

## MMI White Paper

Konzept zur Gestaltung von Man Machine Interfaces mittels der  
sevenstax MMI-Lib

Revisions-Nr.: 1.3  
Status: Released  
Autor: Matthias Krafft, sevenstax GmbH

Initial version: 22.09.03  
Last change: 23.12.03  
Changed by: Matthias Krafft  
Publication: Public  
Filename: MMI Whitepaper V1.3

## Inhaltsverzeichnis

1	Inhalt.....	3
2	Der Weg zum Man Machine Interface.....	4
2.1	Style-Guide.....	4
2.2	Struktur einer MMI.....	4
2.3	Basiselemente.....	5
2.4	Dialoge, Menüs und Benachrichtigungen.....	5
2.5	Berechtigungen und Systemabhängigkeiten.....	5
2.6	MMI-Spezifikation.....	6
3	Übersicht der sevenstax MMI-Bibliothek.....	7
3.1	Aufbau der MMI-Library.....	7
3.2	MMI-Integration in ein System.....	8
4	Anforderungen an das System.....	9
4.1	Tastatortreiber.....	9
4.2	Displaytreiber.....	9
4.3	MMI-Systemaufrufe.....	10
5	Features der sevenstax MMI .....	11
6	Übersicht MMI-Items.....	11
6.1	MMI-Item-Hierarchie innerhalb eines Screens .....	13
6.2	Items der sevenstax MMI Bibliothek.....	14
7	XML-Ressourcen.....	15
7.1	Übersicht der verwendeten XML-Syntax.....	15
7.2	Beispiel-Screen-XML-Datei.....	16
7.3	XML-nach-C-Konverter.....	17
8	Änderungsverzeichnis.....	17

## **1 Inhalt**

Dieses Dokument beschreibt die Erstellung von Benutzerinterfaces für embedded Geräte mittels der von sevenstax entwickelten MMI-Bibliothek. Der Fokus liegt auf Geräten mit kleinem Display und einigen wenigen Tasten zur Bedienung. Es können alphanumerische und grafische Displays angesteuert werden.

Zum Inhalt des Dokumentes gehört die Beschreibung der MMI-Bibliothek, deren Funktionsweise und Elemente, aus denen sich eine Benutzerführung aufbauen lässt, die Erstellung von Benutzerinterfaces sowie die Anforderungen an Hardware und Software.

Die sevenstax MMI ist eine portable Software-Bibliothek zur Erstellung von Mensch-Maschine-Interfaces (MMI) auf beliebigen embedded Systemen. Um in einem weiten Feld einsetzbar zu sein, wurde beim Design Wert auf gute Skalierbarkeit und minimalen Ressourcenbedarf gelegt, ohne jedoch die Flexibilität und die Wartbarkeit zu beeinträchtigen.

## 2 Der Weg zum Man Machine Interface

Ein gutes Man Machine Interface für embedded Geräte ist in erster Linie textorientiert. Grafische Elemente und Icons können als gestalterische Elemente eingefügt werden, tragen aber zum Verständnis der Benutzerführung wenig bei.

Zur Bedienung werden meist wenige Tasten bevorzugt. Eine alphanumerische Tastatur ist zum Eingeben von Texten sinnvoll, benötigt aber viel Platz, ist teurer und vermittelt dem Benutzer den Eindruck einer hohen Komplexität. Minimum für eine Bedienung sind drei Tasten: hoch, runter und Auswahl (bzw. links, rechts und ok). Drei Tasten können durch einen Drehknopf ersetzt werden (links, rechts drehen, drücken). Eine Abbrechen-Taste ist immer zu empfehlen. Zusätzliche Funktionstasten ermöglichen ein schnelles Anspringen von oft genutzten Funktionen und reduzieren die Menütiefe.

Die Ausgaben werden in Ausgabebereiche auf dem Display organisiert und/oder in logische Einheiten, sog. Screens, Displayobjects oder Widgets. In fast allen Anwendungsfällen ist die Organisation der MMI in logischen Einheiten der ideale Weg zu einer guten flexiblen Struktur.

Der erste Schritt zur Erstellung der MMI ist die Erstellung eines Style-Guides.

### 2.1 Style-Guide

In einem Style-Guide werden die konstanten, immer wieder verwendeten Elemente und Mechanismen der MMI definiert. Dazu gehören:

- **Konstruktive Bedingungen** – Displayauflösung, Anzahl der Tasten und deren Beschriftung
- **Displayaufbau** – Kopf-, Fusszeile und Ausgabebereiche, grafische Elemente, Trennelemente,
- **Ausgabeelemente** – Menüeinträge, Editoren, Textausgaben, formatierte Ausgaben wie Uhrzeit und Datum, Spezialelemente
- **Darstellung** - Anzahl der dargestellten Zeilen, verwendete Fonts, Schriftarten wie Fett und Unterstrichen, Symbole und Animationen und deren Bedeutungen
- **Tastenbelegung** – Funktion der Tasten (unabhängige, kontextbezogene) Softkeys, Zustände für einfachen, mehrfachen und dauerhaften Druck der Tasten
- **Navigation** – Auswahl von Elementen, Menüwechsel, Ein- und Aussprung von Editoren, Abbrechen, Zurückgehen, Zustand nach Timeouts
- **Zusätzliche Ausgabemedien** – Ausgaben über Summer und LEDs, Bedeutung und Ausgabezeitpunkte
- **Geräteabhängigkeiten** – MMI-Abhängigkeiten vom Gerätezustand

Ohne Style-Guide ändert sich die MMI wie ein Chamäleon. Für den Anwender wird sie schwer zu durchschauen. Die Bedienung wird sehr komplex.

### 2.2 Struktur einer MMI

Die klassische Struktur einer MMI wird durch den Aufruf von Menüs einer festgelegten Menütiefe definiert. Sie stellt somit abstrakt eine Baumstruktur dar. Fehlermeldungen und besondere Ereignisse durchbrechen meist das starre Konzept.

Eine dynamische Netzstruktur ist die bessere Alternative. Dynamisch heißt, die Vernetzung der logischen Einheiten (Screens) muss nicht zum Zeitpunkt der Kompilierung festgelegt

werden, sondern zur Laufzeit durch das System. Ereignisgesteuerte Aufrufe von Einheiten sind somit unabhängig vom Status der MMI. In einer Netzstruktur lassen sich gut Baumstrukturen und deren Ausnahmen abbilden.

Eine zusätzlich geführte Historie der aufgerufenen Einheiten ermöglicht das Zurückspringen auf zuvor aufgerufene Einheiten (Vorgänger) und verhindern das Navigieren im Kreis (durch Aufruf eines Vorgängers).

### 2.3 Basiselemente

Die logischen Einheiten (Screens) setzen sich aus einzelnen Basiselementen (sog. Items) zusammen. Zu den Basiselementen einer MMI gehören:

- Menüeinträge
- statische und dynamische Texte
- editierbare Texte und numerische Felder
- Auswahlfelder
- Select-Felder und Buttons

Die Basiselemente besitzen wiederum allgemeine (z.B. links-/rechtsbündig) und spezielle (z.B. Schrittweite) Eigenschaften. Besonders zu beachten sind die editierbaren Felder. Beim Editieren sind die Eigenschaften der dahinter stehenden Variable zu berücksichtigen. Zum Beispiel sind Werte für Uhrzeiten auf 24 und 60 begrenzt, bestimmte Zahlenwerte werden nur in einer definierten Schrittweite geändert und die Eingabe einer E-Mail-Adresse benötigt ein '@'.

### 2.4 Dialoge, Menüs und Benachrichtigungen

Aus den einzelnen Elementen entstehen kombinierte Objekte, wie z.B. statischer Text kombiniert mit einem Editfeld.

Aus mehreren Elementen setzt sich eine logische Einheit zusammen. Nicht alle Elemente lassen sich immer gleichzeitig auf dem Display darstellen. In diesem Fall müssen die Elemente einer Einheit gescrollt werden. Elemente, die editiert werden, und Menüpunkte müssen sich selektieren lassen. Scroll-Balken und -bereich sind voneinander abhängig. Zu beachten sind auch die Sonderfälle: Was passiert zum Beispiel mit dem Scroll-Balken, wenn alle editierbaren Felder nicht im sichtbaren Bereich sind?

### 2.5 Berechtigungen und Systemabhängigkeiten

Menüstruktur, Ausgaben und Editierbarkeit können Abhängigkeiten zum System und zum Nutzer aufweisen. Ein Administrator oder Installateur hat oft mehr Rechte und Freiheitsgrade in der Konfiguration als ein normaler Benutzer. Hier ist eine MMI Bibliothek, die eine Rechteverwaltung unterstützt vorteilhaft, da die Zusammenstellung der Menüs, und damit die MMI Implementierung, sich nicht ändert. Es werden lediglich in Abhängigkeit von den Rechten einzelne Menüpunkte ein- oder ausgeblendet.

Auch der Ausbau des Geräts oder die Produktvariante haben einen Einfluss auf die MMI. Sind einzelne Komponenten in einer Variante des Systems nicht bestückt, ist der Versuch einer Ausgabe überflüssig und kann im schlechtesten Fall zu Fehlern führen oder aber den Benutzer in die Irre führen und Fehlfunktionen vermuten lassen. Hier gilt das gleiche wie für die Berechtigungen – es ist sinnvoll diese Menüpunkte auszublenden.

## 2.6 MMI-Spezifikation

Eine klare und visuell aufbereitete Spezifikation der MMI ist sehr wichtig für den Projekterfolg. Schließlich hat gerade die MMI eine große Wirkung auf den zukünftigen Benutzer.

Die Vernetzung der logischen Einheiten (Screens) lässt sich gut grafisch darstellen. Grafische Tools, wie z.B. Visio, helfen bei der Erstellung am PC. Wichtig ist, dass neben der grafischen Darstellung der Struktur auch die textuelle Beschreibung der Inhalte erfolgt.

Die Beschreibung der Inhalte sollte unabhängig von dem Quellcode der MMI geschehen. Dadurch können die Texte der Displayausgaben an einem Übersetzungsbüro zur Erstellung mehrerer Sprachversionen weitergegeben werden. Zusätzlich lassen sich die Ausgaben schnell und flexibel an neue Anforderungen vom Produktmanagement oder Marketing anpassen.

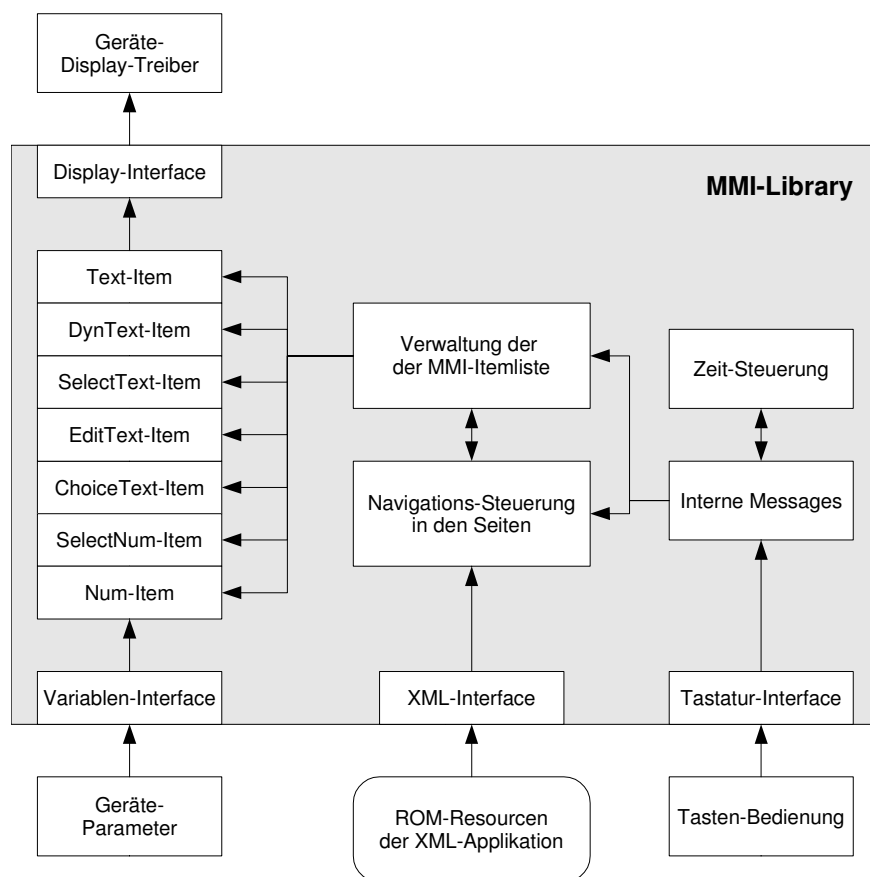
Eine geeignete Beschreibungssprache für den Inhalt der MMI ist XML. Beschrieben werden kann die MMI mit einem beliebigen XML-Werkzeug. Diese sind sowohl als kostenpflichtige wie auch als freie Tools (z.B. das Tool BonFire, das eine Überprüfung der Syntax und Verlinkung ebenso enthält wie verschiedene Gliederungen) erhältlich.

Die in XML entwickelte MMI wird in einem zweiten Schritt über ein mitgeliefertes Tool in das im System verwendete interne Format konvertiert. Als Ergebnis entsteht dann für die MMI eine Source- und eine Header-Datei, die einfach in das Projekt eingebunden wird.

### 3 Übersicht der sevenstax MMI-Bibliothek

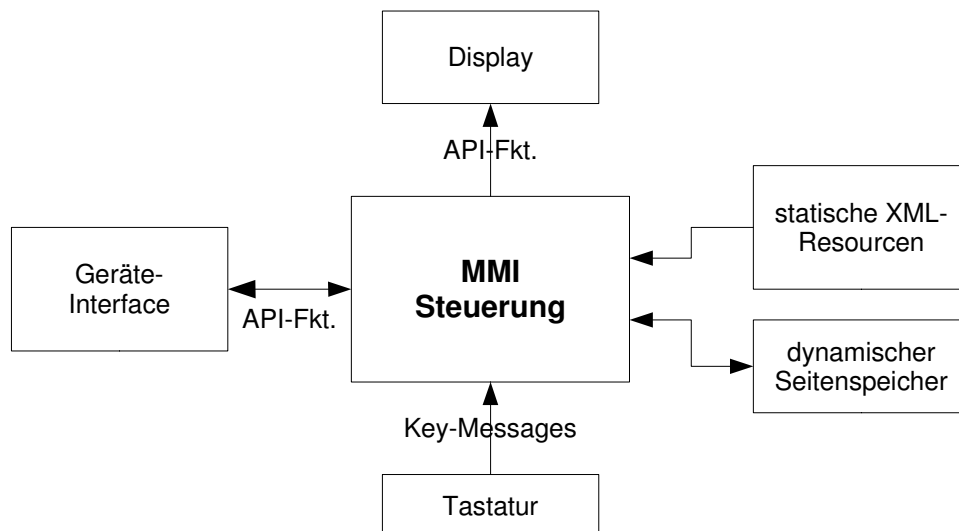
#### 3.1 Aufbau der MMI-Library

Die MMI setzt sich aus kleinen Einheiten, genannt Items, zusammen. Es gibt diese für verschiedenste Aufgaben, etwa für die Ausgabe von Text oder die Eingabe von numerischen Werten. Mehrere dieser Items werden i.a. gleichzeitig auf dem Display dargestellt und bilden dort eine Einheit, den 'Screen'. Die Verbindung und Kombination der Screens bildet dann letztendlich die gesamte MMI. Die Erstellung der Screens und der MMI ist Aufgabe des Anwenders der sevenstax MMI Bibliothek. Durch geeignete Unterstützung durch die Bibliothek, beispielsweise durch Container-Items zur Gruppierung, wird ihm dabei ein hohes Maß an Unterstützung geboten, so dass sich die ihm gestellten Aufgaben schnell und sicher umsetzen lassen.



### 3.2 MMI-Integration in ein System

Die folgende Grafik zeigt wie die sevenstax MMI in ein embedded System integriert wird und die hierfür benötigten Schnittstellen. Einzelheiten zu diesen Schnittstellen finden sich im Kapitel 'Anforderungen an das System'



- Display-Interface: Treiber-API zur Ansteuerung des Displays. Die MMI-Bibliothek nutzt diese API zur Ausgabe auf dem Display.
- Geräte-Interface: API zum Auslesen und Setzen von Gerätedaten.
- Tastatur-Interface: Treiber-API zur Tastatur des Systems. Die Auswertung der betätigten Tasten erfolgt in der MMI-Bibliothek.
- statische XML-Ressourcen: ROM-XML-Seiten mit kompletter MMI-Grundstruktur. In dieser Form wird die sevenstax MMI ressourcensparend im ROM des Systems abgelegt. Das interne Format wird über einen mitgelieferten Konverter aus der XML-Beschreibung der API generiert. Einzelheiten zum XML-Format und zur Erstellung der API finden Sie in den folgenden Kapiteln.
- dynamischer Seitenspeicher: RAM-Speicher zur Steuerung der angezeigten Seite.

## **4 Anforderungen an das System**

### **4.1 Tastaturtreiber**

- Der Tastaturtreiber ist nicht Bestandteil der sevenstax MMI und muss vom Anwender erstellt werden. Der Treiber sollte eine Entprell-Routine enthalten.
- Die sevenstax MMI stellt Funktionalität zur Verfügung, die vom Treiber aufgerufen werden muss
- Übergabeparameter: vom Treiber zur MMI-Software sind zwei Parameter, die zum einen einen Tastaturcode enthalten und zum zweiten eine Beschreibung der Aktion, ob z.B. die Taste gedrückt oder über längere Zeit betätigt wurde. Hierdurch wird es in der MMI möglich auf eine Taste in verschiedener Weise zu reagieren. Die beiden Parameter können beispielsweise folgende Werte annehmen:

Keycode: UP / DOWN / OK / ESC

Status: SINGLEPRESSED / REPEATPRESSED / RELEASED

### **4.2 Displaytreiber**

#### **Allgemeine Anforderungen**

- Der Displaytreiber ist nicht Bestandteil des Lieferumfangs der sevenstax MMI. Er muss vom Anwender erstellt werden.
- Die Art der HW-Anbindung (Memory-mapped, Parallelport, seriell) ist irrelevant.
- Die Display-HW sollte nicht zu langsam sein, z.B. kann ansonsten eine serielle Display-anbindung evtl. zu Performance-Einbußen führen.
- Die MMI erwartet einen definierten einfachen Funktionssatz.
- MMI-Ausgaben sind ausschließlich zeilen- und zeichenbasiert, dadurch für Character- und Grafik-Displays geeignet.
- Eine ggf. erforderliche individuelle pixelgenaue Ausrichtung (bei Grafik-Displays) einer Zeile muss vom Treiber erledigt werden.
- Die MMI benutzt für Grafik-Displays eigene Fonts, diese Zeichen werden als BMP ausgegeben.
- MMI-Ausgaben sind ressourcenschonend ausgelegt (immer nur 1 Zeile / MMI-Call, ein Displayzugriff erfolgt nur wenn Veränderung in Anzeige auftritt)

#### **Benötigte Display-Treiber-API**

- Display-Eigenschaften (Abmessungen, Text/Grafik-Display)
- Display-Clear
- Zeichen-/Cursorpositionierung
- Ausgabe eines Zeichens und eines Strings
- optional: Einstellung des Anzeige-Modus' ( INVERS / UNDERLINED / BOLD )
- optional: Einstellung des Cursor-Modus' ( ON / OFF / FLASH / UNDERLINED )

- Grafik-Display: Pixel-Cursor-Positionierung, Ausgabe einer BMP

### 4.3 MMI-Systemaufrufe

Zur Versorgung der MMI mit Rechenzeit gibt es drei Funktionen, die von dem embedded System aufgerufen werden müssen:

- Die sevenstax MMI Bibliothek enthält eine Message-Queue, die für die interne Kommunikation, auch z.B. mit der Tastatur verwendet wird. Diese Message-Queue muss mit Rechenzeit versorgt werden, wobei die Funktion nur kurze Laufzeiten hat. Für eine gute Bedienbarkeit, z.B. bei Tastendrücken sollte sie häufig aufgerufen werden.
- Bestimmte Funktionen sollten regelmäßig aufgerufen werden, wie beispielsweise die Screen-Update-Funktionen, (typischer Weise z.B. alle 20 ms), damit beim Scrollen kein sichtbares Ruckeln oder Springen auftritt. Auch hier arbeitet die Software rechenzeit-sparend, pro Call wird nur 1 Zeile aufgefrischt
- Für zeitlich exakte Operationen wird ein interner Counter gepflegt, der einmal pro Millisekunde, z.B. per Interrupt-Service-Routine, aufzurufen ist.

## **5 Features der sevenstax MMI**

Die folgende Liste gibt einen Überblick über unterstützte Features der sevenstax MMI:

- Unterstützung von monochromen Text- und Grafik-LCD-Displays
- Keine blockierenden Zustände innerhalb der MMI-Bibliothek. Durch internes Message-Handling wird nur das notwendige Minimum an Rechenzeit benötigt und die Systemsoftware nicht blockiert.
- Die Ausgabe erfolgt ausschließlich über die API des Displaytreibers.
- Eingaben erfolgen ausschließlich über den Tastaturtreiber.
- Mehrere statische/dynamische Elemente sind in einer Zeile darstellbar.
- Unterstützung von Kopf- und Fußzeilen: In der Fußzeile wird eine optionale dynamische Tastenbelegung unterstützt.
- Items können 1- oder 2-zeilig dargestellt werden, innerhalb eines Screens aber immer mit der gleichen Zeilenzahl.
- Inversdarstellung einzelner Worte/Werte wird unterstützt.
- Unterstreichen einzelner Worte/Werte wird unterstützt.
- Scrollen wird unterstützt, d.h. Screens können größer als das Display sein. Unterstützt wird das Scrollen hoch/runter mit oder ohne Anschlag.
- Eigener 6x8-Font für gute Lesbarkeit und Darstellbarkeit auf grafikfähigen LCD-Displays. Der Standard-Font enthält die gängigen westeuropäischen Fremdsprachen-Zeichen. Eigene Zeichen können definiert werden. Der Font ermöglicht z.B. auf einem Display mit 128x64 Pixeln die Darstellung von 21x7 Zeichen, inklusive jeweils einer Leerpixelzeile zur Trennung.
- Ausrichtung von Texten innerhalb eines Items möglich, z.B: Text linksbündig, Variablen und Einheit rechtsbündig.
- Dynamische Elemente können kontinuierlich angezeigt werden, um beispielsweise Änderungen anzuzeigen.
- Das dynamische Ein-/Ausblenden von Items wird unterstützt. Hiermit lassen sich beispielsweise Menüpunkte für bestimmte Benutzer ausblenden oder die MMI ohne Anpassung für verschiedene HW-Konfigurationen (z.B. unterschiedlich bestückte Sensoren) nutzen.

## **6 Übersicht MMI-Items**

MMI-Items stellen die Abbildung der MMI-Elemente in der Software dar.

Sie sind die Grundbausteine, aus denen ein beliebiger Screen zur Laufzeit zusammengesetzt werden kann. Dies ist eine Liste der MMI-Items, wie sie die sevenstax MMI-Software-Library zur Verfügung stellt.

Z.Zt. sind diese MMI-Items definiert:

- Text-Item                                    unveränderlicher Text
- NumX-Item                                   numerische Werte, verschiedene Formate
- ChoiceText-Item                            ein Text aus einer Auswahl von Texten

- Container-Item kann mehrere MMI-Items (keine Container-Items) aufnehmen
- EditText-Item Editor für beliebigen Text
- SelectText-Item änderbare Auswahl eines Text
- SelectNumX Editor für numerische Werte, verschiedene Formate

### **Die Items in der Kurzübersicht:**

#### **Statisches Textanzeige-Element**

- unveränderlicher Text
- nicht editierbar
- z.B. Bezeichnung eines Messwertes oder Menüpunktes
- Ausrichtung im Element: links / rechts / zentriert
- Feldbreite: feste Länge, nicht genutzte Stellen werden mit Leerzeichen aufgefüllt

#### **Dynamisches Textanzeige-Element**

- veränderlich, wird permanent aufgefrischt
- nicht editierbar
- dynamische Werte, z.B. aktuelle Temperatur oder String
- Ausrichtung im Element: links / rechts / zentriert
- Feldbreite: feste Länge, nicht genutzte Stellen werden mit Leerzeichen aufgefüllt

#### **Textauswahl-Element**

- Liste mit vordefinierten statischen Texten
- editierbar: Auswahl mit Up/Down-Tasten („SelectBox“)
- nur jeweils der selektierte Text kommt zur Anzeige
- Ausrichtung im Element: links / rechts / zentriert
- Feldbreite: feste Länge, nicht genutzte Stellen werden mit Leerzeichen aufgefüllt

#### **Numerisches Editor-Element**

- formatierte Ausgabe eines numerischen Wertes (ausschliesslich Ziffern 0..9)
- editierbar: Werte mit Up/Down-Tasten („Counter“)
- Edit-Mode: Schrittweite, Min-Wert, Max-Wert, ink./dekr. evtl. dynamisch
- Formatierung: optionales Trennzeichen an definierten Position
- Ausrichtung im Element: links / rechts / zentriert
- Feldbreite: feste Länge, nicht genutzte Stellen werden mit Leerzeichen aufgefüllt

#### **Übergeordnete Element-Eigenschaften**

Für ein menüorientiertes Bedienkonzept ist eine zeilenorientierte Anordnung der Elemente sinnvoll. In jeder Zeile können jedoch mehrere MMI-Elemente auftreten.

Die Anordnung der Elemente in einer Zeile geschieht über die Definition der Feldbreite. Innerhalb des Feldes wird der Inhalte des Elements ausgerichtet. Das nachfolgende Element wird direkt rechts vom Feldende des Vorgängers angeordnet. Dies hat den Vorteil, dass innerhalb des Feldes genug Platz für weitere Sprachvarianten des Elements bleibt, ohne die Positionierung verändern zu müssen.

Durch die Ausrichtung innerhalb des Feldes kann auf Leerzeichen in den MMI-Ressourcen verzichtet werden. Nötige Leerzeichen werden durch die MMI-Software an der gewünschten Stelle automatisch eingefügt.

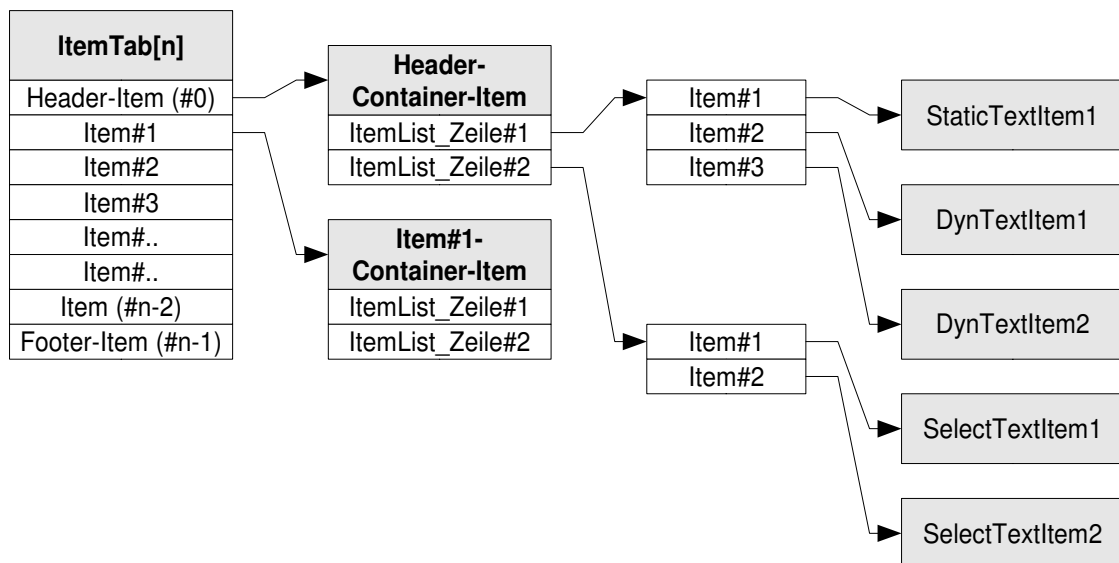
Um Ressourcen zu sparen, sind editierbare und nicht editierbare Varianten der o.a. Elemente separat in der Software ausgeführt.

### Lebenszeit eines MMI-Items

Beim Öffnen eines neuen Screens wird die Ressourcendatei aus dem ROM gelesen. Die Ressourcen werden nach Schlüsselwörtern durchsucht. Dabei entsteht eine Liste von MMI-Items. Jedes neu generierte MMI-Item wird auf dem Heap abgelegt. Sie werden wieder vom Heap entfernt, wenn der Screen verlassen wird.

### 6.1 MMI-Item-Hierarchie innerhalb eines Screens

Alle Elemente eines Screens werden in Form von Items verwaltet. Die Grundstruktur eines Screens ist die sog. ItemTab, eine Tabelle, die ausschliesslich Zeiger auf ihr unbekannte Items enthält. Einträge in der ItemTab sind immer Zeilen-orientiert, ein neues Item beinhaltet also immer auch die Nutzung einer neuen Zeile im Screen.



Ein Eintrag in der ItemTab kann auf ein Container-Item verweisen. Dieses kann weitere Sub-Items in einer Display-Zeile aufnehmen. Dadurch können Items auch innerhalb einer Zeile horizontal positioniert werden. Die Anzahl der Items pro Zeile ist z.Zt. auf 8 beschränkt. Ein Container-Item nimmt z.Zt. max. 2 Zeilen auf.

## Speicherbelegung eines MMI-Items

Die u.a. Aufteilung ROM / Heap / Stack gibt eine Übersicht über die ungefähre Ressourcenbelastung durch ein MMI-Item zur Lebenszeit:

- "ROM-Data" sind die Teile des Items, die im ROM verbleiben und auf die bei jedem Zugriff per Pointer zugegriffen wird. Dies können z.B. statische Strings sein.
- "Heap-Data" sind die Teile des Items, die dynamisch sind und zur Lebenszeit des Items erhalten bleiben müssen. Dies ist z.B. die horizontale Ausrichtung des Items.
- "Stack-Data" sind die Teile des Items, die bei jedem Refresh des Items neu angelegt, genutzt und wieder verworfen werden. Dies kann z.B. der anzuzeigende Messwert sein.

## 6.2 Items der sevenstax MMI Bibliothek

### Nicht editierbare MMI-Items

<i>Item</i>	<i>Aufgabe + Verhalten</i>
Text-Item	<ul style="list-style-type: none"> <li>• bringt statischen Text zur Anzeige</li> <li>• fokussierbar (Invers-Darstellung)</li> <li>• Ausgabe mit Feldbreite / Ausrichtung</li> <li>• einzeilige Ausgabe</li> <li>• kann Sub-Item eines Containers sein</li> </ul>
DynText-Item	<ul style="list-style-type: none"> <li>• bringt ausschliesslich dynamischen Text zur Anzeige</li> <li>• fokussierbar (Invers-Darstellung)</li> <li>• Ausgabe mit Feldbreite / Ausrichtung</li> <li>• einzeilige Ausgabe</li> <li>• kann Sub-Item eines Containers sein</li> </ul>
NumS8-Item NumS16-Item NumS32-Item  NumU8-Item NumU16-Item NumU32-Item	<ul style="list-style-type: none"> <li>• dient ausschliesslich der numerischen Ausgabe eines Wertes</li> <li>• Item ist Wert-Typ-spezifisch (8/16/32, signed, unsigned)</li> <li>• unterstützt bel. Trennzeichen an bel. Position</li> <li>• Ausgabe mit Feldbreite / Ausrichtung</li> <li>• einzeilige Ausgabe</li> <li>• fokussierbar (Invers-Darstellung)</li> <li>• kann Sub-Item eines Containers sein</li> </ul>
ChoiceText-Item	<ul style="list-style-type: none"> <li>• Anzeige eines ausgewählten Textes</li> <li>• intern rein numerisch, Anzeige nur mit ROM-Pointern</li> <li>• Ausgabe mit Feldbreite / Ausrichtung</li> <li>• einzeilige Ausgabe</li> <li>• fokussierbar (Invers-Darstellung)</li> <li>• kann Sub-Item eines Containers sein</li> </ul>
Container-Item	<ul style="list-style-type: none"> <li>• Hauptelement eines Menüs, kann Links ausführen</li> <li>• kann weitere Items aufnehmen (Liste, malloc)</li> <li>• ruft Funktionen der Sub-Items auf</li> <li>• hat keinen eigenen Text</li> <li>• Ausdehnung einzeilig / zweizeilig</li> <li>• fokussierbar (gibt Fokus an alle Sub-Items)</li> <li>• beherrscht im Edit-Mode Support der Sub-Elemente</li> <li>• beherrscht Sequenz für Edit-Reihenfolge</li> <li>• unterstützt Editmode-Abbrechen für alle Sub-Items</li> </ul>

**Editierbare MMI-Items**

<b>Item</b>	<b>Aufgabe + Verhalten</b>
EditText-Item	<ul style="list-style-type: none"> <li>• Zeichenweiser Editor für bel. Text</li> <li>• Anzeige zwischen Klammern</li> <li>• Einzelzeichen-Edit-Mode (Blink-Cursor, wandernd)</li> <li>• fokussierbar (Invers-Darstellung)</li> <li>• Edit-Ende am Feldende (keine Auswertung langer Tastendruck)</li> <li>• virtueller Edit-Bereich mgl. breiter als Anzeigebereich</li> </ul>
SelectText-Item	<ul style="list-style-type: none"> <li>• Anzeige + Editierung eines ausgewählten Textes</li> <li>• Reaktion auf Up/Down-Taste</li> <li>• intern rein numerisch, Anzeige nur mit ROM-Text-Pointern</li> <li>• Ausgabe mit Feldbreite / Ausrichtung</li> <li>• einzeilige Ausgabe</li> <li>• fokussierbar (Invers-Darstellung)</li> <li>• kann Sub-Item eines Containers sein</li> </ul>
SelectNumS8-Item SelectNumS16-Item SelectNumS32-Item  SelectNumU8-Item SelectNumU16-Item SelectNumU32-Item	<ul style="list-style-type: none"> <li>• numerischer Editor über Up/Down-Count</li> <li>• Editor beherrscht Min/Max/Stepsize</li> <li>• Item ist Wert-Typ-spezifisch (8/16/32, signed, unsigned)</li> <li>• unterstützt bel. Trennzeichen an bel. Position</li> <li>• Ausgabe mit Feldbreite / Ausrichtung</li> <li>• einzeilige Ausgabe</li> <li>• fokussierbar (Invers-Darstellung)</li> <li>• kann Sub-Item eines Containers sein</li> </ul>

**7 XML-Ressourcen**

Die MMI der Applikation wird in einer XML-ähnlichen Form erstellt. Diese MMI-Ressourcen entkoppeln die Anzeigefunktionen der MMI von den Inhalten der Applikation.

Um die MMI der Applikation vom rohen Entwurf bis zur Displayanzeige zu bringen, sind diese Schritte nötig:

1. MMI der Applikation in XML beschreiben -> XML-File
2. Konvertieren in C-Sourcecode -> C-File
3. Hinzulinken zum embedded Projektcode -> ROM-Data
4. Zur Laufzeit parsen und MMI-Items generieren -> Item-Tabelle im Heap
5. Items updaten und ausgeben -> Display-Ausgabe

**7.1 Übersicht der verwendeten XML-Syntax**

XML erlaubt die Erweiterung des XML-Sprachschatzes und die Definition eigener Attribute. Als Basis dieser MMI-Definition diente WAP, einer auf kleine Anzeigegeräte angepassten XML-Version.

Tags liegen in der Ressource-Datei immer symmetrisch mit einem Open-Tag (z.B. „<html>“) und einem Close-Tag (z.B. „</html>“) vor. Jedes Tag kann mehrere Attribute enthalten. Für nicht definierte Attribute werden Defaultwerte angenommen

## 7.2 Beispiel-Screen-XML-Datei

Dies ist ein Beispiel einer XML-Ressource-Datei, um die Nutzung der XML-Syntax zu veranschaulichen. Dieser Beispiel-Screen soll dargestellt werden:

Einstellungen	1/2
Drehzahl	1.245
Uhrzeit	23:12
Montag	12.09.03
zurück	ändern

Dies ist der XML-Code zum o.a. Beispiel-Screen:

```
<html name="Appl deutsch">
  <screen name="settings" id="5" >
    <Kopfzeile>
      <text>Einstellungen</text>
      <dyntext var="int:screen_id" width="2"/>
      <dyntext var="int:entry_id" width="2"/>
    </Kopfzeile>
    <body entryheight="2">
      <entry id="2">
        <text>Drehzahl</text>
        <num var="ext:drehzahl" width="4" dotpos="3" dotchar="." type="U16"/>
      </entry>
      <entry id="3">
        <text size="15">Uhrzeit</text>
        <selectnum var="ext:hour" width="2"/>
        <text width="1">:</text>
        <selectnum var="ext:minute" width="2"/>
        <br/>
        <selecttext var="ext:dayofweek" width="12">
          <option>Montag</option>
          <option>Dienstag</option>
          <option>Mittwoch</option>
          <option>Donnerstag</option>
          <option>Freitag</option>
          <option>Samstag</option>
          <option>Sonntag</option>
        </selecttext>
        <selectnum var="ext:day" width="2" />
        <text>.</text>
        <selectnum var="ext:month" size="2"/>
        <text>.</text>
        <selectnum var="ext:year" size="2"/>
      </entry>
      <entry id="4">
        </entry>
    </body>
    <Fußzeile>
      <choicetext var="int:keyleft" width="10">
        <option></option>
        <option>zurück</option>
        <option>abbrechen</option>
      </choicetext>
```

```
<choicetext var="int:keyright" align="right" width="10" >
  <option></option>
  <option>ändern</option>
  <option>auswählen</option>
  <option>bestätigen</option>
</choicetext>
</Fußzeile>
</screen>
</html>
```

### 7.3 XML-nach-C-Konverter

Der mitgelieferte Konverter ermöglicht das automatische Konvertieren der in XML erstellten MMI in das intern verwendete Format. Dieses Format setzt die MMI ressourcensparend in eine/mehrere Sourcecode- und Header-Datei(en) um, die dann in das Projekt integriert wird/werden. Das interne Format ist reines ANSI C und daher mit beliebigen C-Compilern übersetzbar. Das Ergebnis wird dann in der Regel über den Linker in das ROM des Systems gemappt.

## 8 Änderungsverzeichnis

<b>Vers.</b>	<b>Datum</b>	<b>Verantw.</b>	<b>Änderung</b>
1.0	23.09.2003	krf	Erstellt
1.1	10.11.03	bert	Erweitert
1.2	25.11.03	ssu	Korrektur gelesen
1.3	23.12.03	krf	Korrektur gelesen