOFLASH

roloFlash verfügt selbst über eine eigene Firmware, die aktualisiert werden kann.

Starten der Aktualisierung

- Zum Aktualisieren muß sich die Firmware-Datei „RFAVRBAS.HMP“ im Hauptverzeichnis der microSD-Karte befinden.
- Das Aktualisieren wird ausgelöst, wenn
  - der roloFlash **ohne** microSD-Karte auf ein beliebiges Target aufgesteckt wird
  - oder eine vorherige Aktualisierung fehlgeschlagen war. In diesem Fall ist es unerheblich, ob erst der roloFlash auf ein Target aufgesteckt wird und dann die microSD-Karte eingesteckt wird oder die microSD-Karte schon eingesteckt ist.
- Es erfolgt keine Überprüfung, ob die Firmware auf der microSD-Karte neuer oder älter ist. Damit ist es auch möglich, zu einer älteren Version zurückzukehren, falls das erforderlich sein sollte.

Der Vorgang des Aktualisierens

- Das Target dient dabei nur zur Energieversorgung.
- Der Vorgang wird mittels der LEDs angezeigt, siehe Kapitel VIII, 2. „roloFlash-Aktualisierung“.
- Solange die microSD-Karte noch nicht eingesteckt ist, leuchtet LED 1 rot.
- Während der Aktualisierung blinken zusätzlich LED 2 und 3 schnell. Der roloFlash sollte jetzt nicht abgezogen werden.

- Bei Erfolg leuchten anschließend LED 1 und 2 grün.
- Der roloFlash bleibt in diesem Zustand, bis er abgezogen wird.
- Die aktualisierte Firmware steht jetzt zur Verfügung.

Falls die Aktualisierung nicht erfolgreich gewesen sein sollte, verwenden Sie bitte eine frisch unter Windows 7 mit FAT32 formatierte microSD-Karte, auf der sich ausschließlich die Datei „RFAVRBAS.HMP“ befindet.

Für eine Produktion wird empfohlen, die Datei „RFAVRBAS.HMP“ nicht auf der microSD-Karte zu belassen.

VI Liste der mitgelieferten roloBasic-Skripte

NORMAL.BAS

- Prüft, ob der Controller die richtige Signature hat
- Prüft den Spannungsbereich
- Löscht den Chip
- Setzt die Fuses
- Beschreibt und verifiziert das Flash mit `IMAGE.HEX`
- Währenddessen grünes Lauflicht, am Ende bleibt LED 5 auf grün an bei Erfolg
- Schreibt das Ergebnis ins Log-File

VERSIONS.BAS

- Schreibt alle Versionsnummern in das Log-File:
  companyName
  deviceName
  softwareVersion
  hardwareVersion
  bootloaderVersion
  imageVersion

VII Liste der Prozeduren und Funktionen

Prozeduren:

Prozeduren haben keinen Rückgabewert. Die übergebenen Parameter müssen ohne Klammern angegeben werden.

Beispiel:

```
setProgrammingSpeed 1000
```

Funktionen:

Funktionen haben einen Rückgabewert. Die übergebenen Parameter müssen in Klammer gesetzt werden.

Beispiel:

```
f = readFuses(fuses_low)
```

Hat die Funktion keine Parameter, dann können die Klammern weggelassen werden

Beispiel:

```
value = getTargetPresent
```

oder

```
value = getTargetPresent()
```

1 Target allgemein

Das Target kann sich in folgenden Modi befinden:

RunMode

Target läuft ganz normal, als ob roloFlash nicht angeschlossen wäre.

ProgramMode

Target kann programmiert werden.

Manche Prozeduren wechseln den Mode (`programTarget` und `runTarget`).

Andere Prozeduren bzw. Funktionen sind auf einen bestimmten Mode angewiesen. In diesem Fall steht das in der jeweiligen Beschreibung.

1.1 getTargetPresent

```
value = getTargetPresent
```

Ermittelt, ob ein Target angeschlossen ist. Der Mode bleibt dabei unverändert.

Falls das Target im RunMode ist, dann wir das Target vorübergehend in den ProgramMode geschaltet. Ein evtl. auf dem Target laufendes Programm wird dadurch neu gestartet.

Anmerkung:

Bei roloFlash sollte immer ein Target angeschlossen sein, weil es sonst keine Energie gäbe. Die Funktion ist hauptsächlich für Programmiergeräte gedacht, die über eine eigene Energieversorgung verfügen.

Des weiteren ist es denkbar, daß roloFlash auf etwas anderes als ein Target aufgesteckt wird. Daher baut diese Funktion tatsächlich eine Kommunikation mit dem Target auf.

Vorbedingung:

- keine

Parameter:

- keine

Rückgabewert:

0 = kein Target gefunden

1 = Target gefunden

Exceptions:

- keine

1.2 programTarget

`programTarget`

- Stoppt das Target.
- Setzt das Target und den roloFlash in den ProgramMode.

Vorbedingung:

- keine

Parameter:

- keine

Rückgabewert:

- keiner (Prozedur)

Exceptions:

`targetCommunication`	Die Kommunikation mit dem Target funktioniert nicht.

1.3 runTarget

`runTarget`

- Startet das Target.
- Setzt roloFlash in den RunMode.

Vorbedingung:

- keine

Parameter:

- keine

Rückgabewert:

- keiner (Prozedur)

Exceptions:

- keine

1.4 restartTarget

`restartTarget`

Das Target wird neu gestartet, der Reset-Zyklus wird durchlaufen. Wenn das Target im ProgramMode war, wird der ProgramMode wieder hergestellt. Zwischenzeitlich kann eine evtl. auf dem Target befindliche Firmware bereits für kurze Zeit losgelaufen sein.

Es wird empfohlen, dieses Kommando nur einzusetzen, wenn dieses entweder nicht kritisch ist, oder sich noch keine Firmware auf dem Target befindet.

Die Prozedur wird benötigt, wenn man Fuses ändert und die Änderungen sofort aktiviert werden sollen. Das gilt insbesondere für das Aktivieren ei-

nes Quarzes auf der Targetplatine, was dann anschließend eine höhere Programmiergeschwindigkeit ermöglicht.

Ein häufiger Ablauf:

```
writeFuses(f)   ! zum Aktivieren des Quarzes
restartTarget
setSpeed 1000   ! 1 MHz
writeFileToFlash 0, "IMAGE.HEX"
```

Vorbedingung:

- keine

Parameter:

- keine

Rückgabewert:

- keiner (Prozedur)

Exceptions:

`targetCommunication`	Die Kommunikation mit dem Target funktioniert nicht.

1.5 setProgrammingSpeed

`setProgrammingSpeed <speed>`

Setzt die Programmiergeschwindigkeit für das Target.

Vorbedingung:

Das Target muß im ProgramMode sein.

Parameter:

speed

Angabe in kHz. Unterstützte Werte sind:
4000, 2000, 1000, 500, 250, 125 kHz
`speed_min`: Setzt die kleinste Geschwindigkeit (also 125 kHz).
Falls die angegebene Frequenz nicht in dieser Liste ist, dann wird auf den nächsten möglichen Wert abgerundet.

Rückgabewert:

- keiner (Prozedur)

Exceptions:

`targetWrongMode`	Target ist nicht im "ProgramMode"
`rangeValue`	Unzulässiger Wert für speed.
`typeFault`	Unzulässiger Typ für speed.

1.6 getTargetVoltage

```
u = getTargetVoltage
```

Ermittelt die Spannung in mV, die das Target liefert.

Vorbedingung:

- keine

Parameter:

- keine

Rückgabewert:

Ausgelesene Spannung in mV.

Exceptions:

- keine

2 Target-Fuses & -Lock-Bits

2.1 readBits

```
values = readBits(index)
```

Liest die angegebenen Fuses bzw. Lock-Bits aus.

Vorbedingung:

Das Target muß im ProgramMode sein.

Parameter:

index

Gibt an, welche Fuses bzw. Lock-Bits gelesen werden sollen. Es gibt dazu die Konstanten `fuses_low`, `fuses_high`, `fuses_ext` und `lock_bits`.

Bei Controllern, die keine Extended-Fuses haben, ist der ausgelesene Wert bei `fuses_ext` unbestimmt (es wird keine Exception erzeugt).

Rückgabewert:

Ausgelesene Fuses bzw. Lock-Bits.

Exceptions:

`targetWrongMode`	Target ist nicht im "ProgramMode".
`targetCommunication`	Die Kommunikation mit dem Target funktioniert nicht.
`rangeValue`	Unzulässiger Wert für index.
`typeFault`	Unzulässiger Typ für index.

2.2 writeBits

```
writeBits index, values
```

Schreibt die angegebenen Fuses bzw. Lock-Bits.

Vorsicht:

- Setzen Sie die Lockbit erst, nachdem Sie alle anderen Zugriffe auf den Chip ausgeführt haben.
- Falls Sie einen durch Lockbits gesperrten Chip bearbeiten wollen, führen Sie als erstes ein `eraseFlash` aus. Dieses setzt auch die Lockbits wieder zurück.

Vorbedingung:

Das Target muß im ProgramMode sein.

Parameter:

index

Gibt an, welche Fuses bzw. Lock-Bits beschrieben werden sollen. Es gibt dazu die Konstanten `fuses_low`, `fuses_high`, `fuses_ext` und `lock_bits`.

Bei Controllern, die keine Extended-Fuses haben, findet bei `fuses_ext` kein Schreibvorgang statt (es wird keine Exception erzeugt).

values

Zu schreibende Werte.

Rückgabewert:

- keiner (Prozedur)

Exceptions:

`targetWrongMode`	Target ist nicht im "ProgramMode".
`targetCommunication`	Die Kommunikation mit dem Target funktioniert nicht.
`rangeValue`	Unzulässiger Wert für index oder value.
`typeFault`	Unzulässiger Typ für index oder value.

3 Target Signature und Speicherlayout

3.1 getSignature

```
s = getSignature()
```

Liest die Signature des Targets. Anhand dieser lassen sich die verschiedenen Controller unterscheiden.

Falls der verwendete Controller bekannt ist, dann wird das Flash-

und EEPROM-Layout automatisch mit eingestellt.

Falls der verwendete Controller roloFlash nicht bekannt ist, dann muß das Flash- und EEPROM-Layout mittels `setFlashLayout und setEepromLayout` manuell eingestellt werden.

Ob der verwendete Controller roloFlash bekannt ist oder nicht, kann durch Aufruf von `getFlashLayout` ermittelt werden. Falls der Con-

troller nicht bekannt ist, dann liefert `getFlashLayout` die Exception "unknownMemoryLayout".

Vorbedingung:

Das Target muß im ProgramMode sein.

Parameter:

- keine

Rückgabewert:

Ausgelesene Signature. Die Signature wird in einem Byte-Array mit 3 Bytes geliefert.
Für bekannte Targets gibt es passende Konstanten, so daß Sie mit folgender Zeile feststellen können, ob das Target der erwartete Typ ist:

```
if getSignature() = sig_atmega32 ! Test auf ATmega32
```

Exceptions:

`targetWrongMode`	Target ist nicht im "ProgramMode".
`targetCommunication`	Die Kommunikation mit dem Target funktioniert nicht.

3.2 getFlashLayout

```
a = getFlashLayout()
```

Liefert die Größe des Flash und die Größe einer Page im Flash. Diese Werte wurden entweder über `setFlashLayout` gesetzt oder bei einem bekannten Controller beim Aufruf von `getSignature()` automatisch gesetzt.

Vorbedingung:

Das Target muß im ProgramMode sein.

Parameter:

- keine

Rückgabewert:

Array mit 2 Elementen:

a[0] = Größe des Flashs in Bytes

a[1] = Größe einer Page im Flash in Bytes

Exceptions:

`targetWrongMode`	Target ist nicht im "ProgramMode".

3.3 setFlashLayout

```
setFlashLayout <size, pagesize>
```

Die Größe des Flashs und die Größe einer Page im Flash werden gesetzt. Die Werte müssen passend zum Controller der Atmel-Dokumentation entnommen werden.

Falls der verwendete Controller bekannt ist, dann werden die passenden Werte beim Aufruf von `getSignature()` automatisch ermittelt. Der Aufruf dieser Funktion ist dann nicht nötig und wird nicht empfohlen.

Vorbedingung:

Das Target muß im ProgramMode sein.

Parameter:

size

Größe des Flashs in Bytes.

pagesize

Größe einer Page im Flash in Bytes.

Rückgabewert:

- keiner (Prozedur)

Exceptions:

`targetWrongMode`	Target ist nicht im "ProgramMode".

3.4 getEepromLayout

```
a = getEepromLayout()
```

Liefert die Größe des EEPROMs und die Größe einer Page im EEPROM. Diese Werte wurden entweder über `setEepromLayout` gesetzt oder bei einem bekannten Controller beim Aufruf von `getSignature()` automatisch gesetzt.

Vorbedingung:

Das Target muß im ProgramMode sein.

Parameter:

- keine

Rückgabewert:

Array mit 2 Elementen:

a[0] = Größe des EEPROMs in Bytes

a[1] = Größe einer Page im EEPROM in Bytes

Exceptions:

`targetWrongMode`	Target ist nicht im "ProgramMode".

3.5 setEepromLayout

`setEepromLayout <size, pagesize>`

Die Größe des EEPROMs und die Größe einer Page im EEPROM werden gesetzt. Die Werte müssen passend zum Controller der Atmel-Dokumentation entnommen werden.

Falls der verwendete Controller bekannt ist, dann werden die passenden Werte beim Aufruf von `getSignature()` automatisch ermittelt. Der Aufruf dieser Funktion ist dann nicht nötig und wird nicht empfohlen.

Vorbedingung:

Das Target muß im ProgramMode sein.

Parameter:

size

Größe des EEPROMs in Bytes.

pagesize

Größe einer Page im EEPROM in Bytes.

Rückgabewert:

- keiner (Prozedur)

Exceptions:

`targetWrongMode`	Target ist nicht im "ProgramMode".

3.6 setExtendedAddressMode

`setExtendedAddressMode <value>`

Für Controller mit 256 kB Flash oder mehr ist der normale Befehlssatz zum Programmieren über das ISP-Interface nicht mehr ausreichend. Es wird dann ein extended address mode benötigt.

Falls der verwendete Controller bekannt ist, dann wird der passende Wert beim Aufruf von `getSignature()` automatisch ermittelt. Der Aufruf dieser Funktion ist dann nicht nötig.

Vorbedingung:

Das Target muß im ProgramMode sein.

Parameter:

value

value = 0: Extended address mode nicht verwenden

value = 1: Extended address mode verwenden

Rückgabewert:

- keiner (Prozedur)

Exceptions:

`targetWrongMode`	Target ist nicht im "ProgramMode".
`rangeValue`	Unzulässiger Wert für value.
`typeFault`	Unzulässiger Typ für value.

3.7 clearMemoryLayout

`clearMemoryLayout`

Löscht ein bereits vorhandenes Speicherlayout (Flash- und EEPROM-Layout).

Vorbedingung:

- keine

Parameter:

- keine

Rückgabewert:

- keiner (Prozedur)

Exceptions:

- keine

4 Target schreiben und verifizieren mit Hex-Dateien

4.1 EraseFlash

`eraseFlash`

Löscht das gesamte Flash des Targets. Je nach Einstellung in den Fuses (EESAVE) wird dabei das EEPROM auch gelöscht (default) oder nicht.

Vorbedingung:

Das Target muß im ProgramMode sein.

Parameter:

- keine

Rückgabewert:

- keiner (Prozedur)

Exceptions:

`targetWrongMode`	Target ist nicht im "ProgramMode".
`targetCommunication`	Die Kommunikation mit dem Target funktioniert nicht.

4.2 writeFileToFlash

`writeFileToFlash <filesystem, filename>`

Schreibt eine Hex-Datei ins Flash des Targets.

Vorbedingung:

Das Target muß im ProgramMode sein.

Parameter:

filesystem

Der Parameter wird ignoriert und sollte mit 0 angegeben werden.

filename

Es gelten die Bedingungen für Dateinamen, siehe Kapitel 5 “Dateien“.

Rückgabewert:

- keiner (Prozedur)

Exceptions:

`targetMemoryLayout,` `targetExtendedAddress,` `hexFileSize,` `hexFileCRC`	Siehe Kapitel „Exceptions des roloFlashs“.
`targetWrongMode`	Target ist nicht im "ProgramMode".
`targetCommunication`	Die Kommunikation mit dem Target funktioniert nicht.
`typeFault`	Unzulässiger Typ für filename.

`<diverse Exceptions des Dateisystems>`	Siehe Kapitel „Exceptions des Dateisystems“.

4.3 verifyFileToFlash

`verifyFileToFlash <filesystem, filename>`

Vergleicht eine Hex-Datei mit den Daten im Flash.

Vorbedingung:

Das Target muß im ProgramMode sein.

Parameter:

filesystem

Der Parameter wird ignoriert und sollte mit 0 angegeben werden.

filename

Es gelten die Bedingungen für Dateinamen, siehe Kapitel 5 “Dateien“.

Rückgabewert:

- keiner (Prozedur). Falls die Daten unterschiedlich sind, wird eine Exception erzeugt.

Exceptions:

`targetVerify`	Beim Verify wurden andere Daten gelesen. Siehe Kapitel „Exceptions des roloFlashs“.
`targetMemoryLayout, targetExtendedAddress, hexFileSize, hexFileCRC`	
`targetWrongMode`	Target ist nicht im "ProgramMode".
`targetCommunication`	Die Kommunikation mit dem Target funktioniert nicht.

`typeFault`	Unzulässiger Typ für filename.
`<diverse Exceptions des Dateisystems>`	Siehe Kapitel „Exceptions des Dateisystems“.

4.4 writeVerifyFileToFlash

`writeVerifyFileToFlash <filesystem, filename>`

Schreibt eine Hex-Datei ins Flash des Targets und vergleicht das Geschriebene mit der Hex-Datei.

Die Wirkung ist identisch mit dem Aufruf von `writeFileToFlash` und anschließendem `verifyFileToFlash`, kann aber schneller sein.

Vorbedingung:

Das Target muß im ProgramMode sein.

Parameter:

filesystem

Der Parameter wird ignoriert und sollte mit 0 angegeben werden.

filename

Es gelten die Bedingungen für Dateinamen, siehe Kapitel 5 “Dateien“.

Rückgabewert:

- keiner (Prozedur). Falls die Daten unterschiedlich sind, wird eine Exception erzeugt.

Exceptions:

`targetVerify`	Beim Verify wurden andere Daten gelesen.
`targetMemoryLayout, targetExtendedAddress, hexFileSize, hexFileCRC`	Siehe Kapitel „Exceptions des roloFlashs“.

`targetWrongMode`	Target ist nicht im "ProgramMode".
`targetCommunication`	Die Kommunikation mit dem Target funktioniert nicht.
`typeFault`	Unzulässiger Typ für filename.
`<diverse Exceptions des Dateisystems>`	Siehe Kapitel „Exceptions des Dateisystems“.

4.5 writeFileToEeprom

`writeFileToEeprom <filesystem, filename>`

Schreibt eine Hex-Datei ins EEPROM des Targets.

Vorbedingung:

Das Target muß im ProgramMode sein.

Parameter:

filesystem

Der Parameter wird ignoriert und sollte mit 0 angegeben werden.

filename

Es gelten die Bedingungen für Dateinamen, siehe Kapitel 5 “Dateien“.

Rückgabewert:

- keiner (Prozedur)

Exceptions:

`targetMemoryLayout,` `targetExtendedAddress,` `hexFileSize,` `hexFileCRC`	Siehe Kapitel „Exceptions des roloFlashs“.
`targetWrongMode`	Target ist nicht im "ProgramMode".
`targetCommunication`	Die Kommunikation mit dem Target funktioniert nicht.

`typeFault`	Unzulässiger Typ für filename.
`<diverse Exceptions des Dateisystems>`	Siehe Kapitel „Exceptions des Dateisystems“.

4.6 verifyFileToEeprom

`verifyFileToEeprom <filesystem, filename>`

Vergleicht eine Hex-Datei mit den Daten im EEPROM.

Vorbedingung:

Das Target muß im ProgramMode sein.

Parameter:

filesystem

Der Parameter wird ignoriert und sollte mit 0 angegeben werden.

filename

Es gelten die Bedingungen für Dateinamen, siehe Kapitel 5 “Dateien“.

Rückgabewert:

- keiner (Prozedur). Falls die Daten unterschiedlich sind, wird eine Exception erzeugt.

Exceptions:

`targetVerify`	Beim Verify wurden andere Daten gelesen.
`targetMemoryLayout, targetExtendedAddress, hexFileSize, hexFileCRC`	Siehe Kapitel „Exceptions des roloFlashs“.
`targetWrongMode`	Target ist nicht im "ProgramMode".
`targetCommunication`	Die Kommunikation mit dem Target

	funktioniert nicht.
`typeFault`	Unzulässiger Typ für filename.
`<diverse Exceptions des Dateisystems>`	Siehe Kapitel „Exceptions des Dateisystems“.

4.7 writeVerifyFileToEeprom

`writeVerifyFileToEeprom <filesystem, filename>`

Schreibt eine Hex-Datei ins EEPROM des Targets und vergleicht das Geschriebene mit der Hex-Datei.

Die Wirkung ist identisch mit dem Aufruf von `writeFileToEeprom` und anschließendem `verifyFileToEeprom`, kann aber schneller sein.

Vorbedingung:

Das Target muß im ProgramMode sein.

Parameter:

filesystem

Der Parameter wird ignoriert und sollte mit 0 angegeben werden.

filename

Es gelten die Bedingungen für Dateinamen, siehe Kapitel 5 “Dateien“.

Rückgabewert:

- keiner (Prozedur). Falls die Daten unterschiedlich sind, wird eine Exception erzeugt.

Exceptions:

`targetVerify`	Beim Verify wurden andere Daten gelesen.
`targetMemoryLayout,` `targetExtendedAddress,` `hexFileSize,`	Siehe Kapitel „Exceptions des roloFlashs“.

`hexFileCRC`	
`targetWrongMode`	Target ist nicht im "ProgramMode".
`targetCommunication`	Die Kommunikation mit dem Target funktioniert nicht.
`typeFault`	Unzulässiger Typ für filename.
`<diverse Exceptions des Dateisystems>`	Siehe Kapitel „Exceptions des Dateisystems“.

5 Dateien

Dateinamen:

- Dateinamen müssen der 8.3-Regel folgen: „`XXXXXXXX.YYY`“.
- Es sind nur die Zeichen „A“ - “Z“, “0“ - “9“, sowie „_“ und „-“ zulässig.
- Es dürfen nur Großbuchstaben verwendet werden.

Verzeichnisnamen:

- Verzeichnisnamen dürfen maximal aus acht Zeichen bestehen: „`XXXXXXXX`“.
- Ansonsten gelten dieselben Konventionen wie bei Dateinamen.

Aktuelles Verzeichnis ist immer das Hauptverzeichnis:

- Es gibt kein „change directory“. Der aktuelle Pfad bleibt immer das Hauptverzeichnis. Ein Dateiname muß daher immer den kompletten Pfad beinhalten.
- Als Trennzeichen zwischen Verzeichnissen bzw. Dateiname werden „/“ und „\“ unterstützt.

5.1 fsCreate

```
fsCreate <filesystem, filename>
```

Erzeugt die angegebene Datei. Die Datei ist danach noch immer geschlossen. Falls die Datei schon existiert, dann hat die Prozedur keine Wirkung.

Wenn man eine Datei erzeugen und in diese etwas schreiben möchte, dann muß man die Datei zusätzlich noch öffnen:

```
fsCreate 0, "TEST.TXT"
handle = fsOpen(0, "TEST.TXT")
```

Vorbedingung:

- keine

Parameter:

filesystem

Der Parameter wird ignoriert und sollte mit 0 angegeben werden.

filename

Es gelten die Bedingungen für Dateinamen, siehe Kapitel 5 “Dateien“.

Rückgabewert:

- keiner (Prozedur)

Exceptions:

`typeFault`	Unzulässiger Typ für filename.
`<diverse Exceptions des Dateisystems>`	Siehe Kapitel „Exceptions des Dateisystems“.

5.2 fsRemove

```
fsRemove <filesystem, filename>
```

Löscht die angegebene Datei oder das angegebene Verzeichnis, falls vorhanden.

Vorbedingung:

- keine

Parameter:

filesystem

Der Parameter wird ignoriert und sollte mit 0 angegeben werden.

filename

Es gelten die Bedingungen für Verzeichnis- bzw. Dateinamen, siehe Kapitel 5 “Dateien“.

Rückgabewert:

- keiner (Prozedur)

Exceptions:

`fileNotFound`	Die angegebene Datei existiert nicht.
`typeFault`	Unzulässiger Typ für filename.
`<diverse Exceptions des Dateisystems>`	Siehe Kapitel „Exceptions des Dateisystems“.

5.3 fsMkDir

```
fsMkDir <filesystem, dirname>
```

Erzeugt das angegebene Verzeichnis. Falls das Verzeichnis schon existiert, dann hat die Prozedur keine Wirkung.

Vorbedingung:

- keine

Parameter:

filesystem

Der Parameter wird ignoriert und sollte mit 0 angegeben werden.

dirname

Es gelten die Bedingungen für Verzeichnisnamen, siehe Kapitel 5 “Dateien“.

Rückgabewert:

- keiner (Prozedur)

Exceptions:

`typeFault`	Unzulässiger Typ für dirname.
`<diverse Exceptions des Dateisystems>`	Siehe Kapitel „Exceptions des Dateisystems“.

5.4 fsFileExists

```
bool fsFileExists(filesystem, filename)
```

Prüft, ob die angegebene Datei existiert.

Vorbedingung:

- keine

Parameter:

filesystem

Der Parameter wird ignoriert und sollte mit 0 angegeben werden.

filename

Es gelten die Bedingungen für Dateinamen, siehe Kapitel 5 “Dateien“.

Rückgabewert:

0 = Datei existiert nicht

1 = Datei existiert

Exceptions:

`typeFault`	Unzulässiger Typ für filename.
`<diverse Exceptions des Dateisystems>`	Siehe Kapitel „Exceptions des Dateisystems“.

5.5 fsFilesize

```
size = fsFilesize(filesystem, filename)
```

Ermittelt die Größe der angegebenen Datei.

Vorbedingung:

- Datei existiert.

Parameter:

filesystem

Der Parameter wird ignoriert und sollte mit 0 angegeben werden.

filename

Es gelten die Bedingungen für Dateinamen, siehe Kapitel 5 “Dateien“.

Rückgabewert:

Es wird die Dateigröße zurückgegeben.

Exceptions:

`typeFault`	Unzulässiger Typ für filename.
`<diverse Exceptions des Dateisystems>`	Siehe Kapitel „Exceptions des Dateisystems“.

5.6 fsOpen

```
filehandle = fsOpen(filesystem, filename)
```

Öffnet die angegebene Datei.

Vorbedingung:

Die Datei muß bereits existieren. Soll eine neue Datei geöffnet werden, dann muß vorher `fsCreate` verwendet werden.

Parameter:

filesystem

Der Parameter wird ignoriert und sollte mit 0 angegeben werden.

filename

Es gelten die Bedingungen für Dateinamen, siehe Kapitel 5 “Dateien“.

Rückgabewert:

Es wird ein Filehandle zum Zugriff auf die Datei (z. B. für `fsRead` und `fsWrite`) zurückgegeben. Das Filehandle wird außerdem zum Schließen der Datei (`fsClose`) benötigt.

Exceptions:

`typeFault`	Unzulässiger Typ für filename.
`<diverse Exceptions des Dateisystems>`	Siehe Kapitel „Exceptions des Dateisystems“.

5.7 fsRead

```
a = fsRead(filehandle, position, count)
```

Liest die angegebene Anzahl an Bytes aus der Datei.

Vorbedingung:

- Gültiges `Filehandle` mittels `fsOpen`.

Parameter:

filehandle

Das von `fsOpen` erhaltene Filehandle.

position

Die Byte-Position, von der gelesen werden soll.

count

Die Anzahl der zu lesenden Bytes.

Rückgabewert:

Array of Byte mit den gelesenen Daten. Das Array hat die Größe von `count`. Falls nicht mehr genügend Daten zu lesen waren, dann ist das Array entsprechend kleiner. Wird am Dateiende oder darüber hinaus versucht zu lesen, dann wir ein leeres Array mit der Größe 0 zurückgegeben.

Exceptions:

`rangeValue`	Unzulässiger Wert für filehandle, position oder count.
`typeFault`	Unzulässiger Typ für filehandle, position oder count.
`<diverse Exceptions des Dateisystems>`	Siehe Kapitel „Exceptions des Dateisystems“.

5.8 fsWrite

`fsWrite <filehandle, position, array>`

Schreibt die übergebenen Daten in die Datei.

Sollte die Position außerhalb der momentanen Dateigröße sein, dann wird die Datei bis zu dieser Position mit zufälligen Werten aufgefüllt.

Vorbedingung:

- Gültiges Filehandle mittels `fsOpen`.

Parameter:

filehandle

Das von `fsOpen` erhaltene Filehandle.

position

Die Byte-Position, an die geschrieben werden soll.

array

Array of Byte mit den zu schreibenden Daten.

Rückgabewert:

- keiner (Prozedur)

Exceptions:

`rangeValue`	Unzulässiger Wert für filehandle, position oder count.
`typeFault`	Unzulässiger Typ für filehandle, position oder count.
`<diverse Exceptions des Dateisystems>`	Siehe Kapitel „Exceptions des Dateisystems“.

5.9 fsTruncate

```
fsTruncate <filehandle, len>
```

Kürzt die Datei auf die angegebene Länge. Falls die Datei schon kleiner ist, dann ist die Prozedur ohne Wirkung.

Vorbedingung:

- Gültiges Filehandle mittels `fsOpen`.

Parameter:

filehandle

Das von `fsOpen` erhaltene Filehandle.

len

Die Länge, auf die die Datei gekürzt wird.

Rückgabewert:

- keiner (Prozedur)

Exceptions:

`rangeValue`	Unzulässiger Wert für filehandle.
`typeFault`	Unzulässiger Typ für filehandle oder len.
`<diverse Exceptions des Dateisystems>`	Siehe Kapitel „Exceptions des Dateisystems“.

5.10 fsClose

```
fsClose <filehandle>
```

Schließt die Datei. Das angegebene Filehandle wird dadurch ungültig und darf nicht mehr verwendet werden.

Vorbedingung:

- Gültiges Filehandle mittels `fsOpen`.

Parameter:

filehandle

Das von `fsOpen` erhaltene Filehandle.

Rückgabewert:

- keiner (Prozedur)

Exceptions:

`rangeValue`	Unzulässiger Wert für filehandle.
`typeFault`	Unzulässiger Typ für filehandle.
`<diverse Exceptions des Dateisystems>`	Siehe Kapitel „Exceptions des Dateisystems“.

5.11 fsSync

```
fsSync <filesystem>
```

Stellt sicher, daß alle Daten, die evtl. noch nicht auf die Karte geschrieben wurden, nun geschrieben werden. Es wird empfohlen, wenn Schreibzugriffe auf die Karte stattfinden, diese Prozedur anschließend aufzurufen.

Vorbedingung:

- keine

Parameter:

filesystem

Der Parameter wird ignoriert und sollte mit 0 angegeben werden.

Rückgabewert:

- keiner (Prozedur)

Exceptions:

`<diverse Exceptions des Dateisystems>`	Siehe Kapitel „Exceptions des Dateisystems“.

6 LEDs

Immer nur eine LED an:

- Es ist aus roloBasic heraus nicht möglich, mehr als eine LED gleichzeitig einzuschalten. Dadurch können andere Fehlermeldungen vom System besser erkannt werden, da diese immer mehr als eine LED nutzen.

Numerierung und Farben:

- Die LEDs sind in roloBasic genauso nummeriert wie auf dem Gehäuse abgebildet: von 1 bis 5.
- Die LEDs können in grün oder rot leuchten. Dazu stehen die Konstanten `color_green` und `color_red` zur Verfügung.

Nicht blockiernd:

- Alle Prozeduren dieses Kapitels sind nicht blockierend. Das bedeutet, daß z. B. ein mit `ledRunningLight` aktiviertes Lauflicht parallel zur weiteren Ausführung des roloBasics läuft.

6.1 ledOn

`ledOn <index, color>`

Schaltet die LED auf die angegebene Farbe.

Vorbedingung:

- keine

Parameter:

index

Nummer der LED

color

`color_green` bzw. `color_red`

Rückgabewert:

- keiner (Prozedur)

Exceptions:

`rangeValue`	Unzulässiger Wert für index oder color.
`typeFault`	Unzulässiger Typ für index oder color.

6.2 ledOff

`ledOff`

Schaltet alle LEDs aus.

Vorbedingung:

- keine

Parameter:

- keine

Rückgabewert:

- keiner (Prozedur)

Exceptions:

- keine

6.3 ledBlink

`ledBlink <index, color, speed>`

Die LED blinkt mit der angegebenen Geschwindigkeit.

Vorbedingung:

- keine

Parameter:

index

Nummer der LED

color

`color_green` bzw. `color_red`

speed

Geschwindigkeit des Blinkens in ms

Rückgabewert:

- keiner (Prozedur)

Exceptions:

`rangeValue`	Unzulässiger Wert für index, color oder speed.
`typeFault`	Unzulässiger Typ für index, color oder speed.

6.4 ledRunningLight

`ledRunningLight <from, to, color, speed>`

Läßt ein Lauflicht laufen.

Vorbedingung:

- keine

Parameter:

from, to

Das Lauflicht läuft von LED '`from`' bis LED '`to`'.

Ist '`from`' kleiner als '`to`', dann läuft das Lauflicht anders herum.

Ist '`from`' gleich '`to`', dann leuchtet die eine LED ständig.

color

`color_green` bzw. `color_red`

speed

Geschwindigkeit des Lauflichts in ms

Rückgabewert:

- keiner (Prozedur)

Exceptions:

`rangeValue`	Unzulässiger Wert für from, to, color oder speed.
`typeFault`	Unzulässiger Typ für from, to, color oder speed.

6.5 ledRunningLightOutstanding

```
ledRunningLightOutstanding <from, to, color, speed,
outstandingLedNumber>
```

Läßt ein Lauflicht laufen, bei der eine bestimmte LED die andere Farbe aufweist.

Vorbedingung:

- keine

Parameter:

from, to

Das Lauflicht läuft von LED '`from`' bis LED '`to`'.

Ist '`from`' kleiner als '`to`', dann läuft das Lauflicht anders herum.

Ist '`from`' gleich '`to`', dann leuchtet die eine LED ständig.

color

`color_green` bzw. `color_red`

speed

Geschwindigkeit des Lauflichts in ms

outstandingLedNumber

Nummer der LED, die mit der anderen Farbe aufleuchtet.

Rückgabewert:

- keiner (Prozedur)

Exceptions:

`rangeValue`	Unzulässiger Wert für from, to, color, speed oder outstandingLedNumber.
`typeFault`	Unzulässiger Typ für from, to, color, speed oder outstandingLedNumber.

7 Sonstige

7.1 print

```
print a, b, ...
```

Es werden die Parameter `a, b` etc. ausgedruckt. Die Anzahl der Parameter ist beliebig.

Der Ausdruck erfolgt an das Ende der Datei „LOG.TXT“. Wenn die Datei noch nicht existiert, dann wird sie erzeugt.

Vorbedingung:

- keine

Parameter:

`a, b, ...`

Hier können Zahlen und Arrays ausgeben werden. Beispiel:

```
value = 42
print "Der Wert ist: ", value
```

Ist ein angegebener Parameter weder eine Zahl noch ein Char-Array, dann wird nichts ausgegeben.

Rückgabewert:

- keiner (Prozedur)

Exceptions:

`<diverse Exceptions des Dateisystems>`	Siehe Kapitel „Exceptions des Dateisystems“.

7.2 delay

```
delay <duration>
```

Es wird die angegebene Zeitdauer in ms abgewartet. Erst danach kehrt die Funktion zurück.

Vorbedingung:

- keine

Parameter:

duration

Zeitangabe in ms, die gewartet werden soll.

Rückgabewert:

- keiner (Prozedur)

Exceptions:

`rangeValue`	Unzulässiger Wert für duration.
`typeFault`	Unzulässiger Typ für duration.

VIII KONSTANTEN

1 Versionsnummern etc.

Name	Wert / Bedeutung
`companyName`	„halec <http://halec.de>“
`deviceName`	„roloFlash AVR, base version“
`softwareVersion`	Versionsnummer der Firmware
`hardwareVersion`	Versionsnummer der Hardware
`bootloaderVersion`	Versionsnummer des Bootloaders
`imageVersion`	roloFlash erwartet das vom Compiler erzeugte Image in dieser Version. Bitte verwenden Sie daher den zur Firmware des roloFlashs passenden Compiler.

2 Farben für LEDs

Name	Wert / Bedeutung
`color_green`	1
`color_red`	2

3 Signaturen verschiedener Controller

Name	Wert / Bedeutung
Beispiel: `sig_atmega32`	Array mit [0x1E, 0x95, 0x02]

Folgende Konstanten für die Signaturen sind definiert:

```
sig_at89s51
sig_at89s52
sig_at90can128
```

```
sig_at90can32
sig_at90can64
sig_at90pwm2
sig_at90pwm216
sig_at90pwm2b
sig_at90pwm3
sig_at90pwm316
sig_at90pwm3b
sig_at90pwm81
sig_at90scr100h
sig_at90usb1286
sig_at90usb1287
sig_at90usb162
sig_at90usb646
sig_at90usb647
sig_at90usb82
sig_atmega128
sig_atmega1280
sig_atmega1281
sig_atmega1284p
sig_atmega128a
sig_atmega128rfa1
sig_atmega16
sig_atmega162
sig_atmega164p
sig_atmega164pa
sig_atmega165
sig_atmega165p
sig_atmega168
sig_atmega168p
sig_atmega168pa
sig_atmega169
sig_atmega169p
sig_atmega16a
sig_atmega16hva
sig_atmega16hvb
sig_atmega16m1
sig_atmega16u2
sig_atmega16u4
sig_atmega2560
sig_atmega2561
sig_atmega32
sig_atmega324p
sig_atmega324pa
```

```
sig_atmega325
sig_atmega3250
sig_atmega3250p
sig_atmega325p
sig_atmega328p
sig_atmega329
sig_atmega3290
sig_atmega3290p
sig_atmega329p
sig_atmega32a
sig_atmega32c1
sig_atmega32hvb
sig_atmega32m1
sig_atmega32u2
sig_atmega32u4
sig_atmega32u6
sig_atmega48
sig_atmega48p
sig_atmega48pa
sig_atmega64
sig_atmega640
sig_atmega644
sig_atmega644p
sig_atmega644pa
sig_atmega645
sig_atmega6450
sig_atmega649
sig_atmega6490
sig_atmega64a
sig_atmega64c1
sig_atmega64m1
sig_atmega8
sig_atmega8515
sig_atmega8535
sig_atmega88
sig_atmega88p
sig_atmega88pa
sig_atmega8a
sig_atmega8hva
sig_atmega8u2
sig_attiny13
sig_attiny13a
sig_attiny167
sig_attiny2313
```

```
sig_attiny2313a
sig_attiny24
sig_attiny24a
sig_attiny25
sig_attiny26
sig_attiny261
sig_attiny261a
sig_attiny4313
sig_attiny43u
sig_attiny44
sig_attiny44a
sig_attiny45
sig_attiny461
sig_attiny48
sig_attiny84
sig_attiny85
sig_attiny861
sig_attiny861a
sig_attiny87
sig_attiny88
```

IX EXCEPTIONS

Im Handbuch für das roloBasic finden Sie die genaue Beschreibung, wie Exceptions ausgelöst und wieder gefangen werden können. Wird eine Exception nicht gefangen, dann wird die Exception mittels der LEDs angezeigt.

Falls dabei die Exception keine Zahl darstellt, dann wird die Exception „exceptionNotANumber“ ausgegeben. Details dazu finden Sie unter Kapitel X.1.3 „Exception aufgetreten“. Nur vom Benutzer ausgelöste Exceptions können nicht-numerisch sein.

Es gibt verschiedene Arten von Exceptions, die alle gleich behandelt werden:

- Exceptions des roloBasic
- Exceptions des Dateisystems
- Exceptions des roloFlashs
- Vom Benutzter ausgelöste Exceptions

1 Exceptions des roloBasic

Diese Exceptions treten bei Fehlern auf, die nicht speziell etwas mit roloFlash zu tun haben, sondern mit der Abarbeitung des Basics. Ein typisches Beispiel dazu ist eine outofRange-Exception.

Eine Liste dieser Exceptions finden Sie im Handbuch zu roloBasic.

2 Exceptions des Dateisystems

Diese Exceptions treten bei Fehlern im Zusammenhang mit dem Dateisystem oder mit der microSD-Karte auf.

Name	Nummer	Bedeutung
deviceError	101	Es konnte nicht auf der microSD-Karte gelesen/geschrieben werden
badCluster	102	Probleme innerhalb des Dateisystems. Das Dateisystem sollte auf dem PC auf Konsistenz geprüft werden.
notMounted	103	Es wurde versucht, auf die microSD-Karte zuzugreifen, obwohl sie nicht angemeldet ist. Dieses deutet auf ein Problem mit der microSD-Karte hin.
removeError	104	Die microSD-Karte wurde entfernt.
createError	105	Das Erzeugen einer Datei oder eines Verzeichnisses ist fehlgeschlagen.
fileNotOpen	106	Die Datei ist nicht geöffnet.
fileNotFound	107	Die angegebene Datei oder das Verzeichnis konnte nicht gefunden werden.
diskFull	108	Die microSD-Karte ist voll.
truncateError	109	Das Kürzen einer Datei mittels fsTruncate ist fehlgeschlagen.
illegalCluster	110	Probleme innerhalb des Dateisystems. Das Dateisystem sollte auf dem PC auf Konsistenz geprüft werden.
fileLocked	111	Es wurde versucht, eine bereits geöffnete Datei ein zweites Mal zu öffnen. Evtl. wurde ein fsClose vergessen.
outOfFileHandles	112	Die Anzahl der maximal geöffneten Dateien ist auf 3 begrenzt. Es wurde versucht, eine weitere Datei zu öffnen.

3 Exceptions des roloFlashs

Name	Nummer	Bedeutung

exceptionIsNotANumber	200	Es wurde eine Exception erzeugt und innerhalb des Basics nicht mehr gefangen, die keine Zahl ist. In diesem Fall wird die Original-Exception verworfen und durch diese Exception ersetzt. Das kann nur durch vom Benutzer erzeugte Exceptions auftreten, da alle anderen Funktionen ausschließlich die hier beschriebenen numerischen Exceptions nutzen. Beispiel: `throw "Fehler"`
imageTooLarge	201	Das Basicskript ist zu groß. Es können maximal circa 1024 Bytes geladen werden. Bitte überprüfen Sie die Dateigröße der vom Compiler erzeugten Datei.
imageWrongVersion	202	Der verwendete Compiler paßt nicht zur Firmware auf dem roloFlash. Es wird empfohlen, immer jeweils den neuesten Compiler und Firmware für den roloFlash zu verwenden.
productWrongVersion	203	Es wurde versucht, ein Image eines anderen Produktes auf den roloFlash zu laden. Beispielsweise wurde versucht, ein Image für roloFlash in der Industrieversion auf einen roloFlash in der Basisversion zu landen.
imageNotFound	204	Die microSD-Karte konnte zwar eingebunden werden, jedoch nicht die Datei RUN.BIN gefunden werden.
targetWrongMode	210	Die aufgerufene Prozedur oder Funktion erfordert einen bestimmten Mode des Targets. Z. B. setzt die Prozedur `setProgrammingSpeed` den ProgramMode voraus.
targetCommunication	211	Ein Kommunikationsfehler mit dem Target.
targetMemoryLayout	212	Das Speicherlayout des Controllers ist nicht ermittelt worden. Bei bekannten Controllern geschieht dieses automatisch beim Aufruf von `getSignature()`. Ansonsten kann das Layout mittels `setFlashLayout()` und `setEepromLayout()` manuell gesetzt werden.
targetExtendedAddress	213	Es ist nicht ermittelt worden, ob der Controller einen ExtendedAddressMode hat. Bei bekannten

		Controllern geschieht dieses automatisch beim Aufruf von `getSignature()`. Ansonsten kann das mittels `setExtendedAddressMode()` manuell gesetzt werden.
targetVerify	214	Beim Zurücklesen von Daten wurde ein Unterschied festgestellt.
hexFileSize	230	Die Größe der angegebenen Hex-Datei ist nicht plausibel. Eventuell ist die Hex-Datei nicht in Ordnung.
hexFileCRC	231	Beim Parsen der Hex-Datei ist ein Prüfsummenfehler aufgetreten. Eventuell ist die Hex-Datei nicht in Ordnung.

4 Vom Benutzter ausgelöste Exceptions

- Der Benutzer kann mittels `throw` selbst Exceptions auslösen. Diese können numerisch sein und bestehende Werte mitnutzen, z. B.:
```
throw rangeError
```
- Um vom Benutzer erzeugte Exceptions von den anderen Exceptions besser unterscheiden zu können, können andere Exceptionnummern verwendet werden. Es wird hierzu die Konstante `userException` mit dem Wert 1000 zur Verfügung gestellt. Der Vorteil dieses Wertes ist, daß wenn die Exception nicht mehr gefangen wird, sie im Blinkcode besonders gut zu erkennen ist. Die Konstante ist als Offset für eigenen Exceptions nutzbar:
```
throw userException + 1
```
- Es können auch nicht-numerische Exceptions erzeugt werden. Falls eine solche Exception nicht mehr gefangen wird, dann wird sie zum Schluß in die Exception `exceptionIsNotANumber` umgewandelt und der Code rausgeblinkt, z. B.:
```
throw "error"
```

x Bedeutungen von LED-Codes

1 Normaler Betrieb

1.1 Keine microSD-Karte gefunden

LEDs:

1: rot

2:

3:

4:

5:

Bedeutung:

Keine microSD-Karte gefunden, bzw. ist nicht mit FAT32 formatiert.

1.2 Exception aufgetreten

Wenn eine Exception aufgetreten ist und diese nicht aufgelöst (gefangen) wurde, dann wird die Nummer der Exception durch einen Blinkcode ausgegeben.

LEDs:

1: rot: geht am Anfang des Blinkcodes kurz aus und wieder an

2: rot: blinkend, Anzahl entspricht 1000-er der Exception

3: rot: blinkend, Anzahl entspricht 100-er der Exception

4: rot: blinkend, Anzahl entspricht 10-er der Exception

5: rot: blinkend, Anzahl entspricht 1-er der Exception

Bedeutung:

Zählen Sie, wie oft die einzelnen LEDs aufblinken, und Sie erhalten den Code der Exception.

Dieser Code kann entstanden sein, indem

- im Skript eine entsprechende „`throw`“-Anweisung ausgeführt wurde. Beispiel:
```
if getSignature() <> sig_atmega32 !atMega32?
  throw 1234 !Exception 1234 erzeugen
endif
```
- eine Funktion / Prozedur konnte Ihre Aufgabe nicht erfüllen und hat eine Exception erzeugt.

2 roloFlash-Aktualisierung

Die Aktualisierung der roloFlash-Firmware ist im Kapitel V “Aktualisieren von roloFlash“ beschrieben.

2.1 Aktualisierung läuft

LEDs:

1: rot

2: grün \ im Wechsel schnell blinkend

3: grün /

4:

5:

Bedeutung:

Datei RUN.BIN nicht gefunden. Bitte führen Sie die Schritte unter IV.1 „Vorbereitung der microSD-Karte am PC (z. B. in der Entwicklung)“ durch.

2.2 Aktualisierung mit Erfolg abgeschlossen

LEDs:

1: grün

2: grün

3:

4:

5:

Bedeutung:

Die Aktualisierung wurde mit Erfolg abgeschlossen. Nach dem Abziehen steht die neue Firmware bei der nächsten Nutzung zur Verfügung.

2.3 Aktualisierung fehlerhaft

LEDs:

1: rot

2:

3: rot

4:

5:

Bedeutung:

Die Aktualisierung schlug fehl. Die alte Firmware steht evtl. noch zur Verfügung.

Mögliche Abhilfe:

- Aktualisierung nochmals versuchen.
- Aktualisierung mit einer anderen Firmware durchführen.

XI SPEZIFIKATION

Technische Daten

- Unterstützt Controller der Atmel-AVR-Serie mit ISP-Interface:
  - AT89
  - AT90
  - ATtiny (nicht ATtiny4/5/9/10)
  - ATmega
- Programmierung des Mikrocontrollers über 6poligen ISP-Stecker
- Spannungsversorgung über den zu programmierenden Mikrocontroller (2,0 - 5,5 Volt)
- Programmierung von:
  - Flash
  - EEPROM
  - Fusebits (LO, HI, EXT)
  - Lockbits
- Unterstütztes Dateisystem: FAT32
- Unterstütztes Dateiformat: Intel HEX (".HEX")
- Unterstützte Speicherkarten-Formate: microSD, microSDHC