LCU1 · Linux Control Unit

Oberflächen-Engine

Inhalt

1 Einführung

Dieses Dokument beschreibt die Oberflächen-Engine (UI-Engine) der LCU1.
Mit Hilfe der UI-Engine läßt sich eine grafische Benutzeroberfläche für das TFT-Display der LCU1 erstellen.
Für den Zugriff auf das TFT-Display wird die Bibliothek DirectFB (http://www.directfb.org) verwendet.
Die Oberfläche ist in einzelne Seiten unterteilt, vergleichbar mit Webseiten. Jede Seite wird mittels einer XML-Datei beschrieben. Eine solche XML-Datei besteht aus einem deklarativen Teil, der den grafischen Aufbau der Seite aus einfachen Elementen beschreibt, sowie aus einem dynamischen Teil, der ausführbaren Code für die Benutzerinteraktion enthält. Der ausführbare Code kann auf Benutzereingaben und Ereignisse von aussen reagieren. Als Reaktion auf diese Ereignisse kann der Code die grafischen Elemente der Benutzeroberfläche manipulieren.
Der ausführbare Code wird in TCL (http://www.tcl.tk) geschrieben. Für die programmatische Interaktion mit der Oberfläche wurde TCL um eigene Befehle für die UI-Engine erweitert.

2 Start der UI-Engine

Die UI-Engine ist in der Firmware der LCU1 enthalten. Die ausführbare Datei liegt unter `/bin/uiengine`. Die Seitenbeschreibung der Demoapplikation befindet sich im Verzeichnis `/usr/share/ui`.

Für den Start der Demoapplikation reicht der Befehl

```
# uiengine
```

Eine Übersicht über die Optionen erhält man mit

```
# uiengine --help
```

2.1 Optionen für Logausgaben

Abgesehen von den Logmeldungen der Bibliothek DirectFB werden Logmeldungen der UI-Engine per default über Syslog geloggt. Sie erscheinen daher in der Datei /var/log/messages

Über die Option `-l` läßt sich das Loglevel einstellen, über die Option `-c` läßt sich alternativ auf die Textkonsole loggen. Mit folgendem Aufruf werden alle Logmeldungen auf der Konsole angezeigt (Level=DEBUG):

```
# uiengine -c -l 1
```

2.2 Verzeichnis für XML-Dateien setzen

Die UI-Engine liest per default alle XML-Dateien im Verzeichnis `/usr/share/ui` ein. Ein anderes Verzeichnis läßt sich mit dem Kommandozeilenparameter `-d` angeben.
Zum Testen einer eigenen Oberfläche bietet es sich an, diese per scp oder WinSCP in die Ramdisk der LCU1 zu übertragen, z.B. nach `/var/ui`. Von dort kann sie dann per
`# uiengine -d /var/ui`
gestartet werden. Die endgültige Version einer eigenen Oberfläche sollte man dann in der schreibbaren NAND-Partition ablegen, z.B. unter `/usr/local/ui`.

2.3 Startseite setzen

Von allen Seiten in den Seitenbeschreibungsdateien muss genau eine mit dem XML-Attribut `start="true"` als Startseite markiert sein. Diese wird dann als erste Seite beim Starten der UI-Engine geladen. Zu Testzwecken kann es hilfreich sein, eine andere Seite als Startseite zu laden. Dafür gibt es die Option `-s`. So zeigt Aufruf

```
# uiengine -s network
```
direkt die Seite mit den Netzwerkeinstellungen an.

2.4 UI-Engine automatisch starten

Beim Starten der LCU1 wird im Auslieferungszustand automatisch die Applikation `demoapp` gestartet. Dies geschieht durch das Startskript `/etc/init.d/S99application`. Dieses Startskript liest die Datei `/usr/local/etc/autostart` ein und startet die dort angegebene Applikation mit den dort angegebenen Parametern.
In diese Datei kann man statt der Applikation `demoapp` auch die UI-Engine mit den gewünschten Optionen eintragen. Der Inhalt der Datei `/usr/local/etc/autostart` könnte dann z.B. so aussehen:

```
AUTOSTART_APP=uiengine
AUTOSTART_ARGS="-d /usr/local/ui"
```

3 Beschreibung der grafischen Seiten als XML

Die grafischen Seiten der UI-Engine bestehen hauptsächlich aus nur zwei einfachen Elementen, Grafiken und Text. Die Grafiken liegen als PNG mit Transparenz vor. Dadurch lassen sich diese nahtlos übereinanderstapeln. Für die Texte können beliebige TrueType-Schriftarten verwendet werden.
Aus diesen beiden einfachen Elementen lassen sich unterstützt durch den TCL-Code auch komplexe Oberflächenelemente realisieren.
Eine kommentierte formale Definition des für die Seitenbeschreibung verwendeten XML-Formates befindet sich im XML-Schema uidescription.xsd.
Dieses Kapitel vermittelt anhand eines einfachen Beispiels einen schnellen Überblick über den Aufbau der Seitenbeschreibung. Im Anschluss wird das Format der Seitenbeschreibungsdateien detailliert beschrieben.

3.1 Schnelleinstieg anhand eines Beispiels

```
<ui>
  <page id="hello" font="fonts/decker.ttf" font_height="30"
        onload="OnPageLoad">

    <elements>
      <image id="background" x="0" y="0"
             file="images/background.png" onclick="exit"/>
      <text id="text_1" x="160" y="50" height="30" center="true"/>
    </elements>

    <code><![CDATA[
        proc OnPageLoad { previousPage reason } {
            ui set text_1.text "Hello world!!!"
        }
]]></code>
  </page>
</ui>
```

Das obige Beispiel zeigt den grundlegenden Aufbau der Seitenbeschreibung. Im Folgenden wird ein kurzer Überblick über die beschriebene Seite gegeben. Eine tiefere Beschreibung folgt weiter unten.
Das Beispiel ist in der Firmware der LCU enthalten und kann durch Eingabe von
```
# uiengine -s hello
```
ausgeführt werden.

- `<ui>`
  Das Root-Element ist immer `<ui>`
  - `<page>`
    Unterhalb des Root-Elementes folgen beliebig viele `<page>`-Elemente. Jedes `<page>`-Element beschreibt eine Oberflächenseite. Aus Gründen der Übersichtlichkeit wird empfohlen, abgesehen von Ausnahmefällen nur eine Oberflächenseite pro XML-Datei zu beschreiben.
    - `id="hello"`
      Das `<page>`-Element hat ein Attribut `id`, welches die in Verlinkungen verwendete Id der Seite angibt. Diese Seite hat also die Id `"hello"`.
    - `font="fonts/decker.ttf"`
      Hiermit wird die default-Schriftart für die Oberflächenseite gesetzt. Unabhängig davon kann jedes UI-Element seine eigene Schriftart verwenden.
    - `font_height="30"`
      Setzt die default-Schriftgröfe für die Oberflächenseite. Unabhängig davon kann jedes UI-Element seine eigene Schriftgröße verwenden.
    - `onload="OnPageLoad"`
      Gibt an, dass bei jedem Anzeigen der Seite die TCL-Methode OnPageLoad aufgerufen wird.
    - `<elements>`
      Root-Element für die Auflistung der grafischen Elemente.
      - `<image>`
        Stellt eine PNG-Grafik dar. In diesem Fall eine Display-füllende Hintergrundgrafik.
        - `id="background"`
          Definiert eine Id für die Grafik. Die Angabe ist optional. Über die Id kann aus den TCL-Code auf das Grafikelement zugegriffen werden, um es z.B. unsichtbar zu schalten oder zu verschieben.
        - `x="0" y="0"`
          Gibt die Position der Grafik an. In diesem Fall liegt die obere linke Ecke der Grafik in der oberen linken Ecke des Displays.
        - `file="images/background.png"`
          Gibt den Dateinamen der Grafikdatei an. In diesem Fall handelt es sich um einen Pfad relativ zum Verzeichnis mit den XML-Dateien.
        - `onclick="exit"`
          Dieses Attribut gibt an, was passieren soll, wenn der Anwender innerhalb des Bereiches der Grafik auf den Touchscreen drückt. In diesem Fall wird dann die UI-Engine beendet. Es ließe sich hier z.B. auch ein Link auf eine andere Seite angeben oder eine TCL-Methode aufrufen.
      - `<text>`
        Ist ein Textelement, für das programmatisch der anzuzeigende Text gesetzt werden soll. Statischer Text ließe sich auch als Attribut `text` direkt angeben.
        - `id="text_1"`
          Definiert die Id für den programmatischen Zugriff auf das Textelement.
        - `x="160" y="50"`
          Gibt die Position des Textes an. Da auch center="true" gesetzt ist (siehe unten), bezieht sich die Position auf die horizontale Mitte der oberen Kante des Textes.
        - `height="30"`

Gibt die Höhe des Elementes in Pixeln an. Für Textelemente wird dadurch gleichzeitig die Schriftgröße angegeben.

- `center="true"`
  Gibt an, dass der Text zentriert angezeigt werden soll. Gleichzeitig wird dadurch der Referenzpunkt für die Positionierung in die Mitte der oberen Kante verschoben.

- `<code>`
  Innerhalb des `<code>`-Elements wird der TCL-Programmtext für das dynamische Verhalten der Seite angegeben. Um den Programmtext lesbar zu halten und keine XML-Zeichen durch Escape-Sequenzen darstellen zu müssen, empfiehlt es sich, diesen Bereich wie im Beispiel explizit als CDATA zu markieren.
  - `proc OnPageLoad`
    Ist die TCL-Methode, die oben als Handler für `onload`-Ereignis der Seite angegeben ist. Der Handler bekommt beim Aufruf zwei Parameter übergeben, die Id der vorherigen Seite und den Grund, warum die Seite angezeigt wird. Als Grund wird entweder `"SHOW"` oder `"BACK"` übergeben. Durch Prüfen der beiden Parameter kann der TCL-Code insbesondere bei Seiten mit untergeordneten Seiten korrekt reagieren.

3.2 Details der Seitenbeschreibung

3.2.1 Pfadangaben

Es werden innerhalb der XML-Datei an einigen Stellen Dateisystempfade angegeben. Das ist z.B. der Fall, wenn eine Grafikdatei oder eine TrueType-Datei referenziert werden.
Alle Pfadangaben können relativ oder absolut erfolgen. Relative Pfadangaben beziehen sich auf das Verzeichnis, in dem die XML-Dateien liegen, per default also auf `/usr/share/ui`.

3.2.2 Root-Element

Das Rootelement der Seitenbeschreibungsdatei ist immer `<ui>`. Unterhalb des Root-Elementes darf es beliebig viele Elemente `<page>` geben. Aus Gründen der Übersichtlichkeit wird empfohlen, abgesehen von Ausnahmefällen nur eine Oberflächenseite pro XML-Datei zu beschreiben.

3.2.3 Element `<page>`

Jedes <page>-Element beschreibt eine einzelne Seite der Benutzeroberfläche.

3.2.3.1 Attribute des `<page>`-Elements

- `id [String, Pflicht]`
  Dieses Attribut ist verpflichtend. Es weist der Seite eine Id zu, die bei Verweisen und im TCL-Code verwendet werden kann, um sich auf die Seite zu beziehen.
- `onload [String, optional]`
  Dieses Attribut ist optional. Es gibt den Namen einer TCL-Methode an, die jedes mal aufgerufen wird, wenn die Seite angezeigt wird. Für nähere Informationen zur Implementierung der TCL-Methode siehe 4.1.3.1 .
- `start [Boolean, optional]`
  Dieses Attribut ist optional. Wenn `start="true"` angegeben ist, wird dadurch die entsprechende Seite als Startseite markiert. Es muss genau eine Seite als Startseite markiert sein, ansonsten verweigert die UI-Engine die Ausführung.
- `waitcursor [String, optional]`
  Dieses Attribut ist optional. Es wird nur für die Startseite (also bei `start="true"`) ausgewertet. Es gibt den Pfad zur PNG-Datei mit dem für die komplette Applikation

verwendeten Cursor für länger dauernde Operationen ("Eieruhr") an.

- `font [String, optional]`
  Dieses Attribut ist optional. Es gibt den Pfad zur für diese Seite als Standardschriftart zu verwendende TrueType-Datei an. Untergeordnete Elemente können die Standardschriftart mit einer eigenen Schriftart überschreiben.
- `font_height [Integer, optional]`
  Dieses Attribut ist optional. Es gibt die Standardschriftgröße für die Seite in Pixeln an. Untergeordnete Elemente können eine eigene Schriftgröße angeben.
- `onAudioStart [String, optional]`
  Dieses Attribut ist optional. Es gibt den Namen einer TCL-Methode an, die als Eventhandler aufgerufen wird, wenn die Ausgabe einer Audiodatei startet. Für nähere Informationen zur Implementierung der TCL-Methode siehe Fehler: Referenz nicht gefunden.
- `onAudioStop [String, optional]`
  Dieses Attribut ist optional. Es gibt den Namen einer TCL-Methode an, die als Eventhandler aufgerufen wird, wenn die Audioausgabe einer Audiodatei beendet ist. Für nähere Informationen zur Implementierung der TCL-Methode siehe 4.3.2.2 .
- `onAudioProgress [String, optional]`
  Dieses Attribut ist optional. Es gibt den Namen einer TCL-Methode an, die als Eventhandler jede Sekunde aufgerufen wird, während eine Audiodatei wiedergegeben wird. Für nähere Informationen zur Implementierung der TCL-Methode siehe 4.3.2.3 .

3.2.3.2 Untergeordnete Elemente des <page>-Elements

- `elements`
  Dieses Element enthält eine Liste der Oberflächenelemente, aus denen die Oberflächenseite besteht. Siehe 3.2.4 .
- `triggers`
  Dieses Element enthält eine Liste von Triggerdefinitionen, um auf Änderungen an immediateC-Werten reagieren zu können. Siehe 3.2.9 .
- `code`
  Dieses Element enthält den TCL-Code, der das dynamische Verhalten der Oberflächenseite definiert. Siehe 3.2.11 .

3.2.4 Element <elements>

Dieses Element enthält eine Liste der Oberflächenelemente, aus denen die Oberflächenseite besteht.

3.2.4.1 Untergeordnete Elemente des <elements>-Elements

- `<image>`
  Zeigt eine PNG-Grafik an. Siehe 3.2.6 .
- `<text>`
  Zeigt einen Text an. Siehe 3.2.7 .
- `<datetime>`
  Zeigt das aktuelle Datum bzw. die aktuelle Zeit an. Siehe 3.2.8 .

3.2.5 Gemeinsame Attribute der Oberflächenelemente

Die Oberflächenelemente unterstützen eine Menge gemeinsamer Attribute. Diese werden im folgenden beschrieben.

- `id [String, optional]`
  Weist dem Oberflächenelement eine Id zu, über die aus dem TCL-Code heraus auf das Oberflächenelement zugegriffen werden kann.
- `x [Integer, optional, default 0]`

X-Koordinate des Oberflächenelementes in Pixeln. Bezugspunkt ist normalerweise die linke Kante des Elementes. Oberflächenelemente, die Text anzeigen, haben ein Attribut `center` für die zentrierte Textdarstellung. Wenn dieses Attribut gesetzt ist, bezieht sich die X-Koordinate auf die horizontale Mitte des Elementes.

- `y [Integer, optional, default 0]`
  X-Koordinate des Oberflächenelementes in Pixeln. Bezugspunkt ist die obere Kante des Oberflächenelementes.
- `width [Integer, optional, default 0]`
  Breite des Oberflächenelementes in Pixeln.
- `height [Integer, optional, default 0]`
  Höhe des Oberflächenelementes in Pixeln.
- `visible [Boolean, optional, default false]`
  Gibt an, ob das Oberflächenelement sichtbar ist oder nicht.
- `onclick [String, optional]`
  Definiert eine Aktion für das Klicken auf den Touchscreen innerhalb des durch das Oberflächenelementes aufgespannten Rechtecks. Es wird ein Aktionstyp und ein optionaler Parameter angegeben. Beide werden durch einen Doppelpunkt getrennt. Es sind die folgenden Aktionstypen definiert:
  - `exit`
    Benötigt keinen Parameter. Beendet die Applikation. Beispiel: `onclick="exit"`.
  - `back`
    Benötigt keinen Parameter. Schaltet zurück auf die vorherige Oberflächenseite. Beispiel: `onclick="back"`.
  - `link`
    Zeigt die als Parameter angegebene Oberflächenseite an. Es ist die Id der gewünschten Seite anzugeben. Beispiel: `onclick="link:network"`.
  - `code`
    Ruft die angegebene TCL-Methode auf. Die Methode bekommt die Id des Oberflächenelementes als Parameter übergeben. Nähere Informationen zur Implementierung der TCL-Methode siehe 4.1.3.2 . Beispiel: `onclick="code:OnOk"`.

3.2.6 Element `<image>`

Dieses Element beschreibt eine PNG-Grafik oder eine Liste von PNG-Grafiken, zwischen denen per Code umgeschaltet werden kann.

3.2.6.1 Attribute des `<image>`-Elements

Es werden alle unter 3.2.5 beschriebenen Attribute unterstützt. Die Breite und die Höhe werden automatisch berechnet, falls sie nicht angegeben wurden.
Zusätzlich werden die folgenden Attribute unterstützt:

- `file [String, optional]`
  Dieses Attribut gibt den Pfad zur PNG-Datei an, die angezeigt werden soll. Wird es nicht angegeben, so muss mindestens ein untergeordnetes Element `<file>` angegeben werden.

3.2.6.2 Untergeordnete Elemente des `<image>`-Elements

- `<file>`
  Es können beliebig viele `<file>`-Elemente angegeben werden. Jedes `<file>`-Element definiert den Pfad einer anzuzeigende PNG-Datei. Initial wird die erste definierte Datei angezeigt. Aus dem TCL-Code heraus kann mit der Eigenschaft index zwischen den definierten Dateien umgeschaltet werden. Dadurch lässt sich z.B. eine Checkbox mit zwei

verschiedenen Zuständen durch zwei Grafiken darstellen. Der Dateipfad wird als Text des `<file>`-Elementes angegeben, z.B. `<file>images/checkbox_off.png</file>`.

3.2.7 Element `<text>`

Dieses Element beschreibt ein Oberflächenelement zur Textanzeige.

3.2.7.1 Attribute des `<text>`-Elements

Es werden alle unter 3.2.5 beschriebenen Attribute unterstützt. Das dort beschriebene Attribut `height` wird ausgewertet, um die Schriftgröße festzulegen.
Zusätzlich werden die folgenden Attribute unterstützt:

- `text [String, optional]`
  Dieses Attribut gibt den Text an, der angezeigt werden soll.
- `font [Boolean, optional]`
  Dieses Attribut gibt den Pfad zur zu verwendenden TrueType-Schriftart an. Wenn es fehlt, wird die Standardschriftart der Oberflächenseite verwendet.
- `color [Integer, optional, default 0x000000]`
  Dieses Attribut gibt die Textfarbe an. Diese ist in 24bit RGB-Codierung als Dezimalzahl oder mit vorangestelltem `"0x"` als Hexadezimalzahl anzugeben. Orange entspricht damit z.B. `color="0xffff00"`. Blau entspricht `color="0x0000ff"`.
- `center [Boolean, optional]`
  Dieses Attribut gibt an, ob der Text horizontal zentriert werden soll. In der Standardeinstellung wird der Text linksbündig dargestellt. Bezugspunkt ist die als Attribut `x` angegebene X-Koordinate.

3.2.8 Element `<datetime>`

Dieses Element zeigt das aktuelle Datum und / oder die aktuelle Zeit als Text an. Die Anzeige wird automatisch aktualisiert. Die Aktualisierung erfolgt abhängig von der Genauigkeit der Formatangabe (siehe unten) sekündlich oder minütlich.

3.2.8.1 Attribute des `<datetime>`-Elements

Es werden ausser dem Attribut text alle Attribute des <text>-Elementes unterstützt. Siehe 3.2.7.1 .
Zusätzlich werden die folgenden Attribute unterstützt:

- `format [String, optional, default "%x %X"]`
  Dieses Attribut gibt den Formatstring an, mit dem Zeit und Datum formatiert werden. Es wird das Format der C-Funktion `strftime()` verwendet.

3.2.9 Element `<triggers>`

Dieses Element enthält eine Liste von Triggerdefinitionen. Eine Triggerdefinition ordnet einem Signal aus der Logiksteuerung (icserver) eine TCL-Methode zu, die aufgerufen wird, wenn sich der Wert des Signals geändert hat.

3.2.9.1 Untergeordnete Elemente des `<triggers>`-Elements

- `<trigger>`
  Enthält eine Triggerdefinition. Siehe 3.2.10 .

3.2.10 Element `<trigger>`

Dieses Element enthält eine Triggerdefinition. Eine Triggerdefinition ordnet einem Signal aus der Logiksteuerung (icserver) eine TCL-Methode zu, die aufgerufen wird, wenn sich der Wert des Signals geändert hat.

3.2.10.1 Attribute des <trigger>-Elements

Das Element `<trigger>` hat die folgenden Attribute:

- `value [String]`
  Die Id oder der Alias des Logiksignals, das überwacht werden soll.
- `method [String]`
  `Dieses Attribut` gibt den Namen einer TCL-Methode an, die jedes mal aufgerufen wird, wenn der Wert des Logiksignals sich geändert hat. Für nähere Informationen zur Implementierung der TCL-Methode siehe 4.4.2.1 .
- `global [Boolean, optional, default=false]`
  Wenn dieses Attribut auf true gesetzt ist, wird die Triggermethode auch aufgerufen, wenn die Seite, die den Trigger definiert gerade nicht angezeigt wird. Im Normalfall wird die Triggermethode nur aufgerufen, wenn die definierende Seite die aktuell dargestellte Seite ist.

3.2.11 Element <code>

Dieses Element definiert in Form von TCL-Code das dynamische Verhalten der Oberflächenseite. Der TCL-Code wird als Text des `<code>`-Elementes angegeben.
Um den TCL-Programmtext lesbar zu halten und keine XML-Zeichen durch Escape-Sequenzen darstellen zu müssen, empfiehlt es sich, diesen Bereich wie im Beispiel in Kapitel 3.1 explizit als CDATA zu markieren.

4 Implementierung des dynamischen Verhaltens in TCL

Dynamisches Verhalten der Oberflächenseiten läßt sich durch Einbetten von TCL-Code in die XML-Dateien zur Seitenbeschreibung beschreiben. TCL ist eine einfache, leicht zu erweiternde Skriptsprache, die sich leicht in eigene Applikationen einbinden läßt. Nähere Informationen und Dokumentation zu TCL findet man unter http://www.tcl.tk.
Die Firmware der LCU1 enthält TCL in der Version 8.2. Die Dokumentation zu dieser Version befindet sich online unter http://www.tcl.tk/man/tcl8.2. TCL wird von der UI-Engine als Bibliothek eingebunden. Es gibt eine globale Instanz des TCL-Interpreters. Der Code für jede Seite lebt innerhalb des TCL-Interpreters in seinem eigenen Namespace. Der Name des Namespace ist die Id der jeweiligen Seite. Damit hat der Code für jede Oberflächenseite seinen eigenen Namensraum für Methoden und globale Variablen. Durch Angabe des entsprechenden Namespaces kann auf Code und Variablen der anderen Seiten zugegriffen werden.
Unabhängig davon besteht aus allen Interpreter-Instanzen heraus Zugriff auf die Oberflächenelemente aller Oberflächenseiten.
Der TCL-Code aller `<code>`-Elemente in allen Seitenbeschreibungen wird unmittelbar beim Starten der UI-Engine innerhalb des entsprechenden Namensraumes im TCL-Interpreter ausgeführt. Damit stehen alle nicht-lokalen Variablen und Methoden von Anfang an zur Verfügung.
Die UI-Engine definiert zusätzliche TCL-Befehle für die vier Teilbereiche Oberflächensteuerung, Displayhinterleuchtung, Audioausgabe und (immediateC)-Logiksteuerung. Diese werden im folgenden beschrieben.

4.1 TCL-Erweiterung zur Oberflächensteuerung

Es wurde der Befehl `ui` mit mehreren Unterbefehlen hinzugefügt. Zusätzlich wurden zwei Ereignisse definiert, die zum Aufruf von TCL-Methoden führen.

4.1.1 Eigenschaften der Oberflächenelemente

In der XML-Beschreibung wird für jede Oberflächenseite eine Id festgelegt. Für jedes Oberflächenelement kann optional eine Id festgelegt werden. Diese Ids werden zum Zugriff auf Oberflächenseiten und Oberflächenelemente verwendet.

Jedes Oberflächenelement hat eine von Elementtyp abhängige Menge von Eigenschaften. Jede Eigenschaft hat eine feste Id. Der Zugriff auf die Eigenschaften erfolgt hierarchisch indem die Ids der Seite, des Elementes und der Eigenschaft durch Punkte getrennt aneinandergehängt werden. Die komplette Id der Text-Eigenschaft des Textelementes im Beispiel aus Kapitel 3.1 wäre also `hello.text_1.text`. Dabei ist `hello` die Id der Seite, `text_1` die Id des Textelementes und `text` die Id der Text-Eigenschaft.
Mit dieser kompletten Id kann aus jeder beliebigen Oberflächenseite auf den Text zugegriffen werden. Code in der aktuellen Oberflächenseite darf die Id der Seite weglassen. Die abgekürzte Id wäre dann also `text_1.text`.
Wird eine Eigenschaft durch den entsprechenden TCL-Befehl geändert, so wird diese Änderung sofort an der Oberfläche sichtbar.

4.1.1.1 Gemeinsame Eigenschaften aller Oberflächenelemente

- `x [Integer]`
  X-Koordinate des Oberflächenelementes in Pixeln. Bezugspunkt ist normalerweise die linke Kante des Elementes. Oberflächenelemente, die Text anzeigen, haben ein Attribut `center` für die zentrierte Textdarstellung. Wenn dieses Attribut gesetzt ist, bezieht sich die X-Koordinate auf die horizontale Mitte des Elementes.
- `y [Integer]`
  X-Koordinate des Oberflächenelementes in Pixeln. Bezugspunkt ist die obere Kante des Oberflächenelementes.
- `width [Integer]`
  Breite des Oberflächenelementes in Pixeln.
- `height [Integer]`
  Höhe des Oberflächenelementes in Pixeln.
- `visible [Boolean]`
  Gibt an, ob das Oberflächenelement sichtbar ist oder nicht.

4.1.1.2 Eigenschaften von Grafikelementen

- `index [Integer]`
  Wählt bei Grafikelementen mit mehreren hinterlegten Grafikdateien diejenige aus, die angezeigt wird.

4.1.1.3 Eigenschaften von Textelementen

- `text [String]`
  Der Text, der angezeigt wird
- `color [Integer]`
  Diese Eigenschaft gibt die Textfarbe an. Diese ist in 24bit RGB-Codierung als Dezimalzahl oder mit vorangestelltem `"0x"` als Hexadezimalzahl anzugeben. Beim Abfragen der Eigenschaft wird immer die Hexdarstellung zurückgegeben.

4.1.2 Befehle

4.1.2.1 Setzen von Eigenschaften (Befehl `ui set`)

Syntax:
`ui set [<pageId>.]<controlId>.<propertyId> <value>`

Dieser Befehl setzt die Eigenschaft `<propertyId>` des Oberflächenelementes `<controlId>` auf der Seite `<pageId>` auf den Wert `<value>`.
Die Angabe `<pageId>` ist optional. Wenn diese fehlt, wird die Seite angenommen, die zum

aktuellen TCL-Interpreter gehört.

4.1.2.2 Abfragen von Eigenschaften (Befehl `ui get`)

Syntax:
`ui get [<pageId>.]<controlId>.<propertyId>`

Dieser Befehl liefert den Wert der Eigenschaft `<propertyId>` des Oberflächenelementes `<controlId>` auf der Seite `<pageId>` zurück.
Die Angabe `<pageId>` ist optional. Wenn diese fehlt, wird die Seite angenommen, die zum aktuellen TCL-Interpreter gehört.

4.1.2.3 Anzeigen einer anderen Seite (Befehl `ui load`)

Syntax:
`ui load <pageId>`

Dieser Befehl springt zur Seite `<pageId>`. Die aktuelle Seite wird in der Seitenhistorie gespeichert und kann mit dem Befehl `ui back` wieder aufgerufen werden.

4.1.2.4 Zurückspringen zur vorherigen Seite (Befehl `ui back`)

Syntax:
`ui back`

Dieser Befehl springt zurück zur in der Seitenhistorie gespeicherten letzten Seite. Diese wird aus der Historie entfernt.

4.1.2.5 Abfragen des Basisverzeichnisses(Befehl `ui basedir`)

Syntax:
`ui basedir`

Dieser Befehl liefert den Pfad zu dem Verzeichnis zurück, das die Seitenbeschreibungsdateien enthält. Wird die UI-Engine ohne Parameter `-d` aufgerufen, liefert dieser Befehl also immer `"/usr/share/ui"` zurück.

4.1.3 UI-Ereignisse

Die UI-Engine erzeugt zwei Ereignisse, die im Zusammenhang mit der Oberfläche stehen. Wenn die aktuelle Seite wechselt, wird das Ereignis `load` im Kontext der neuen Seite erzeugt. Wenn der Anwender ein Oberflächenelement berührt, wird das Ereignis `click` erzeugt.

4.1.3.1 UI-Ereignis `"load"`

Der Ereignishandler für dieses Ereignis wird per XML mit dem Attribut `onload` des `<page>`-Elementes definiert. Siehe 3.2.3.1 . Es wird eine TCL-Methode mit dem dort angegebenen Namen aufgerufen. Diese muss wie folgt definiert sein:

```
proc OnPageLoad { previousPage reason } {
}
```

Diese Methode wird aufgerufen, unmittelbar bevor die neue Seite angezeigt wird. In dieser Methode durchgeführte Änderungen an Eigenschaften werden daher sofort und ohne Flackern angezeigt.
Der Parameter `previousPage` gibt die Id der vorherigen Seite an. Falls die neu angezeigte Seite die erste Seite ist, ist `previousPage` ein leerer String.

Der Parameter `reason` gibt den Grund an, warum die Seite angezeigt wird. Er kann die beiden Werte „`SHOW`“ und „`BACK`“ annehmen. Wird die Seite über die Seitenhistorie als vorherige Seite aufgerufen, ist der Wert des Parameters „`BACK`“. In allen anderen Fällen ist er „`SHOW`“.

4.1.3.2 UI-Ereignis `"click"`

Der Ereignishandler für dieses Ereignis wird per XML mit dem Attribut `onclick` eines Oberflächenelementes definiert. Siehe 3.2.5 . Falls als Aktion `code` angegeben ist, wird die TCL-Methode mit dem dort als Parameter angegebenen Namen aufgerufen. Diese muss wie folgt definiert sein:

```
proc OnClick { control } {
}
```

Der Parameter `control` wird auf die Id des entsprechenden Oberflächenelementes gesetzt. Dadurch können mehrere gleichartige Oberflächenelemente oft den gleichen Ereignishandler verwenden.

4.2 TCL-Erweiterung zur Hinterleuchtung

Die Displayhinterleuchtung unterstützt eine maximale Helligkeit, eine minimale Helligkeit und eine Verzögerungszeit für das automatische herunterdimmen.
Nach dem Start wird zunächst die maximale Helligkeit angezeigt. Nach Ablauf der Verzögerungszeit wird automatisch auf die minimale Helligkeit heruntergedimmt. Jede Berührung des Touchscreens startet die Verzögerung neu. Ist eine Verzögerung von 0 eingestellt, wird nicht heruntergedimmt.
Für die Kontrolle der Hinterleuchtungseinstellungen wurde der TCL-Befehl `backlight` mit mehreren Unterbefehlen integriert.

4.2.1 Befehle

4.2.1.1 Minimale und maximale Helligkeit setzen und abfragen (Befehl `backlicht range`)

Syntax:
`backlight range [<min> <max>]`

Dieser Befehl setzt die minimale Helligkeit auf `<min>` und die maximale Helligkeit auf `<max>`. Der Wertebereich für beide Werte ist `0-255`.
Der Befehl gibt die aktuellen Werte als Liste mit zwei Elementen zurück. Werden `<min>` und `<max>` nicht angegeben, so wird nur die aktuelle Einstellung zurückgeliefert.

4.2.1.2 Verzögerungszeit setzen und Abfragen (Befehl `backlight time`)

Syntax:
`backlight time [<seconds>]`

Dieser Befehl setzt die Verzögerungszeit für das automatische Herunterdimmen auf `<seconds>` Sekunden. Ein Wert von 0 deaktiviert das Automatische herunterdimmen.
Der Befehl gibt die aktuelle Verzögerungszeit zurück. Wird `<seconds>` nicht angegeben, so wird nur die aktuelle Verzögerungszeit zurückgeliefert.

4.3 TCL-Erweiterung zur Audioausgabe

In die UI-Engine ist Funktionalität zum Ansteuern des Audioplayers `madplay` und zur

Ansteuerung des Audiomixers integriert. Dafür wurde der TCL-Befehl `audio` mit mehreren Unterbefehlen integriert.
Um den Status der Audioausgabe an der grafischen Oberfläche anzeigen zu können, wurden drei Ereignisse definiert.

4.3.1 Befehle

4.3.1.1 Lautstärke setzen und abfragen (Befehl `audio volume`)

Syntax:
```
audio volume <channel> [<percent>|up|down]
```

Dieser Befehl setzt die Lautstärke für den Mixerkanal `<channel>` auf den prozentualen Wert `<percent>`. Bei Angabe von `up` oder `down` als Wert wird die Lautstärke um eine Stufe inkrementiert bzw. dekrementiert. Es wird die neue Lautstärke in Prozent zurückgeliefert. Wird kein Wert angegeben, so wird die aktuelle Lautstärke in Prozent zurückgeliefert.
Die Namen der Mixerkanäle sind von der Hardware abhängig. Sie lassen sich am einfachsten über den Linuxbefehl `amixer` ermitteln. Für den Fall der LCU mit dem Audiomodul IAM100 sind die Kanäle in der Beschreibung des IAM100 dokumentiert. Für die Wiedergabe von MP3-Dateien sind die beiden Kanäle `PCM` und `Master` relevant.

4.3.1.2 Mixerkanal schalten (Befehl `audio switch`)

Syntax:
```
audio switch <channel> [<status>]
```

Dieser Befehl schaltet den Mixerkanal `<channel>` abhängig vom boolschen Parameter `<status>` ein oder aus. Der neue Status wird zurückgegeben. Wird `<status>` nicht angegeben, so wird der aktuelle Status zurückgeliefert.
Für die Namen der Mixerkanäle siehe 4.3.1.1 .

4.3.1.3 Wiedergabe starten (Befehl `audio play`)

Syntax:
```
audio play [-add] [-skip <millis>] [-duration <millis>] [-repeat]
  <mp3_file>
```

Dieser Befehl ersetzt die aktuelle Wiedergabeliste durch die Datei <mp3_file>.
Bedeutung der optionalen Parameter:

- `-add`
  Die Datei wird an die Wiedergabeliste angehängt. Im Normalfall wird die Wiedergabeliste durch die neue Datei ersetzt.
- `-skip <millis>`
  Es werden die ersten `<millis>` Millisekunden der MP3-Datei übersprungen.
- `-duration <millis>`
  Die Wiedergabe der MP3-Datei wird nach `<millis>` Millisekunden beendet.
- `-repeat`
  Die Wiedergabeliste wird zyklisch wiederholt. Im Normalfall wird die Wiedergabeliste geleert, wenn die Wiedergabe der letzten Datei beendet wurde.

Nach diesem Befehl wird sofort mit der Abarbeitung der neuen Wiedergabeliste begonnen.

4.3.1.4 Wiedergabe beenden (Befehl `audio stop`)

Syntax:

```
audio stop
```

Dieser Befehl leert die Wiedergabeliste und beendet damit die Wiedergabe.

4.3.1.5 Wiedergabestatus abfragen (Befehl `audio info`)

Syntax:
```
audio info
```

Dieser Befehl liefert ein Array mit Informationen zum Status der Wiedergabe. Das Array wird in Form einer Liste zurückgegeben und muss zunächst mit dem TCL-Befehl `array set` in eine Arrayvariable eingelesen werden.
Das Array enthält die folgenden Felder:

- `FilePath`
  Absoluter Pfad der Audiodatei.
- `TotalTimeSeconds`
  Gesamte Spielzeit der Datei in Sekunden.
- `Title`
  Titel der Datei aus den IDv3 Tags.
- `Artist`
  Interpret der Datei aus den IDv3 Tags.
- `Album`
  Name des Albums aus den IDv3 Tags.

4.3.2 Audio-Ereignisse

Während der Wiedergabe einer Audiodatei werden drei verschiedene Ereignisse erzeugt. Beim Start der Wiedergabe einer neuen Datei wird das Ereignis audioStart erzeugt. Während der Wiedergabe wird in regelmäßigen Abständen das Ereignis audioProgress erzeugt. Wenn die Wiedergabe beendet wurde, wird das Ereignis audioStop erzeugt.

4.3.2.1 Audio-Ereignis "`audioStart`"

Der Ereignishandler für dieses Ereignis wird per XML mit dem Attribut `onAudioStart` des `<page>`-Elementes definiert. Siehe 3.2.3.1 . Es wird eine TCL-Methode mit dem dort angegebenen Namen aufgerufen. Diese muss wie folgt definiert sein:

```
proc OnAudioStart { eventId info } {
}
```

Diese Methode wird aufgerufen, sobald die Wiedergabe einer neuen Audiodatei beginnt.
Der Parameter `eventId` ist immer "`start`".
Der Parameter `info` ist ein Array mit Informationen zur aktuell wiedergegebenen Audiodatei. Das Array wird in Form einer Liste übergegeben und muss zunächst mit dem TCL-Befehl `array set` in eine Arrayvariable eingelesen werden.
Für die Felder des Arrays siehe 4.3.1.5 .

4.3.2.2 Audio-Ereignis "`audioStop`"

Der Ereignishandler für dieses Ereignis wird per XML mit dem Attribut `onAudioStop` des `<page>`-Elementes definiert. Siehe 3.2.3.1 . Es wird eine TCL-Methode mit dem dort angegebenen Namen aufgerufen. Diese muss wie folgt definiert sein:

```
proc OnAudioStop { eventId } {
}
```

Diese Methode wird aufgerufen, sobald die Wiedergabe einer Audiodatei beendet wurde.
Der Parameter `eventId` ist immer "`stop`".

4.3.2.3 Audio-Ereignis "`audioProgress`"

Der Ereignishandler für dieses Ereignis wird per XML mit dem Attribut `onAudioProgress` des `<page>`-Elementes definiert. Siehe 3.2.3.1 . Es wird eine TCL-Methode mit dem dort angegebenen Namen aufgerufen. Diese muss wie folgt definiert sein:

```
proc OnAudioProgress { eventId totalTime remainingTime } {
}
```

Diese Methode wird regelmäßig aufgerufen, solange eine Audiodatei wiedergegeben wird.
Der Parameter `eventId` ist immer "`progress`".
Der Parameter `totalTime` gibt die gesamte Spielzeit der aktuellen Datei in Sekunden an.
Der Parameter `remainingTime` gibt die verbleibende Spielzeit der aktuellen Datei in Sekunden an.

4.4 TCL-Erweiterung zur Logiksteuerung

4.4.1 Befehle

4.4.1.1 Setzen des Wertes eines Logiksignals (Befehl `control set`)

Syntax:
`control set <signalId> <value>`

Dieser Befehl setzt das Logiksignal mit dem Namen oder Alias `<signalId>` auf den Wert `<value>`. Der Wertebereich für den Wert hängt von der Bitbreite des Logiksignals ab.

4.4.1.2 Abfragen des Wertes eines Logiksignals (Befehl `control get`)

Syntax:
`control get <signalId>`

Dieser Befehl gibt den Wert des Logiksignals mit dem Namen oder Alias `<signalId>` als Integer zurück. Der Wertebereich für den Rückgabewert hängt von der Bitbreite des Logiksignals ab.

4.4.1.3 Einrichten eines Triggers für ein Logiksignal (Befehl `control trigger`)

Syntax:
`control trigger <signalId> <eventHandler>`

Dieser Befehl verknüpft einen Eventhandler mit einem Logiksignal. In der Folge wird dann die TCL-Methode `<eventHandler>` jedesmal aufgerufen, wenn sich der Wert des Logiksignals `<signalId>` ändert. Diese Befehl entspricht dem XML-Element `<trigger>` (siehe 3.2.10). Zur Implementierung des Ereignishandlers siehe 4.4.2.1 .

4.4.2 Logik-Ereignisse

4.4.2.1 Logik-Ereignis "trigger"

Der Ereignishandler für dieses Ereignis wird per XML mit dem Attribut `method` des `<trigger>`-Elementes definiert. Siehe 3.2.10 . Es wird eine TCL-Methode mit dem dort

angegebenen Namen aufgerufen. Diese muss wie folgt definiert sein:

```
proc OnLogicTrigger { signalId signalValue } {
}
```

Diese Methode wird wird aufgerufen, wenn sich der Wert des entsprechenden Logiksignals geändert hat.
Der Parameter `signalId` enthält den Namen oder Alias des Logiksignals, wie mit dem Attribut `value` des `<trigger>`-Elementes definiert.
Der Parameter `signalValue` enthält den neuen Wert des Logiksignals.