



**MKC Michels & Kleberhoff Computer GmbH**

Vohwinkeler Str. 58, D-42329 Wuppertal

Tel.: ++49 (0)202 27317 0 Fax: ++49 (0)202 27317 49

Internet: <http://www.mkc-gmbh.de>

**System-Programmierer**

**Handbuch**

### **Hinweise:**

Die Informationen in diesem Handbuch wurden sorgfältig zusammengestellt und überprüft. Dieses Handbuch wird stetig auf dem aktuellen Zustand gehalten. Jedoch wird von **MKC** keine Gewähr für fehlerhafte Informationen übernommen.

**MKC** behält sich das Recht vor, jederzeit ohne weitere Ankündigung technische Änderungen zur Verbesserung der Zuverlässigkeit, der Funktion oder des Designs der Software und Überarbeitungen des Handbuchs durchzuführen. Änderungen des Handbuchs zwischen 2 Ausgaben werden im Text nicht markiert.

Das Datum einer Ausgabe bezieht sich auf das Handbuch. Dieses muss nicht mit dem Datum der Änderung der Software übereinstimmen. Bei der Versionsgeschichte wird der Grund für die Handbuch Änderungen genannt.

**MKC** übernimmt keine Haftung für die Anwendung der hier beschriebenen Software. **MKC** übernimmt weiterhin keine Haftung für Schäden oder Folgeschäden, die durch Verwendung dieser Software entstehen. Diese Haftungseinschränkung bezieht sich sowohl auf jeden direkten Abnehmer sowie auf alle seine Kunden und alle Anwender dieser Software.

Mündliche Zusagen über die Anwendbarkeit dieser Software gelten als nicht erfolgt.

Die unten angegebenen Lieferversionen sind zur Zeit verfügbar. Damit ist nicht zugesagt, dass alle diese Versionen weiterhin lieferbar bleiben. **MKC** behält sich das Recht vor, die Produktion dieser Software aus technischen Gründen ohne vorherige Ankündigung einzustellen.

### **Kommentare:**

Kommentare oder Korrekturen jedweder Art sind dem Autor jederzeit willkommen. Senden Sie sie bitte an:

**MKC Michels & Kleberhoff Computer GmbH**  
**Vohwinkeler Str. 58**  
**42329 Wuppertal**

oder

**info@mkc-gmbh.de**





## Inhaltsverzeichnis

<b>1</b>	<b>EINLEITUNG.....</b>	<b>9</b>
1.1	Zielsetzung.....	9
1.2	Konventionen.....	9
<b>2</b>	<b>AUFBAU DES SYSTEMS .....</b>	<b>10</b>
2.1	Prozessor.....	10
2.1.1	FLASH-Speicherbereich (Intern).....	10
2.2	SDRAM.....	11
2.3	NAND-FLASH.....	11
2.4	eWebSrv FPGA.....	11
2.5	Trägerkarte (Extension).....	11
<b>3</b>	<b>BETRIEBSSYSTEM ( EWEBSRV-MODUL).....</b>	<b>12</b>
3.1	Systemstart.....	12
3.1.1	BootLoader.....	12
3.1.2	Boot-Prozess.....	12
3.2	Netzwerk.....	15
3.2.1	Konfiguration über die serielle Schnittstelle.....	15
3.2.2	Konfiguration über die Ethernet-Schnittstelle.....	16
3.2.2.1	Einstellen über Server-Dienst.....	16
3.3	Treiber.....	17
3.3.1	NAND-Flash Hardware Treiber.....	17
3.3.2	Serielle Schnittstellen.....	17
3.3.3	MKC Daten-Strukturen.....	17
3.3.4	Datenpunkte.....	17
3.3.4.1	Unit-Devices.....	18
3.3.4.2	Application-Devices.....	20
3.4	System-Schnittstelle SYSFS.....	21
3.4.1	Treiber-Attribute.....	21
3.4.2	System-Attribute: CONFIG und PARAM.....	22
3.4.2.1	CONFIG Zweig eWebSrv.....	23
3.4.2.2	PARAM Zweig eWebSrv.....	24
3.4.3	Device-Attribute: Datenpunkte.....	25
3.4.3.1	Unit-Devices.....	25
3.4.3.2	Application-Devices.....	26
3.4.3.3	Signal-Handling bei Datenpunkten.....	26
3.4.4	Anwendungs-Attribute.....	27
3.4.4.1	TCPCOMM Zweig eWebSrv.....	27
<b>4</b>	<b>BETRIEBSSYSTEM (FPGA UND EXTENSION).....</b>	<b>28</b>
4.1	Allgemein.....	28
4.2	Treiber für das eWebSrv FPGA.....	28
4.2.1	Grundfunktionen (FPGA).....	28
4.2.2	FPGA 4102 (IO48).....	28
4.2.3	FPGA 4104 (SERC3X).....	28
4.2.4	Treiber für serielle DSP Schnittstellen (SERC3X).....	29
4.3	Treiber für die eWebSrv Extension.....	32
4.3.1	EXT 4106 (eWebTest).....	32
4.4	System-Schnittstelle SYSFS.....	32
4.4.1	Grundfunktionen (FPGA) .....	32
4.4.2	FPGA 4102 (IO48).....	33

4.4.2.1	CONFIG Zweig FPGA 4102.....	33
4.4.2.2	PARAM Zweig FPGA 4102.....	33
4.4.3	FPGA 4104 (SERC3X).....	34
4.4.3.1	CONFIG Zweig FPGA 4104.....	34
4.4.3.2	PARAM Zweig FPGA 4104.....	34
4.4.4	EXT 4106 (eWebTest).....	35
4.4.4.1	CONFIG Zweig EXT 4106 .....	35
4.4.4.2	PARAM Zweig EXT 4106.....	36

## **5 ENTWICKLUNGSUMGEBUNG.....37**

5.1	Allgemein.....	37
5.1.1	Aktualisierung des Systems.....	37
5.2	Systemerstellung.....	38
5.2.1	Konfiguration.....	38
5.2.2	„Build“-Prozess.....	38
5.3	Installation/Wiederherstellung.....	39
5.3.1	Vorbereitungen.....	39
5.3.2	Prozedur.....	39

## **6 ANWENDUNGS-ENTWICKLUNG.....40**

6.1	Grundlagen.....	40
6.2	Kompletter Entwicklungs-Zyklus .....	40
6.2.1	Vorbereitungen.....	40
6.2.2	Erstellen einer neuen Anwendung.....	41
6.2.3	Installation der neuen Anwendung.....	41

## **7 ANHANG.....42**

7.1	Definitionen.....	42
7.1.1	Allgemein.....	42
7.2	Beschreibung des SDK.....	42
7.3	Weiterführende Informationen.....	42
7.3.1	MKC-Handbücher.....	42
7.3.2	Online-Material.....	42
7.4	Rechtliche Bestimmungen.....	42
7.4.1	Lizenzierung.....	42

## Liste der Abbildungen

Abbildung 1: eWebSrv.....	10
Abbildung 2: Flussdiagramm Systemstart.....	13
Abbildung 3: Flussdiagramm Linux Start.....	14
Abbildung 4: Makefile.....	41

## Liste der Tabellen

Tabelle 1: SYSFS: Datenpunkt Treiber.....	21
Tabelle 2: SYSFS: Hauptverzeichnis CONFIG/PARAM.....	22
Tabelle 3: SYSFS: CONFIG eWebSrv.....	23
Tabelle 4: SYSFS: Parameter eWebSrv.....	24
Tabelle 5: SYSFS: Attribute von Unit-Devices.....	25
Tabelle 6: SYSFS: Attribute von Datenpunkten (Unit-Devices).....	25
Tabelle 7: SYSFS: Attribute Application-Devices.....	26
Tabelle 8: SYSFS: Parameter TCPCOMM-Server.....	27
Tabelle 9: MKCSERC3X_IOCTL_STATUS.....	31
Tabelle 10: SYSFS: FPGA allgemein.....	32
Tabelle 11: SYSFS: CONFIG FPGA IO48.....	33
Tabelle 12: SYSFS: Parameter FPGA IO48.....	33
Tabelle 13: SYSFS: CONFIG FPGA SERC3X.....	34
Tabelle 14: SYSFS: Parameter FPGA SERC3X.....	34
Tabelle 15: SYSFS: CONFIG Extension eWebTest.....	35
Tabelle 16: SYSFS: Parameter Extension eWebTest.....	36
Tabelle 17: MKC-Verzeichnisse.....	40
Tabelle 18: SDK-Inhalt.....	42

### Liste der Abkürzungen

AIN	Analoger Eingangs-Datenpunkt
AOT	Analoger Ausgangs-Datenpunkt
Application-Device	Eintrag eines (speziellen) Zeichen-orientierten Geräts (Character-Device)
AV	Arbeitsverzeichnis der Entwicklungs-Umgebung
CFM	FLASH-INTERN, Interner Flash-Speicherbereich des Prozessors
CONFIG	Konfiguration (-Daten)
CONFIG-Bereich	Bereich der Konfiguration-Daten innerhalb des CFM
DIN	Digitaler Eingangs-Datenpunkt
DOT	Digitaler Ausgangs-Datenpunkt
DIO	Digitaler Ein-/Ausgangs-Datenpunkt, Varianten : IN, OUT und OUT_OD
DP	Datenpunkt(e)
eWebSrv-System	Ein Embedded System basierend auf dem eWebSrv
EXT	Extension, Erweiterungen die sich auf der Trägerkarte befinden
FLASH	FLASH-EXTERN, Speicherbereich des NAND-FLASH Bausteins
ID	Interne Identifikationsnummer
LOCAL	Unit-Konfigurationsdatei
PARAM	Parameter (-Daten)
PARAM-Bereich	Bereich der Parameter-Daten innerhalb des CFM
Public-Device	Applikation-Device „public“; immer auf einem eWebSrv-System vorhanden
REV	Interne Revisionsnummer
Unit-Device	Eintrag eines (speziellen) Zeichen-orientierten Geräts (Character-Device)
WC	„Working copy“, ein Speicherabbild des kompletten PARAM-Bereichs

# 1 Einleitung

## 1.1 Zielsetzung

Dieses Dokument ist in zwei Bereiche gegliedert. Der erste Bereich beschreibt die System-Schnittstellen und Treiber die zum Betrieb notwendig sind. Im zweiten Teil wird die komplette Entwicklungs-Umgebung beschrieben.

**Hinweis:** Sollten Sie die Handbücher „Hardware“, „Benutzer“ und „System-Anwender“ noch nicht gelesen haben, tun Sie dies bitte als erstes und fahren dann hier fort.

## 1.2 Konventionen

In den folgenden Kapiteln sind Anwahlen in Feldern oder Menüs **fett** und notwendige Eingaben des Benutzers **fett kursiv** angegeben. So ist zum Beispiel die Anwahl des Menüpunktes „Menü1“ und die Eingabe der Zahl 255 im Text folgendermaßen beschrieben: Anwahl **Menü1** und Eingabe **255**.

## 2 Aufbau des Systems

Es gelten die in den Handbücher Benutzer und System-Anwender beschriebenen Definitionen für Konfiguration, Parameter und Mess-/Stellwerte.

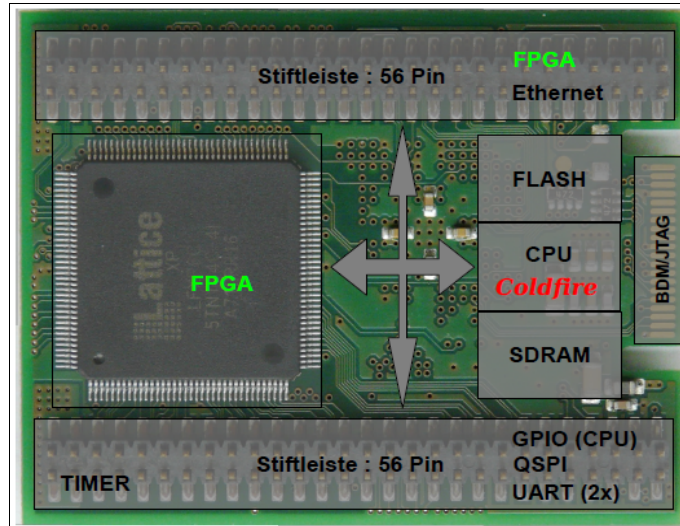


Abbildung 1: eWebSrv

Der eWebSrv lässt sich in die Subsysteme Prozessor, SDRAM, Flash-Baustein, FPGA und Trägerkarte (Extension) aufteilen.

### 2.1 Prozessor

#### 2.1.1 FLASH-Speicherbereich (Intern)

Die notwendige **Konfiguration** (CONFIG) des Gerätes erfolgt durch **mhc** während des „Produktionstests“ und wird im Testprotokoll dokumentiert. Dabei werden für alle im Gerät eingesetzten Platinen der Platinen-Name, die Platinen-Revision, die Bestückungs-Variante, die Seriennummer und das Produktions-Datum gespeichert. Für diese Daten werden im CFM (FLASH-INTERN) mehrere Kacheln (Pages) reserviert.

Jedes Gerät mit einem eWebSrv wird in einem definiertem Auslieferungszustand an den Anwender geliefert. Um Änderungen der **Parameter** (PARAM) speichern zu können, werden im CFM zwei Kacheln reserviert (CFM-Kacheln sind unabhängig voneinander löschar). Diese werden abwechselnd beschrieben.

Diese Vorgehensweise garantiert eine zuverlässige Aktualisierung, auch bei z.B. einem Stromausfall. Die alte Kachel bleibt gültig, bis die neue fehlerfrei geschrieben werden konnte.

Für die Sicherung der remanenten **Stellwerte** wird im CFM ein Speicherbereich aus mehrere Kacheln reserviert. Dieser Bereich wird als Ringspeicher betrieben, wobei immer alle Mess- und Stellwerte gesichert werden.

Programme im User-Mode können und dürfen nicht direkt auf diese Speicherbereiche zugreifen. Der Zugriff erfolgt entweder über das SYSFS Dateisystem oder über Devices mit Standard Funktionen ( 'open', 'read', 'write' und 'close').

## 2.2 SDRAM

In der derzeitigen Ausbaustufe beträgt der „freie“ Speicher 16 MByte abzüglich des Kernels, aktiver Programme und einer RAM-Disk.

## 2.3 NAND-FLASH

Der externe FLASH-Speicherbereich wird durch ein NAND-FLASH realisiert. Dieser Speicher wird über ein Flash-Filesystem (JFFS2) angesprochen und stellt dem Betriebssystem zwei Partitionen zur Verfügung. Die erste Partition wird für den Kernel (Verzeichnis '/boot') benutzt, während in der zweiten das übrige System residiert mit der für Linux-Systeme typischen Verteilung der Verzeichnisse und Dateien. Die Aufteilung ist 4 MByte für die erste und 28 MByte für die zweite Partition.

## 2.4 eWebSrv FPGA

Das FPGA des eWebSrvs kann verschiedene Funktionen realisieren, für die verschiedene Treiber bereitgestellt werden. Die Beschreibung dieser Treibermodule ist im Kapitel 4.2 zu finden.

## 2.5 Trägerkarte (Extension)

Der eWebSrv wird auf verschiedenen Trägerkarten eingesetzt, somit werden auch hier verschiedene Treiber benötigt. Die Beschreibung dieser Treibermodule ist im Kapitel 4.3 zu finden.

## 3 Betriebssystem ( eWebSrv-Modul)

### 3.1 Systemstart

#### 3.1.1 BootLoader

Im CFM des Prozessors ist ein BootLoader/Debugger implementiert. Dieser basiert auf dem dBUG von Freescale und beinhaltet verschiedene Erweiterungen. Wichtigste Funktion ist :

- den uClinux Kernel aus einem JFFS2 Flash-Filesystem zu laden und auszuführen.

#### 3.1.2 Boot-Prozess

In diesem Prozess wird das Gerät initialisiert. Hierzu werden die Daten aus dem CONFIG/PARAM-Bereich und der Mess-/Stellwerte im CFM ausgewertet.

Der BootLoader überprüft die jeweiligen CFM Bereiche auf gültige Header für Konfiguration, Parameter und Daten. Sind alle Bereiche gültig, wird Linux gestartet. Anderenfalls erscheint der dBUG-Prompt auf der Konsole (1. serielle Schnittstelle).

Linux übernimmt in der Regel keine Einstellungen eines BootLoaders. Jeder Treiber ist für die Initialisierung seiner Hardware selbst verantwortlich, es gibt keinen zentralen Prozess dafür.

Linux-Treiber sind entweder als Kernel-Treiber oder als nachladbare Kernel-Module implementiert. Erstere werden mit dem Kernel (/boot/linux.bin) zusammen geladen und initialisiert. Diese sind dann während der gesamten Laufzeit des Systems vorhanden. Die Kernel-Module werden zur Laufzeit des Systems geladen (modprobe), d.h. nachdem der Kernel geladen und ausgeführt wird und können teilweise auch entladen werden (rmmod). Somit entsteht der Vorteil den gesamten Linux-Kernel in einem statischen und einen dynamischen Teil aufteilen zu können.

Der Treiber für das NAND-FLASH muss zwingend als Kernel-Treiber vorliegen, da bereits beim Start des Kernels auf das Filesystem zugegriffen wird.

**Hinweis:** Auf der eWebTest befinden sich die seriellen Schnittstellen „term“ (1. Schnittstelle) und „aux“ (2. Schnittstelle). Unter einem Linux-System werden diese als „ttyS0“ bzw. „ttyS1“ bezeichnet ( Zählweise von „0“ beginnend).

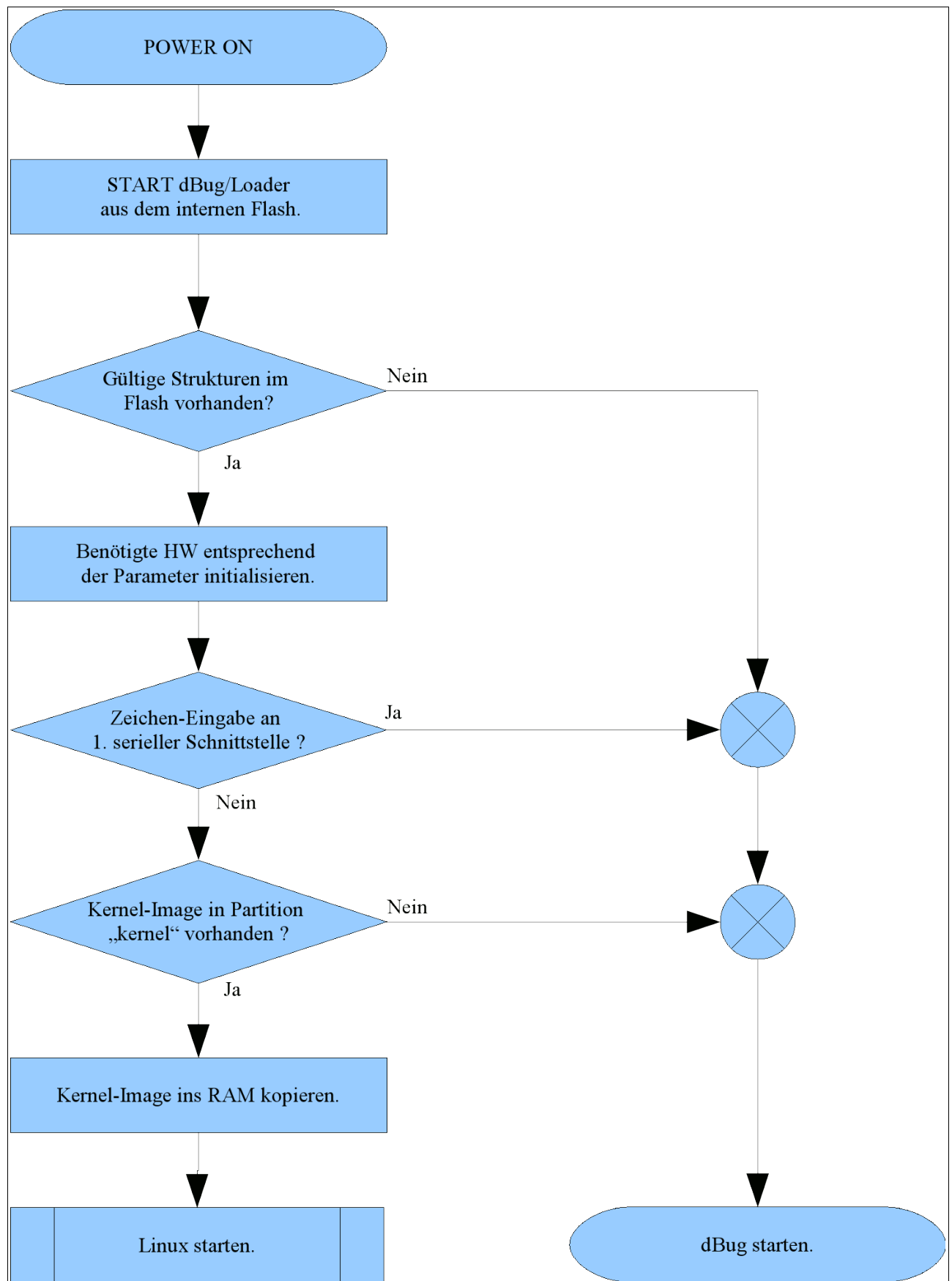


Abbildung 2: Flussdiagramm Systemstart

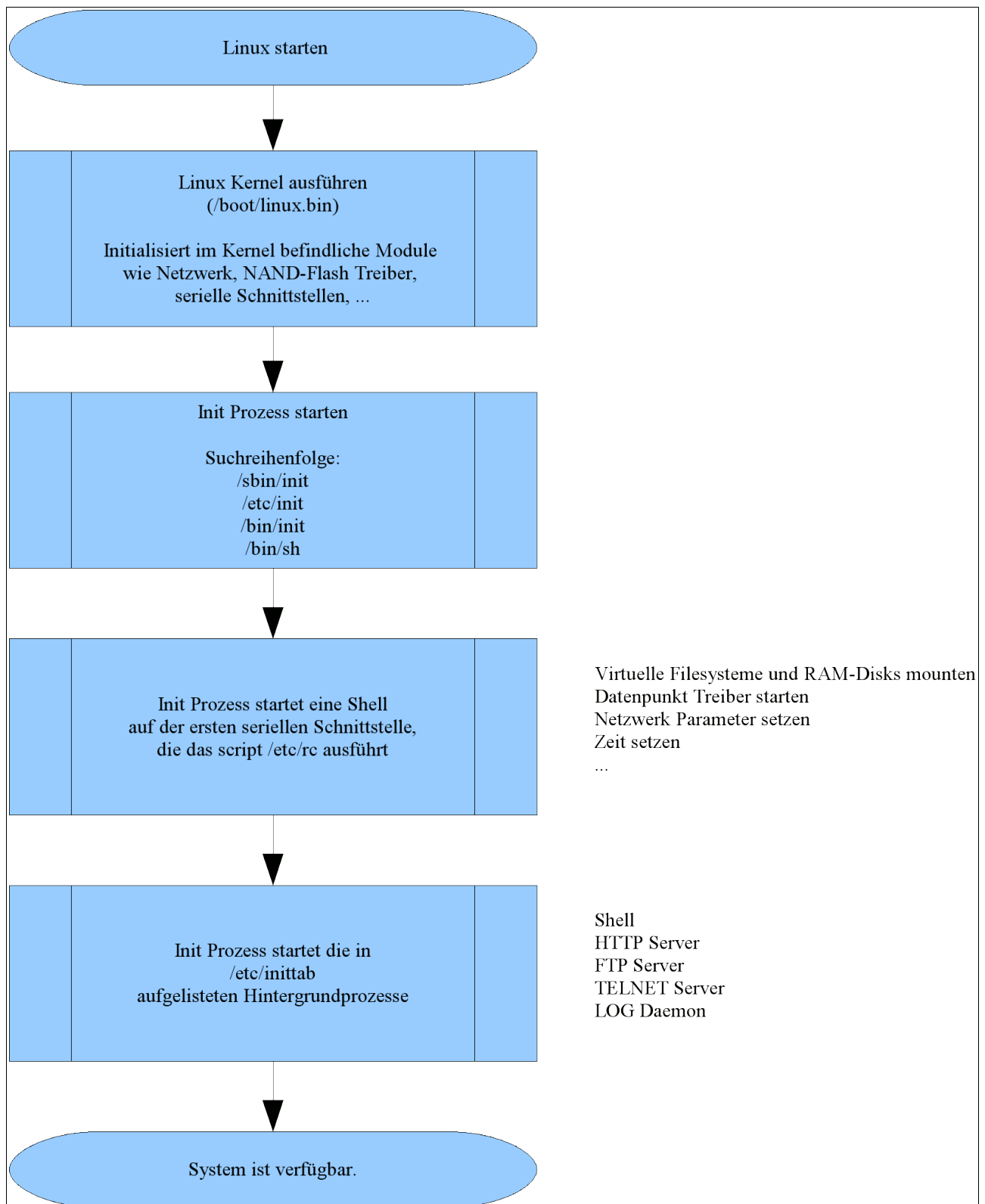


Abbildung 3: Flussdiagramm Linux Start

## 3.2 Netzwerk

Die Kommunikation mit dem Gerät erfolgt ausschließlich über ein IP-Netzwerk.

Beim Systemstart wird ein HTTP-Server und in Abhängigkeit von der Systemkonfiguration weitere Server (TELNET,FTP) aktiviert. Diese Prozesse sind unter der Kontrolle von inetd oder inittab und werden im Bedarfsfall automatisch neu gestartet

Der HTTP-Server ermöglicht es einen Standard-Browser (IE, Firefox, Opera, usw.) für die Bedienung einzusetzen. Über dieses Protokoll können die Mess-/Stellwerte, die Konfiguration und die Parameter des Gerätes gelesen und verändert werden.

Des weiteren ist die Übertragung von Mess- und Stellwerten über das Netzwerk mittels IP-Protokoll (UDP/TCP) implementiert. Hier werden die Daten in definierter Form übertragen.

Die prinzipiellen Probleme netzwerkfähiger Geräte:

1. Wird das Gerät zum ersten mal verwendet, befindet sich das Gerät im Auslieferungszustand. Diese Netzwerk-Einstellungen stimmen meistens nicht mit der vorhandenen Infrastruktur überein. Das Gerät ist somit im Netzwerk nicht erreichbar.
2. Soll das Gerät in einer anderen Infrastruktur (Wartung, Austausch, usw.) betrieben werden, so ist dieses - wie unter Punkt 1 beschrieben - eventuell im Netzwerk nicht erreichbar.
3. Der Anwender hat dem Gerät eine IP-Adresse zugewiesen. Das Gerät soll zu einem späteren Zeitpunkt in einer veränderten Netzwerk-Infrastruktur betrieben werden (Wartung, Austausch, usw.). Eventuell ist die früher zugewiesene IP-Adresse in der neuen (veränderten) Umgebung nicht erreichbar.
4. Die einmal eingestellte IP-Adresse ist nicht mehr bekannt, bzw. falsch eingetragen worden. Das Gerät kann im Netzwerk nicht mehr angesprochen werden.

Aus diesen Gründen ist ein Verfahren definiert, um die IP-Adresse eines eWebSrv-Gerätes über das Netzwerk zu parametrieren, wenn die aktuelle IP-Adresse unbekannt ist (und dadurch der Zugriff auf das Gerät mit HTTP-Protokoll nicht möglich ist).

### 3.2.1 Konfiguration über die serielle Schnittstelle

Auf der ersten seriellen Schnittstelle läuft eine Linux Konsole mit „Root“-Rechten für Servicezwecke. An dieser Schnittstelle kann ein Terminal betrieben werden und entsprechende Einstellungen am System vorgenommen werden. Das Verfahren ist im Handbuch „Inbetriebnahme“ ausführlich beschrieben.

## 3.2.2 Konfiguration über die Ethernet-Schnittstelle

Da häufig der Zugriff auf die serielle Schnittstelle nicht möglich ist, gibt es bei **MHC** ein Verfahren zur Konfiguration über das Netzwerk. Dieses ermöglicht ein Ändern der Netzwerkeinstellungen, wobei die Einstellungen der Netzwerkkarte des lokalen Rechners nicht umgestellt werden müssen. Auch ist die Kenntnis der Netzwerkeinstellungen des Gerätes nicht nötig.

- Server-Dienst (UDP Broadcast)

### 3.2.2.1 Einstellen über Server-Dienst

Wird dieser Dienst auf dem Gerät gestartet, kann die Netzwerkschnittstelle des eWebSrv über das Netzwerk (UDP-Protokoll) eingestellt werden. Der zugehörige Server 'Netzwerk (UDP)' wird im Handbuch „System-Anwender“ beschrieben.

**Hinweis:** Der Server ist standardmäßig nicht aktiviert. Wahlweise kann der eWebSrv auch mit aktiviertem Server ausgeliefert werden.

### 3.3 Treiber

#### 3.3.1 NAND-Flash Hardware Treiber

Dieses Modul stellt dem darüber liegenden NAND Geräte-Treiber die Hardware-spezifischen Teile zur Verfügung und initialisiert alle nötigen Datenstrukturen. Auch die Partitionen des Flash-Bereiches werden hier festgelegt.

#### 3.3.2 Serielle Schnittstellen

Für den Zeitpunkt des Systemstarts (Bootvorgangs) stellt dieser Treiber einen Sonderfall dar. Die Parameter wie die Baudrate müssten aus dem PARAM-Bereich auslesen werden; die Informationen stehen zu diesem Zeitpunkt aber noch nicht zur Verfügung. Das wird dadurch gelöst, dass dieser Treiber die fixen Einstellungen des Bootloaders übernimmt.

#### 3.3.3 **MHC** Daten-Strukturen

CONFIG- und PARAM-Bereich sind von zentraler Bedeutung für das ganze System.

Aus diesem Grund werden CONFIG und PARAM über das virtuelle Dateisystem SYSFS (Kapitel 3.4) abgebildet und können hierüber gelesen und gegebenenfalls geschrieben werden. Jeder einzelne dieser Einträge kann von einem Anwenderprogramm wie ein reguläre Datei behandelt werden, d.h. mittels 'open – read/write – close' (C-Funktionen) benutzt werden.

Dieser Treiber, bezeichnet als **MHC**-Treiber, ist als ladbares Modul ausgelegt und wird beim Systemstart mit der Ausführung von '/etc/rc' (runcommand) initialisiert. Erst dann stehen die Einträge der Strukturen dem System zur Verfügung.

Durch diese Implementierung ist es leicht möglich, das System bei einer Änderung/Erweiterung der Strukturen ohne Update des Kernels zu aktualisieren.

#### 3.3.4 Datenpunkte

Wie auf CONFIG und PARAM sind auch auf die Daten-Bereich für die digitalen und analogen Datenpunkte keine direkten Zugriffe erlaubt. Zugriffe erfolgen nur über vom System zur Verfügung gestellten Geräte-Treiber mit den dazugehörigen Funktionen. Das stellt einerseits die Möglichkeit bereit unterschiedliche Rechte zu verteilen, andererseits ist sichergestellt, dass keine gleichzeitigen Zugriffe seitens mehrerer paralleler Prozesse erfolgen.

Die Daten-Definitionen finden sich ebenfalls im SYSFS wieder. Der Zugriff auf die Datenpunkte selbst erfolgt über Device-Deskriptoren im Verzeichnis '/dev/data'.

Die vollständige Implementierung befindet sich im gleichen Treiber wie für CONFIG und PARAM.

Das Aktualisierungs-Intervall für Datenpunkte ist festgelegt auf 10 ms.

### 3.3.4.1 Unit-Devices

Der Datenbereich eines Gerätes kann über ein Unit-Device mittels read/write Systemaufruf linear gelesen oder beschrieben werden. Der Device-Deskriptor befindet sich im Verzeichnis '/dev/data'. Der Name des Unit0 Devices kann mit einem SYSFS Aufruf ermittelt werden. Die SYSFS Datei lautet

```
'/sys/devices/platform/mkc/local_unit_name'
```

und ist aus dem Eintrag 'device-name' in der data-definition Struktur generiert. Angenommen der Befehl

```
cat /sys/devices/platform/mkc/local_unit_name
```

liefert 'ewebsrv', dann findet sich der device-Deskriptor für die Unit0 unter

```
'/dev/data/ewebsrv'
```

Es handelt sich um ein Character-Device.

Arbeitet der eWebSrv mit weiteren über das Netzwerk verbundenen Geräten (z.B. IONet) zusammen, müssen die Datenpunkte der externen Geräte im eWebSrv abgebildet werden. Für jedes dieser Geräte existiert eine weitere Unit (1...n) mit einem frei bestimmbar Namen.

Die Datensynchronisation mit den externen Geräten erfolgt immer, indem der komplette Datenblock des Gerätes transferiert wird. Das Unit-Device dient nur zum Datenaustausch von Geräten untereinander.

Aufrufe (C-Funktionen):

- `int open (const char *filename, int flags[, mode_t mode])`
- `int close (int filedes)`
- `ssize_t read (int filedes, void *buffer, size_t size)`
- `ssize_t write (int filedes, const void *buffer, size_t size)`
- `int ioctl (int filedes, int command, ...)`
- `int poll (struct pollfd *fds, nfds_t nfds, int timeout)`

Dies sind die Funktionsaufrufe aus Anwendersicht (d.h. User-Programme oder andere Treiber), nicht die Funktionen des Treibers. Funktionen des Treibers können nicht direkt aufgerufen werden.

Für eine generelle Beschreibung dieser Funktionen siehe: <http://www.gnu.org/software/libc/manual/>

**ioctl**-Aufrufe:

- `MKCDEF_IOCTL_ACCESS_ID`

Die Aufrufe im Einzelnen:

```
ergebnis = ioctl( int filedes, MKCDEF_IOCTL_ACCESS_ID, short value)
Setzt den read/write mode (data-definition oder data-value) und den Filepointer auf 0.
ergebnis: ok := 0, im Fehlerfall := <0
```

Die 'read/write' Befehle sind so implementiert als ob diese auf einem „streaming device“ arbeiten, d.h. ein 'seek' wird nicht unterstützt. Sie lesen/schreiben entweder den Definitionsblock (data-definition) oder den Datenblock (data-value) der Unit.

Mit dem ioctl-Befehl wird der zu transferierende Bereich gewählt, indem die entsprechende ID (z.B. 'value' = ID\_DATA\_DEFINITION) übergeben wird; gleichzeitig wird der interne Filepointer des Devices zu '0' gesetzt.

Um das „Handling“ zu vereinfachen, wurden „Inline“-Funktionen definiert. Diese sind im Aufruf analog den Standard 'read/write' Aufrufen:

- `ssize_t read0 (int filedes, void *buffer, size_t size)`
- `ssize_t write0 (int filedes, const void *buffer, size_t size)`
- `ssize_t read_ddef (int filedes, void *buffer, size_t size)`
- `ssize_t write_ddef (int filedes, const void *buffer, size_t size)`
- `ssize_t read_dval (int filedes, void *buffer, size_t size)`
- `ssize_t write_dval (int filedes, const void *buffer, size_t size)`

Alle diese Aufrufe setzen mittels 'ioctl' den Mode passend und den Filepointer auf '0'. 'read0/write0' werten den übergebenen Speicherbereich (buffer) aus, um den Mode zu setzen. Die 'read/write\_ddef' bzw. 'read/write\_dval' Aufrufe setzen den Mode (data-definition bzw. data-value) unabhängig vom Inhalt des übergebenen Speicherbereichs.

Beim ersten Lesen eines Devices ist die Größe des angeforderten Datenblocks unbekannt. Die allgemeine Vorgehensweise ist zuerst nur den Header (decl\_def\_hdr) der Struktur auslesen und im zweiten Schritt entweder die komplette Struktur oder nur den Rest. Hierfür ein Beispiel:

```
fd = open(„/dev/data/public“,O_RDONLY);
read_dval(fd, buffer, sizeof(decl_def_hdr));
rest = buffer->header.size_of - sizeof(decl_def_hdr);
read(fd, buffer + sizeof(decl_def_hdr), rest);
close(fd);
```

Ab der Firmware 2.0.1.1 unterstützt der Treiber auch asynchrone Benachrichtigungen. Wenn dieser Modus aktiviert ist wird ein Signal (SIGPOLL) gesendet, sobald Daten vom Device gelesen werden können. Für eine generelle Beschreibung dieser Funktion siehe: <http://www.gnu.org/software/libc/manual/>.

### 3.3.4.2 Application-Devices

Diese Devices stellen die Schnittstelle zu Datenpunkten für Anwendungsprogramme dar. Beim Start des Treibers werden Application-Devices mit ihren Deskriptoren in '/dev/data' erzeugt. Es existiert mindestens ein Application-Device namens „public“ (Public-Device) für alle Datenpunkte die keinem bestimmten Device zugeordnet sind. Weitere werden über Konfigurationsdateien im Verzeichnis '/etc/app.init/' erzeugt.

Aufrufe (C-Funktionen):

- `int open (const char *filename, int flags[, mode_t mode])`
- `int close (int filedes)`
- `ssize_t read (int filedes, void *buffer, size_t size)`
- `ssize_t write (int filedes, const void *buffer, size_t size)`
- `int ioctl (int filedes, int command, ...)`
- `int poll (struct pollfd *fds, nfds_t nfds, int timeout)`

Das sind die Funktionsaufrufe aus Anwendersicht (d.h. User-Programme oder andere Treiber), nicht die Funktionen des Treibers. Funktionen des Treibers können nicht direkt aufgerufen werden.

Für eine generelle Beschreibung dieser Funktionen siehe: <http://www.gnu.org/software/libc/manual/>

**ioctl**-Aufrufe:

- `MKCDEF_IOCTL_DATAREADMODE`
- `MKCDEF_IOCTL_ACCESS_ID`

Die Aufrufe im Einzelnen:

`ergebnis = ioctl( int filedes, MKCDEF_IOCTL_DATAREADMODE, int *mode )`  
 Bestimmt, ob die Datenpunkte eines Devices zyklisch gelesen werden (Inhalt von 'mode' = 0) oder nur beim read Aufruf (Inhalt von 'mode' = 1); ergebnis := filepointer oder im Fehlerfall := < 0

`ergebnis = ioctl( int filedes, MKCDEF_IOCTL_ACCESS_ID, short value)`  
 Analog dem 'ioctl' für Unit-Devices

Für die **'read/write'** Befehle gilt hier das Gleiche wie für die Unit-Devices (auch read0/write0 usw.)

Die lokalen Datenpunkte aller Devices werden im Speicherabbild zyklisch aktualisiert. Diese Automatik kann bei jedem Device abgeschaltet werden (`ioctl MKCDEF_IOCTL_DATAREADMODE`). Dann muss die Aktualisierung allerdings durch das Anwendungsprogramm erfolgen ( indem es die Daten liest).

Werden Daten geschrieben, dient das 'write'-Bit in der Datenstruktur dazu, die Fertigstellung des Schreibvorgangs anzuzeigen. Beim Schreiben in den Datenbereich werden die Daten im Speicherabbild abgelegt und der Hintergrundprozess zum Setzen der Ports gestartet. Damit ist der Funktionsaufruf abgeschlossen und kehrt zum Anwender zurück. Hat der Hintergrundprozess seine Arbeit getan, setzt dieser das 'write'-Bit in der Datenstruktur zurück. Der Anwender kann den Datenpunkt anfordern um den Eintrag ('write'-Bit) der Datenstruktur auszuwerten. Solange das 'write'-Bit gesetzt ist, wird ein Datenpunkt im Speicherabbild vom unabhängig laufenden Lese-Prozess nicht aktualisiert.

Ab der Firmware 2.0.1.1 unterstützt der Treiber auch asynchrone Benachrichtigungen. Wenn dieser Modus aktiviert ist wird ein Signal (SIGPOLL) gesendet, sobald Daten vom Device gelesen werden können. Für eine generelle Beschreibung dieser Funktion siehe: <http://www.gnu.org/software/libc/manual/>.

### 3.4 System-Schnittstelle SYSFS

Das virtuelle Dateisystem SYSFS dient dem Auslesen von Informationen und dem Setzen von Einstellungen im laufenden System. Die einzelnen Parameter/Einstellungen werden unter Linux Attribute genannt.

Das Dateisystem ist unter `/sys/` eingehängt und beinhaltet Verzeichnisse, welche die Organisation der Geräte übernehmen, die im System integriert sind.

Die an dieser Stelle relevanten Unterverzeichnisse sind:

- `/sys/bus/platform/drivers'`      Attribute für Treiber
- `/sys/devices/platform'`      Attribute für Geräte

Jedes Attribut wird repräsentiert durch eine ASCII Textdatei, die gelesen und/oder geschrieben werden kann. Im einfachsten Fall zeigt der Befehl 'cat' den Inhalt des Attributs, ein 'echo' auf die Datei ändert das Attribut, sofern es nicht read-only ist.

Aus einem Anwenderprogramm erfolgt der Zugriff auf die Attribute mittels 'open' – 'read/write' – 'close'. Auch hier gilt: der übergebene Wert ist immer ein String, wobei die maximale Länge des Strings eine Speicherseite/ „mem-page“ beträgt.

Die im Folgenden aufgelisteten Dateien und Verzeichnisse sind nicht vollständig. Alle zusätzlichen Einträge innerhalb des SYSFS werden nur intern benötigt.

#### 3.4.1 Treiber-Attribute

Im Verzeichnis `/sys/bus/platform/drivers'` befinden sich zwei Unterverzeichnisse

- `/sys/bus/platform/drivers/MKCFPGAdrv'` (siehe Kapitel 4.4.1)
- `/sys/bus/platform/drivers/MKCdefDriver'`

Über die folgenden Attribute können Devices zum Datenpunkt Treiber hinzugefügt werden.

<code>/sys/bus/platform/drivers/MKCdefDriver/</code>	
<code>version</code>	Versions String des Treibers (read-only)
<code>addunit</code>	Unit-Device hinzufügen (write-only)
<code>addapp</code>	Application-Device hinzufügen (write_only)

**Tabelle 1: SYSFS: Datenpunkt Treiber**

Die `addunit` und `addapp` Attribute bekommen einen formatierten String der folgenden Form übergeben:

```
echo device-name minor-nr privat default-file-size >addunit
echo device-name minor-nr #din #dot #dio #ain #aot >addapp
```

<code>device-name</code>	der Name mit dem das Device erzeugt wird.
<code>minor-nr</code>	der Index, mit dem das Device im Treiber identifiziert wird
<code>privat</code> (nur <code>addunit</code> )	'1': Ports des Gerätes, die nicht einer Applikation zugewiesen wurden, sind im Public-Device nicht sichtbar.
<code>default-file-size</code>	Größe einer binären Data-Definition-Datei (default data-definition files). Im Verzeichnis <code>/etc/default/</code> befinden sich entsprechende Dateien für die IONet-Geräte.
<code>#din, ..., #aot</code>	Anzahl Datenpunkte des entsprechenden Typs.

**Hinweis:** Diese Attribute sollten nicht isoliert verwendet werden, sei es von der Kommandozeile oder einem Programm. Beim Systemstart wird das Programm `gen_devs` ausgeführt, es erstellt automatisch die Unit- und Application-Devices des Systems. Das Programm und die Konfigurationsdateien werden im Handbuch „System-Anwender“ ausführlich erklärt.

Außerdem befinden sich in diesem Verzeichnis Links zu den zugehörigen Devices.

### 3.4.2 System-Attribute: CONFIG und PARAM

Im Verzeichnis '/sys/devices/platform' befinden sich mindestens 3 weitere Unterverzeichnisse. Im Verzeichnis 'mkc' befinden sich die Konfigurations- und Parameterdaten des Systems. Alle weiteren Unterverzeichnisse (neben mkc) beherbergen die Einträge der Data-Definition Bereiche für die Datenpunkte.

<i>/sys/devices/platform/mkc/</i>	
<i>local_unit_name</i>	Name der Unit0, d.h. 'device_name' aus der data_definition des eWebSrv
<i>firmware</i>	Versionsnummer des Gesamtimages (Linux+Treiber+Anwendungen) zur Compile-Zeit
<i>config</i>	Unterverzeichnis Konfigurations-Daten (CONFIG) - read-only !
<i>param</i>	Unterverzeichnis Parameter-Daten (PARAM)
<i>tcpcomm</i>	Unterverzeichnis für TCPCOMM-Server

***Tabelle 2: SYSFS: Hauptverzeichnis CONFIG/PARAM***

### 3.4.2.1 CONFIG Zweig eWebSrv

<i>/sys/devices/platform/mkc/config/</i>	
<i>id</i>	ID der CONFIG Datenstruktur (0x4100)
<i>rev</i>	Revision der CONFIG Datenstruktur
<i>eth_modes</i>	Whitespace getrennte Liste der möglichen Ethernet-Einstellungen (Übertragungsmode)
<i>mac</i>	MAC Adresse des eWebSrv
<i>length_dio</i>	Anzahl DIO Datenpunkte
<i>length_partition</i>	Anzahl der Partitionen im Flash
<i>length_serial</i>	Anzahl der seriellen Schnittstellen
<i>dio&lt;index&gt;/</i>	Unterverzeichnis eines DIO Datenpunktes (length_dio mal vorhanden)
<i>id</i>	ID der Datenstruktur des DP (0x1005)
<i>rev</i>	Revision der Datenstruktur des DP
<i>terminal</i>	Terminal Name des DP
<i>type</i>	Whitespace getrennte Liste der möglichen Typ-Einstellungen des DP
<i>remanent</i>	Whitespace getrennte Liste der möglichen Remanenz Zustände
<i>owner</i>	Whitespace getrennte Liste der möglichen Owner
<i>ext</i>	Unterverzeichnis für extension Boards (oder Trägerkarte)
<i>fpga</i>	Unterverzeichnis für das FPGA auf dem eWebSrv
<i>partition&lt;index&gt;/</i>	Unterverzeichnis für eine Partition
<i>name</i>	Name der Partition
<i>size</i>	Größe der Partition in Bytes (hex)
<i>start_addr</i>	Startadresse der Partition (hex)
<i>pcb/</i>	Unterverzeichnis für Hardware info des eWebSrv
<i>id</i>	ID der Datenstruktur (0x1000)
<i>rev</i>	Revision der Datenstruktur
<i>part_no</i>	Part Number (Seriennummer)
<i>name</i>	Name der Hardware (EWEBSRV)
<i>revision</i>	Platinenrevision
<i>var</i>	Platinenvariante
<i>mod</i>	Platinenmodifikation
<i>day</i>	Herstellungstag
<i>month</i>	Herstellungsmonat
<i>year</i>	Herstellungsjahr
<i>logic_rev</i>	Revision des FPGA auf dem eWebSrv
<i>serial&lt;index&gt;/</i>	Unterverzeichnis einer seriellen Schnittstelle (length_serial mal vorhanden)
<i>id</i>	ID der Datenstruktur (0x1008)
<i>rev</i>	Revision der Datenstruktur
<i>terminal</i>	Bezeichnung der Schnittstelle
<i>baud</i>	Whitespace getrennte Liste der möglichen Baudraten
<i>data</i>	Whitespace getrennte Liste der möglichen Datenbits
<i>stop</i>	Whitespace getrennte Liste der möglichen Stoppbits
<i>parity</i>	Whitespace getrennte Liste der möglichen Parity Einstellunge
<i>handshake</i>	Whitespace getrennte Liste der möglichen Handshake Einstellungen

**Table 3: SYSFS: CONFIG eWebSrv**

Die Zugriffe auf die Daten des CONFIG-Bereichs sind immer Zugriffe auf das CFM. Es sind ausschließlich Lese-Zugriffe gestattet.

### 3.4.2.2 PARAM Zweig eWebSrv

<i>/sys/devices/platform/mkc/param/</i>	
id	ID der Parameter Datenstruktur (0x4101)
rev	Revision der Parameter Datenstruktur
auth_login_ro	HTTP login Name : Anwender
auth_login_rw	HTTP login Name : Operator
auth_login_su	HTTP login Name : Administrator
auth_password_ro	HTTP login, Anwender-Passwort
auth_password_rw	HTTP login, Operator-Passwort
auth_password_su	HTTP login, Administrator-Passwort
eth_mode	Index in die Liste von eth_modes (CONFIG-Bereich)
eth_start	Zu startende Server. Bitmaske
eth_start_xxx	Maskierte Versionen von eth_start
ip	IP Adresse des eWebSrv
mask	Netmaske des eWebSrv
gateway	IP Adresse des Gateway
http_port	Port des HTTP Servers
tcp_data_enable	IP Maske des TCP Daten Servers
tcp_data_port	Port des TCP Daten Servers
tcp_data_timeout	Timeout des TCP Daten Servers
udp_data_enable	IP Maske des UDP Daten Servers
udp_data_port	Port des UDP Daten Servers
udp_data_timeout	Timeout des UDP Daten Servers
udp_data_ip	IP des UDP Daten Servers
udp_eth_config_enable	IP Maske des UDP Config Servers
udp_eth_config_port	Port des UDP Config Servers
ntp_ip	IP eines NTP Zeitservers zur Synchronisation der Systemzeit
sync	Read: 0 -> WC und CFM gleich / 1 -> WC und CFM unterschiedlich Write: WC Daten werden ins CFM kopiert.
<i>dio&lt;index&gt;/</i>	
id	ID der Datenstruktur des DP (0x2002)
rev	Revision der Datenstruktur des DP
remanent	Remanenz des Daten Ausgangs
shift	Shiftfaktor des Dateneingangs (Mittelwertbildung)
type	Typ des DP
owner	Owner Flag
<i>ext</i>	Unterverzeichnis für extension Boards (oder Trägerkarte)
<i>fpga</i>	Unterverzeichnis für das FPGA auf dem eWebSrv
<i>serial&lt;index&gt;/</i>	
id	ID der Datenstruktur (0x2005)
rev	Revision der Datenstruktur
baud	Baudrate (Index in die CONFIG-Liste)
data	Datenbits (Index in die CONFIG-Liste)
stop	Stoppbits (Index in die CONFIG-Liste)
parity	Parity Einstellung (Index in die CONFIG-Liste)
handshake	Handshake Einstellung (Index in die CONFIG-Liste)

**Tabelle 4: SYSFS: Parameter eWebSrv**

Die Zugriffe auf die Parameterdaten sind immer Zugriffe auf eine Working-Copy (WC) des CFM Bereiches. Diese WC wird beim Start des Treibers angelegt.

Am Inhalt der Datei 'sync' ist erkennbar, ob die WC ins CFM kopiert werden muss. "1" als Rückgabewert zeigt an, dass synchronisiert werden muss, d.h. WC und CFM sind unterschiedlich. Wird ein beliebiger Wert in die Datei 'sync' geschrieben, werden WC und CFM wieder synchronisiert.

### 3.4.3 Device-Attribute: Datenpunkte

Neben dem Verzeichnis mkc in '/sys/devices/platform' befinden sich mindestens zwei weitere Verzeichnisse.

Das erste Unterverzeichnis beinhaltet Daten des Unit-Device 0; der Name dieses Verzeichnisses leitet sich aus dem Eintrag 'device\_name' im Definitions-Bereich (data\_definition) ab und lautet z.B. 'eWebSrv'.

Der Name kann durch Auslesen der SYSFS-Datei '/sys/devices/platform/mkc/local\_unit\_name' ermittelt werden.

Das andere Verzeichnis beinhaltet Daten des Application-Device 'public'.

Wurden weitere Units oder Application-Devices erzeugt, sind hier die entsprechenden Verzeichnisse zu finden.

#### 3.4.3.1 Unit-Devices

Der grundsätzliche Aufbau aller Unit-Devices ist immer gleich. In Abhängigkeit von der jeweiligen Plattform und Funktionalität der Unit können noch weitere Einträge hinzukommen.

/sys/devices/platform/<Name>/	
major	Major Nr. des zugehörigen Device
minor	Minor Nr. des zugehörigen Device
size_ndef	Größe des data_definition Bereiches in Bytes
error	Error-Flags für Verbindung zu externen Geräten
din<index>/	Unterverzeichnis jeweils pro DIN Datenpunkt
...	weitere Unterverzeichnisse, <index> von 0 bis ...
dot<index>/	Unterverzeichnis jeweils pro DOT Datenpunkt
...	weitere Unterverzeichnisse, <index> von 0 bis ...
dio<index>/	Unterverzeichnis jeweils pro DIO Datenpunkt
...	weitere Unterverzeichnisse, <index> von 0 bis ...
ain<index>/	Unterverzeichnis jeweils pro AIN Datenpunkt
...	weitere Unterverzeichnisse, <index> von 0 bis ...
aot<index>/	Unterverzeichnis jeweils pro AOT Datenpunkt
...	weitere Unterverzeichnisse, <index> von 0 bis ...

**Tabelle 5: SYSFS: Attribute von Unit-Devices**

Auch der Aufbau der jeweiligen Datenpunkte ist immer gleich, hier am Beispiel eines AOT-Datenpunktes dargestellt.

aot<index>/	Unterverzeichnis eines AOT Datenpunktes
id	ID der Datenstruktur des DP
rev	Revision der Datenstruktur des DP
terminal	Terminal Name
addsignal	Signal-Handling: Hinzufügen eines Listen-Eintrages Eintrag/Eingabe : [PID, Signal, Event-Maske]
remsignal	Signal-Handling: Löschen eines Listen-Eintrages durch Eingabe [PID]

**Tabelle 6: SYSFS: Attribute von Datenpunkten (Unit-Devices)**

### 3.4.3.2 Application-Devices

Ein weiteres Verzeichnis namens 'public' beinhaltet die Definitionen des Application-Devices '/dev/data/public'.

<i>/sys/devices/platform/public/</i>	
major	Major Nr. des zugehörigen Device
minor	Minor Nr. des zugehörigen Device
length_din	Anzahl für alle verfügbarer DIN
length_dot	Anzahl für alle verfügbarer DOT
length_dio	Anzahl für alle verfügbarer DIO
length_ain	Anzahl für alle verfügbarer AIN
length_aot	Anzahl für alle verfügbarer AOT
port	DP zum Device hinzufügen; wird benutzt vom Programm gen_devs
size_ddef	Größe des data_definition Bereiches in Bytes
<i>din&lt;index&gt;</i>	Softlink auf das Verzeichnis in der Unit dieses DP
...	weitere Softlinks
<i>dot&lt;index&gt;</i>	Softlink auf das Verzeichnis in der Unit dieses DP
...	weitere Softlinks
<i>dio&lt;index&gt;</i>	Softlink auf das Verzeichnis in der Unit dieses DP
...	weitere Softlinks
<i>ain&lt;index&gt;</i>	Softlink auf das Verzeichnis in der Unit dieses DP
...	weitere Softlinks
<i>aot&lt;index&gt;</i>	Softlink auf das Verzeichnis in der Unit dieses DP
...	weitere Softlinks

**Table 7: SYSFS: Attribute Application-Devices**

Wenn eine Anwendung Datenpunkte für den exklusiven Gebrauch benötigt, sollten diese für ein privates Application-Device reserviert werden. (Siehe Handbuch System-Anwender)

### 3.4.3.3 Signal-Handling bei Datenpunkten

Damit eine Anwendung auf dem eWebSrv den Zustand bzw. eine Zustandsänderungen auf einem/mehreren Datenpunkten erkennen kann, wird diese im Polling-Verfahren arbeiten, d.h. zyklisch von dem entsprechenden Applikation-Device lesen. Eine weitere Methode auf Zustandsänderungen einzelner/mehrerer Datenpunkte reagieren zu können, ist das Auslösen von Signalen. Hierbei wird dem Programm mit der entsprechenden PID (Prozess-ID) in Abhängigkeit von einer Event-Maske ein definiertes Signal gesendet.

Beispiel:

```
echo "9999,SIGUSR1,SIG_MODE_RISING" >sys/devices/platform/public/dot0/addsignal
echo "9999,SIGUSR2,SIG_MODE_FALLING" >sys/devices/platform/public/dot0/addsignal
```

Der erste Aufruf bewirkt, dass bei jedem Zustandswechsel 0 → 1 von DOT 0 des Public-Devices das Signal SIGUSR1 an den Prozess 9999 gesendet wird. Der zweite Aufruf bewirkt, dass bei jedem Zustandswechsel 1 → 0 von DOT 0 des Public-Devices das Signal SIGUSR2 an den Prozess 9999 gesendet wird.

Das Attribut „remsignal“ dient dem Löschen von Signal-Einträgen, beispielsweise wenn ein Programm durch Neustart eine neue PID bekommt. Ein Aufruf an dieses Attribut löscht alle Einträge mit der angegebenen PID aus der Liste des Datenpunktes. Diese Verfahrensweise ist zwingend notwendig, damit an falsche/nicht vorhandene Prozesse keine Signale gesendet werden.

Beispiel:

```
echo "9999" >sys/devices/platform/public/dot0/remsignal
```

Löscht die Signal-Liste für diesen Prozess (PID) von DOT 0.

### 3.4.4 Anwendungs-Attribute

Im Handbuch „System-Anwender“ ist der TCPCOMM Server beschrieben. Dieser dient dazu, einem Host-System über das TCP-Netzwerk den Zugriff auf ein serielles Device zu ermöglichen.

#### 3.4.4.1 TCPCOMM Zweig eWebSrv

In diesem SYSFS-Zweig befinden sich die Parameter dieser Server. Sie werden von 'gen\_tcpcomm' während des Systemstarts aus den Einträgen von INI-Dateien (Verz. 'etc/tcpcomm.init/') dynamisch erzeugt.

<i>/sys/devices/platform/mkc/tcpcomm/</i>	
<code>addserver</code>	Einen TCPCOMM Server Eintrag erzeugen
<code>length_tcpcomm</code>	Anzahl der TCPCOMM Server
<i>tpc&lt;index&gt;/</i>	Unterverzeichnis eines TCPCOMM-Servers (length_tcpcomm mal vorhanden)
<code>ip</code>	IP Maske des TCPCOMM Servers
<code>pid</code>	Process-id des (laufenden) TCPCOMM Server-Prozesses
<code>port</code>	TCP-Port des Servers
<code>comm</code>	Device-Name des verbundenen seriellen Devices
<code>enable</code>	0 oder Name eines Ausgangs-Datenpunkt zur Steuerung der seriellen Gegenstelle; Optional Beispiel: dot0
<code>available</code>	0 oder Name eines Eingangs-Datenpunkt zur Erkennung der seriellen Gegenstelle; Optional Beispiel: din0
...	weitere Unterverzeichnisse für TCPCOMM-Server

**Tabelle 8: SYSFS: Parameter TCPCOMM-Server**

## 4 Betriebssystem (FPGA und Extension)

### 4.1 Allgemein

Auf dem eWebSrv befindet sich ein FPGA indem spezifische Erweiterungen implementiert wurden. Um diese nutzen zu können muss ein Geräte-Treiber vorhanden sein, der Anwendungsprogrammen diese Funktionalität bereitstellt. Zur Zeit sind zwei FPGA-Varianten definiert.

Der eWebSrv wird immer auf einer Trägerkarte betrieben. Diese Trägerkarte kann dem Gesamtsystem weitere Funktionalität hinzufügen. Diese Systemerweiterungen (Extension) werden, wie bei einem FPGA, auch wieder mittels eines modularen Treibers eingebunden.

### 4.2 Treiber für das eWebSrv FPGA

Die in einem FPGA befindlichen Systemerweiterungen werden mittels eines nachladbaren Moduls unterstützt. Im CONFIG-Bereich des eWebSrvs befinden sich Einträge, die das FPGA definieren. Der **mhc**-Treiber benutzt die ID dieses Headers um das zugehörigen Modul selbstständig nachzuladen.

Beispiel:

Die ID des FPGAs ist 4102 (ID\_CONFIG\_EWEBSRV\_FPGA\_IO48). Daraufhin sucht und lädt der **mhc**-Treiber das Modul mit dieser ID. Über einen in allen FPGA Erweiterungen gleichen Kernel-Funktionsaufruf (function-call) wird nun das Modul transparent in den **mhc**-Treiber eingebunden. Damit werden für die im FPGA 4102 befindlichen 48 DIOs zusätzliche Einträge im SYSFS (im Zweig fpga) eingefügt. Die Unit- und Application-Devices werden entsprechend um weitere Indizes erweitert.

Stellt ein FPGA eine spezifische Funktionalität zur Verfügung muss das Treibermodul die entsprechenden Schnittstellen dafür bereitstellen.

#### 4.2.1 Grundfunktionen (FPGA)

In jeder realisierten FPGA-Implementierung müssen zwei Grundfunktionen vorhanden sein. Die dazugehörigen Funktionsaufrufe sind im Kernel implementiert.

Die Grundfunktionen sind:

- Reset extension Board
- Reset Phy (Ethernet Transceiver)

Das Setzen/Lesen der zwei RESET Leitungen erfolgt über Attribute im SYSFS. Im Verzeichnis /usr/bin sind die Programme bzw. Shell-Skripte für diese Funktionen. Sie werden im Handbuch „System-Anwender“ erklärt.

#### 4.2.2 FPGA 4102 (IO48)

Das FPGA 4102 stellt dem System 48 weitere Datenpunkte des Typs DIO zur Verfügung. Die Aktualisierung erfolgt zusammen mit den Datenpunkten der CPU. Der Zugriff auf CONFIG/PARAM wird über das SYSFS abgebildet.

#### 4.2.3 FPGA 4104 (SERC3X)

Das FPGA 4104 beinhaltet 8 serielle Schnittstellen zu einem Texas DSP TMS320C3x. In diesem Treiber ist die Unterstützung (SYSFS) für den CONFIG-/PARAM-Bereich enthalten.

#### 4.2.4 Treiber für serielle DSP Schnittstellen (SERC3X)

Der Treiber für die im FPGA 4104 befindlichen seriellen Schnittstellen ist ein eigenständiges Modul und unterstützt 8 Devices (Standard Linux Character-Device):

- `/dev/mkcs3x0 .. /dev/mkcs3x7`

Die Schnittstellen verfügen jeweils über ein 32 \* 32Bit Sende-FIFO und ein 256 \* 32Bit Empfangs-FIFO. Der Datentransfer zum/vom FIFO erfolgt im Interrupt Betrieb. Die Daten aus dem Empfangs-FIFO werden in einem Lesebuffer (Kernel) abgelegt.

Außerdem erzeugt und bedient der Treiber alle für diese Schnittstellen benötigten SYSFS Einträge.

Aufrufe (C-Funktionen):

- `int open` (const char \*filename, int flags[, mode\_t mode])
- `int close` (int filedes)
- `ssize_t read` (int filedes, void \*buffer, size\_t size)
- `ssize_t write` (int filedes, const void \*buffer, size\_t size)
- `int ioctl` (int filedes, int command, ...)
- `int poll` (struct pollfd \*fds, nfds\_t nfds, int timeout)
- `int fsync` (int filedes)

Dieses sind die Funktionsaufrufe aus Anwendersicht (d.h. User-Programme oder andere Treiber), nicht die Funktionen des Treibers. Funktionen des Treibers können nicht direkt aufgerufen werden.

Für eine generelle Beschreibung dieser Funktionen siehe: <http://www.gnu.org/software/libc/manual/>

#### WICHTIG:

Wie oben erwähnt, ist das FIFO 32 Bit breit und es kann auch nur 32 Bit breit gelesen und beschrieben werden. 'read/write' Aufrufe des Betriebssystems sind byteweise orientiert. Das hat zur Konsequenz, das Sende-Daten idealerweise in 4 Byte - „Paketen“ organisiert sind (:= einem 32 Bit Schreibzugriff).

Beispiel:

Werden 5 Byte von einem Anwenderprogramm geschrieben, dann werden 4 Byte in einen Schreib-Puffer (und danach ins FIFO) geschrieben und der aufrufende Prozess wird darüber informiert (Rückgabewert des 'write' Funktionsaufrufs = 4). Der darauf folgenden Schreibaufruf des Anwender-Programms muss dann mit dem 5. Byte beginnen!

**open:**

Der Treiber unterstützt „blocking / non-blocking“ Zugriffe. Bei Aufruf werden die FIFOs geleert, die Speicher (Puffer) initialisiert, der Kanal im FPGA aktiviert und der Lese-Interrupt eingeschaltet.

**close:**

Führt zunächst einen 'flush' des 'write'-Puffers durch, deaktiviert dann die Interrupts und die Kanäle im FPGA.

**write:**

Sind vorherige Schreibzugriffe abgeschlossen, werden die Daten in einen Puffer kopiert und zum Aufrufer (Programm) zurückgekehrt. Ist der vorherige 'write' noch nicht abgeschlossen, kehrt der Aufruf entweder mit Fehlercode: -EAGAIN (non-blocking) zurück oder es wird gewartet bis der vorherige 'write' abgeschlossen ist (blocking) und danach der aktuelle Schreibzugriff weiter durchgeführt. Ist kein Fehler aufgetreten, kehrt die Funktion mit der Anzahl der kopierten Bytes zurück.

Muss eine Anwendung aus irgendeinem Grund darauf warten, dass die Daten physikalisch gesendet wurden, ist 'fsync' explizit aufzurufen.

Beim Aufruf wird im Treiber der Pufferspeicher für den kompletten Transfer oder bis zum Puffer-Limit reserviert. Sollen mehr Daten als diese maximale Anzahl gesendet werden, ist das Anwendungsprogramm dafür zuständig den Transfer in geeignete Datenblöcke zu unterteilen.

Zum Setzen der maximalen Größe wird ein 'ioctl' Aufruf implementiert. Standardeinstellung für das Puffer-Limit ist 3 KByte (ca. 2 Ethernet Frames).

Sind Daten im Pufferspeicher, wird der Sende-Interrupt des Kanals aktiviert. Der Interrupt wird aktiv wenn das FIFO fast leer ist. Dann wird das FIFO mit neuen Daten gefüllt.

Ist der Pufferspeicher leer, also der Transfer komplett, wird der Sende-Interrupt deaktiviert.

Fehlercodes:

-EAGAIN: vorheriger Schreibzugriff noch aktiv („non-blocking“ \_Zugriff)

-ENOMEM: kein Speicher mehr verfügbar

-EFAULT: Kopierfehler

**read:**

Bei einem Aufruf wird zunächst überprüft, ob Daten im Puffer vorliegen. Ist der Puffer leer, kehrt die Funktion unmittelbar mit Fehlercode: -EAGAIN zurück (non-blocking) oder wartet auf Daten (blocking). Sind Daten vorhanden, kehrt der Aufruf mit der vorhandenen bis zur maximal angeforderten Anzahl der Daten zurück. Werden keine oder zu wenige Daten aus dem Device abgeholt, kann der Puffer und das FIFO vollständig belegt sein (Handshake-Mode deaktiviert). Dies führt zu einem Überlauf des FIFOs bzw. Datenverlust, wenn weitere Daten empfangen werden müssen. Um den Zustand vom FIFO eines Devices festzustellen, gibt es Flags die vom Anwender gelesen werden können ('ioctl' Aufruf, siehe Tabelle 9).

Um einen Datenverlust durch einen FIFO-Überlauf zu verhindern, ist es möglich den Handshake-Mode zu aktivieren. Ist dieser Modus aktiv und das FIFO voll, werden keine weiteren Daten angenommen. Die Voraussetzung dafür ist, dass die Gegenstelle diesen Verfahren unterstützt. Genauer hierzu im Handbuch „Hardware“.

Fehlercodes:

-EAGAIN: Keine Daten verfügbar („non-blocking“ \_Zugriff)

-EFAULT: Kopierfehler

-EIO: FIFO voll

**poll:**

Der poll Aufruf dient dazu, ohne zu blockieren zu ermitteln, ob auf einen Kanal geschrieben oder gelesen werden kann. Dazu bekommt die Funktion in einem array übergeben, auf welchen Kanälen auf bestimmte Ereignisse gewartet werden soll.

**fsync:**

Der Aufruf kommt erst zum Anwender zurück, wenn der 'write' Puffer leer ist.

**ioctl:**

Es sind folgende ioctl-Aufrufe implementiert:

**MKCSERC3X\_IOCTL\_STATUS (RO)**

Bit	Name	Reset	Beschreibung
[12..0]	CNT_R	0	Anzahl Worte im Empfangspuffer
[13]	N_EMPTY_R	0	Empfangs FIFO leer
[14]	ALMOST_FULL_R	0	Empfangs FIFO fast voll
[15]	FULL_R	0	Empfangs FIFO voll
[28:16]	CNT_X	0	Anzahl Worte im Sendepuffer
[29]	EMPTY_X	1	Sende FIFO leer
[30]	ALMOST_EMPTY_X	1	Sende FIFO fast leer
[31]	N_FULL_X	1	Sende FIFO voll

**Tabelle 9: MKCSERC3X\_IOCTL\_STATUS**

Auslesen der FIFO Status-Bits (32 Bit); Rückgabe in argp.

**Hinweis:** Die Status-Bits mit voran gesetzten „N\_“ sind negiert.

**MKCSERC3X\_IOCTL\_BAUD (RW / 32 Bit)**

Lesen/Setzen der Baudrate in Bits/Sek. Übergabe der Baudrate in argp. Wird -1 übergeben, wird die aktuelle Baudrate in argp zurückgeliefert.

**MKCSERC3X\_IOCTL\_DELAY (RW / 16 Bit)**

Lesen/Setzen des Receive Interrupt Verzögerung in Takten der Schnittstelle. D.h. ist das FIFO noch nicht voll darf diese Anzahl Takte kein Transfer erfolgen bevor ein Interrupt ausgelöst wird. Übergabe des Delay in argp. Wird -1 übergeben, wird das aktuelle Delay in argp zurückgeliefert.

**MKCSERC3X\_IOCTL\_HANDSHAKE (RW / 32 Bit)**

Lesen/Setzen des Handshake-Modes (FIFO). Übergabe von [0:Aus oder 1:Ein] in argp. Wird -1 übergeben, wird der aktuelle Zustand in argp zurückgeliefert.

**MKCSERC3X\_IOCTL\_AVAILABLE (RO / 32 Bit)**

Lesen der Anzahl der Bytes im Lesepuffer. Wird in argp zurückgeliefert.

**MKCSERC3X\_IOCTL\_RBUFF\_SIZE (RW / 16 Bit)**

Lesen/Setzen der Lesepuffer-Größe in KB. Wird -1 übergeben, wird die aktuelle Größe in argp zurückgeliefert. Zur Ausführung dieses Kommandos wird sofort das FIFO geleert, der Lesepuffer gelöscht und neu angelegt. Das Kommando sollte deshalb nur unmittelbar nach einem open verwendet werden.

**MKCSERC3X\_IOCTL\_WBUFF\_MAX (RW / 16 Bit)**

Lesen/Setzen der maximale Größe des Schreibpuffers in KB. Wird -1 übergeben, wird die aktuelle Größe in argp zurückgeliefert. Der neue Wert ist ab dem nächsten write gültig.

**Fehlercodes:**

- ENOTTY: unbekanntes Kommando
- ENOMEM: kein Speicher mehr verfügbar (RBUFF\_SIZE)
- EFAULT: Kopierfehler

### 4.3 Treiber für die eWebSrv Extension

Der entsprechende Extension-Treiber wird vom **mhc**-Treiber gestartet. Im CONFIG-Bereich des eWebSrvs befinden sich Einträge, die die Extension (Erweiterung) definieren. Der Treiber benutzt die ID dieses Headers um den dazugehörigen Treiber selbstständig nachzuladen. Das trifft für alle Zugriffe auf Systemerweiterungen zu.

Beispiel:

Die ID der Extension ist 4106 (ID\_CONFIG\_EWEBSRV\_EXT\_EWEBTEST). Daraufhin sucht und lädt der Treiber das Modul mit dieser ID. Über einen in allen Erweiterungen gleichen Kernel-Funktionsaufruf (function-call) wird nun das Modul transparent in den **mhc**-Treiber eingebunden. Für die Extension werden zusätzliche Einträge im SYSFS (im Zweig 'ext') eingefügt und gegebenenfalls weitere Unit- und Application-Devices generiert.

Stellt eine Trägerkarte weitere Funktionalität zur Verfügung, muss das Treibermodul die entsprechenden Schnittstellen bereitstellen.

#### 4.3.1 EXT 4106 (eWebTest)

Bei der eWebTest handelt es sich um eine Trägerkarte, die als Test und Entwicklungssystem für den eWebSrv dient. Im SYSFS stellt der Treiber den CONFIG/PARAM-Bereich bereit, beispielsweise für die PCB Daten, die definierten Datenpunkte, etc. .

### 4.4 System-Schnittstelle SYSFS

Sowohl für die FPGAs 4102 und 4104 als auch die Extension 4106 werden spezifische Attribute im SYSFS bereit gestellt bzw. für Zugriffe auf Einstellungen benötigt. In den folgenden Unterkapiteln werden diese Attribute beschrieben.

#### 4.4.1 Grundfunktionen (FPGA)

<code>/sys/bus/platform/drivers/MKCFPGAdrv/</code>	
version	Version String des Treibers
id	ID des FPGA
rev	Revision des FPGA
rstout	RSTOUT Pin des eWebSrv Setzen/Lesen
rstphy	Reset PHY Setzen/Lesen

*Tabelle 10: SYSFS: FPGA allgemein*

Die Signale sind low-aktiv. Schreiben einer "1" aktiviert den betreffenden Ausgang, d.h. das Signal wird physikalisch '0'.

**Hinweis:** Diese Attribute sollten nicht isoliert verwendet werden, sei es von der Kommandozeile oder einem Programm. Beim Systemstart werden benötigte Programme (z.B. **rst\_out.sh**) automatisch ausgeführt. Diese Programme werden im Handbuch „System-Anwender“ ausführlich erklärt.

## 4.4.2 FPGA 4102 (IO48)

### 4.4.2.1 CONFIG Zweig FPGA 4102

Die Konfiguration der Datenpunkte im FPGA 4102.

<code>/sys/devices/platform/mkc/config/fpga/</code>	
<code>id</code>	ID der Struktur (0x4102)
<code>rev</code>	Revision der Struktur
<code>length_dio</code>	Anzahl DIO Datenpunkte des eWebSrv FPGA
<code>dio&lt;index&gt;/</code>	Unterverzeichnis eines DIO Datenpunktes (length_dio mal vorhanden)
<code>id</code>	ID der Datenstruktur des DP (0x1005)
<code>rev</code>	Revision der Datenstruktur des DP
<code>terminal</code>	Terminal Name des DP
<code>type</code>	Whitespace getrennte Liste der möglichen Typ-Einstellungen des DP
<code>remanent</code>	Whitespace getrennte Liste der möglichen Remanenz Zustände
<code>owner</code>	Whitespace getrennte Liste der möglichen Owner
...	weitere Unterverzeichnisse für DP

**Tabelle 11: SYSFS: CONFIG FPGA IO48**

### 4.4.2.2 PARAM Zweig FPGA 4102

Die Parameter der Datenpunkte im FPGA 4102.

<code>/sys/devices/platform/mkc/param/fpga/</code>	
<code>id</code>	ID der Parameter Datenstruktur (0x4103)
<code>rev</code>	Revision der Parameter Datenstruktur
<code>dio&lt;index&gt;/</code>	Unterverzeichnis eines DIO Datenpunktes
<code>id</code>	ID der Datenstruktur des DP (0x2002)
<code>rev</code>	Revision der Datenstruktur des DP
<code>remanent</code>	Remanenz des Daten Ausgangs
<code>shift</code>	Shiftfaktor des Dateneingangs (Mittelwertbildung)
<code>type</code>	Typ des DP
<code>owner</code>	Owner Flag
...	weitere Unterverzeichnisse für DP

**Tabelle 12: SYSFS: Parameter FPGA IO48**

### 4.4.3 FPGA 4104 (SERC3X)

#### 4.4.3.1 CONFIG Zweig FPGA 4104

Die Konfiguration der 8 serielle Schnittstellen zu einem Texas DSP im FPGA 4104 (SERC3X).

/sys/devices/platform/mkc/config/fpga/	
id	ID der Struktur (0x4104)
rev	Revision der Struktur
length_serc3x	Anzahl der DSP Schnittstellen des eWebSrv FPGA
serc3x<index>/	Unterverzeichnis einer DSP Schnittstelle (length_serc3x mal vorhanden)
id	ID der Datenstruktur (0x1009)
rev	Revision der Datenstruktur
terminal	Name der Schnittstelle
baud	Whitespace getrennte Liste der möglichen Baudraten
rcv_delay	Whitespace getrennte Liste der möglichen Receive Delay Werte
handshake	Whitespace getrennte Liste der möglichen Handshake Werte
...	weitere Unterverzeichnisse für DSP Schnittstellen

*Tabelle 13: SYSFS: CONFIG FPGA SERC3X*

#### 4.4.3.2 PARAM Zweig FPGA 4104

Die Parameter der SERC3X Schnittstellen.

/sys/devices/platform/mkc/param/fpga/	
id	ID der Parameter Datenstruktur (0x4105)
rev	Revision der Parameter Datenstruktur
serc3x<index>	Unterverzeichnis einer DSP Schnittstelle
id	ID der Datenstruktur (0x2006)
rev	Revision der Datenstruktur
baud	Baudrate (Index in die CONFIG-Liste)
rcv_delay	Receive Delay (Index in die CONFIG-Liste)
handshake	Handshake (Index in die CONFIG-Liste)
...	weitere Unterverzeichnisse für DSP Schnittstellen

*Tabelle 14: SYSFS: Parameter FPGA SERC3X*

#### 4.4.4 EXT 4106 (eWebTest)

##### 4.4.4.1 CONFIG Zweig EXT 4106

Beinhaltet die Daten der eWebTest Platine.

/sys/devices/platform/mkc/config/ext/	
id	ID der Struktur (0x4106)
rev	Revision der Struktur
length_ain	Anzahl AIN Datenpunkte
length_aot	Anzahl AOT Datenpunkte
length_din	Anzahl DIN Datenpunkte
<b>pcb/</b>	<b>Unterverzeichnis für Hardware-Informationen der eWebTest</b>
id	ID der Datenstruktur (0x1000)
rev	Revision der Datenstruktur
part_no	Part Number (Seriennummer)
name	Name der Hardware (EWEBTEST)
revision	Platinenrevision
var	Platinenvariante
mod	Platinenmodifikation
logic_rev	Revision des FPGA auf der Trägerkarte, in diesem Fall 'none'
...	Weitere Einträge
<b>ain&lt;index&gt;/</b>	<b>Unterverzeichnis eines AIN Datenpunktes</b>
id	ID der Datenstruktur
rev	Revision der Datenstruktur
terminal	Name des Datenpunktes
owner	Whitespace getrennte Liste der möglichen Owner
length_range	Anzahl der möglichen Bereiche (ranges)
range<index>/	Mess-Bereich eines AIN Datenpunktes
id	ID der Datenstruktur
rev	Revision der Datenstruktur
...	Weitere Einträge für Korrektur und Umrechnung
<b>aot&lt;index&gt;/</b>	<b>Unterverzeichnis eines AOT Datenpunktes</b>
id	ID der Datenstruktur
rev	Revision der Datenstruktur
terminal	Name des Datenpunktes
owner	Whitespace getrennte Liste der möglichen Owner
length_range	Anzahl der möglichen Bereiche (ranges)
range<index>/	Stellwert-Bereich eines AOT Datenpunktes
id	ID der Datenstruktur
rev	Revision der Datenstruktur
...	Weitere Einträge für Korrektur und Umrechnung
<b>din&lt;index&gt;/</b>	<b>Unterverzeichnis eines DIN Datenpunktes</b>
id	ID der Datenstruktur
rev	Revision der Datenstruktur
terminal	Name des Datenpunktes
owner	Whitespace getrennte Liste der möglichen Owner

**Tabelle 15: SYSFS: CONFIG Extension eWebTest**

#### 4.4.4.2 PARAM Zweig EXT 4106

Die Parameter der Datenpunkte auf der eWebTest Platine.

<i>/sys/devices/platform/mkc/param/ext/</i>	
<i>id</i>	ID der Parameter Datenstruktur (0x4107)
<i>rev</i>	Revision der Parameter Datenstruktur
<i>ain&lt;index&gt;/</i>	Unterverzeichnis eines AIN Datenpunktes
<i>id</i>	ID der Datenstruktur
<i>rev</i>	Revision der Datenstruktur
<i>hysteresis</i>	Hysteresis-Wert des Datenpunktes
<i>range</i>	Eingestellter Mess-Bereich
<i>shift</i>	Shiftfaktor des Dateneingangs (Mittelwertbildung)
<i>owner</i>	Owner Flag
<i>aot&lt;index&gt;/</i>	Unterverzeichnis eines AOT Datenpunktes
<i>id</i>	ID der Datenstruktur
<i>rev</i>	Revision der Datenstruktur
<i>remanent</i>	Remanenz des Daten Ausgangs
<i>range</i>	Eingestellter Stellwerte-Bereich
<i>owner</i>	Owner Flag
<i>din&lt;index&gt;/</i>	Unterverzeichnis eines DIN Datenpunktes
<i>id</i>	ID der Datenstruktur
<i>rev</i>	Revision der Datenstruktur
<i>shift</i>	Shiftfaktor des Dateneingangs (Mittelwertbildung)
<i>owner</i>	Owner Flag

**Tabelle 16: SYSFS: Parameter Extension eWebTest**

## 5 Entwicklungsumgebung

In diesem Kapitel wird der Aufbau und die Verwendung der uClinux Entwicklungsumgebung für den eWebSrv beschrieben. Zum Erstellen der Software für den eWebSrv sind zunächst 3 Dinge erforderlich:

- PC mit Linux Installation inkl. grafischer Umgebung (z.B. KDE), Entwicklungsprogrammen (z.B. 'gcc') und Standard-Tools (z.B. 'make').
- uClinux Distribution mit möglichst aktuellen „Patches“. **MHC** arbeitet mit uClinux-dist-20100628, Download-Adresse: <http://www.uclinux.org/pub/uClinux/dist/>
- Eine sogenannte „Toolchain“. Diese stellt die komplette „Build“-Umgebung zur Verfügung (Cross-Compiler, Linker, etc.), Download-Adresse: <http://www.uclinux.org/pub/uClinux/m68k-elf-tools/>. (Zum Installieren der Toolchain muss das Installations-Skript ausgeführt werden.)

### 5.1 Allgemein

Der Source-Code befindet sich in einer Datei der Form: 'ewebdrv.uclinux.x.x.x.x.tar.bz2' im Verzeichnis 'source' des SDKs. Die HTML-Seiten befinden sich ebenfalls in dieser Datei.

Diese Datei muss in ein Arbeitsverzeichnis kopiert und entpackt werden.

Beispiel: `tar -xjf ewebdrv.uclinux.2.0.1.1.tar.bz2'`

In diesem Verzeichnis befinden sich nun die Hauptverzeichnisse 'uclinux' und 'html'. Das Verzeichnis '**uclinux**' wird in der weiteren Beschreibung '**AV**' genannt.

Das Shell-Skript 'm68k-uclinux-tools-20xxxxxx.sh' (im Verzeichnis toolchain) muss in ein temporäres Verzeichnis kopiert und ausgeführt werden.

Damit ist die „Build“-Umgebung erzeugt.

#### 5.1.1 Aktualisierung des Systems

Da der eWebSrv vollständig konfiguriert und somit lauffähig ausgeliefert wird, können einzelne Dateien per FTP auf den eWebSrv geladen werden.

**Warnung:** Bitte beachten Sie das einige systemnahe Dateien/Programme (z.B. busybox) in einem laufendem System nicht ersetzt werden können. Sollte dies geschehen ist das System teilweise oder sogar vollständig unbenutzbar.

## 5.2 Systemerstellung

In diesem Kapitel werden alle Schritte erklärt, um ein komplettes eWebSrv-System zu erstellen.

### 5.2.1 Konfiguration

Von der Kommandozeile im AV aufrufen:

- **make xconfig**  
Menüpunkt „**Vendor/Product Selection**“ anwählen  
Gewünschten Hersteller (vendor) **MKC** und Gerät (System) einstellen.  
Tool mit „**Save and Exit**“ verlassen.
- **make xconfig**  
Menüpunkt „**Kernel/Library/Defaults Selection**“ anwählen und gewünschte Einstellungen mit 'y' aktivieren.  
Mit „**Main Menu**“ → „**Save and Exit**“ dieses Modus verlassen. Das gewünschte Einstellungs-Menü wird angezeigt. Die gewünschten Parameter des Kernel, die Bibliotheken und Anwendungen einstellen.
- **make vendor\_flashcfg**  
Die **mkc**-Config/Parameter Header-Dateien befinden sich zentral in einem Verzeichnis des Vendor-Zweigs (**mkc**). Hier werden Plattform-spezifische Softlinks in verschiedenen Verzeichnis erzeugt.

Der Quelltext für alle Systeme befindet sich in einem Verzeichnisbaum. Die Parameter eines Gesamtsystems befinden sich in hunderten von Konfigurationsdateien und in dem Zweig 'AV/vendors'.

In dem Zweig 'AV/vendors/MKC' befindet sich für jedes **mkc**-System ein eigenes Verzeichnis. In diesen Verzeichnissen befinden sich Standard-Konfigurationen (automatisch erzeugt), Start-Dateien des Zielsystems, sowie ein Makefile zur Erstellung der Umgebung des Zielsystems.

Die Auswahl der Parameter erfolgt über eine grafische Oberfläche, die mittels 'make xconfig' gestartet wird.

Zur Zeit sind folgende **mkc**-Systeme definiert, der Plattform-Name stellt gleichzeitig auch den Verzeichnisnamen dar:

- **EWEBTEST** eWebSrv mit FPGA IO48 auf der eWebTest Trägerkarte
- **EWEBIO48** eWebSrv mit FPGA IO48
- **EWEBSERC3X** eWebSrv mit FPGA SERC3X
- **EWEBINIT** RAM basiertes Test-System
- **EWEBUPDATE** RAM basiertes Installations-System

In der Makefile Umgebung findet sich dieser Name in der Variablen \$(CONFIG\_PRODUCT) wieder.

### 5.2.2 „Build“-Prozess

Von der Kommandozeile im AV aufrufen:

- **make**  
Das komplette System wird erstellt. Unter 'AV/romfs' wird ein Verzeichnis für das Zielsystem erstellt. Der Kernel (linux.bin) befindet sich in 'AV/images'.

## 5.3 Installation/Wiederherstellung

### 5.3.1 Vorbereitungen

Im Verzeichnis 'AV/vendors/MKC/' befindet sich eine Datei namens 'mkc\_releases'. In dieser Datei sind für jedes Produkt die 4 Release Nummern abgelegt.

Beim Aufruf von **'make; make release'** wird unter '/srv/ftp' ein Verzeichnisbaum aus dem Namen des Produktes und dem Release String angelegt. Also z.B.: EWEBTEST/2.0.1.1/ mit den Verzeichnissen kernel und system.

In das 'kernel' Verzeichnis wird linux.bin kopiert, in das 'system' Verzeichnis der Inhalt von AV/romfs. Von allen Dateien wird nun der md5-hash generiert, der zusammen mit den Dateien selbst zu 2 gepackten Dateien komprimiert wird, von den wieder der md5-hash generiert wird.

Diese Dateien werden zusammen mit einer Datei für die HTML-Seiten unter '/srv/ftp' abgelegt.

Um beim obigen Beispiel zu bleiben:

- EWEBTEST.2.0.1.1.kernel.tar.gz
- EWEBTEST.2.0.1.1.kernel.tar.gz.md5
- EWEBTEST.2.0.1.1.sys.tar.gz
- EWEBTEST.2.0.1.1.sys.tar.gz.md5
- EWEBTEST.2.0.1.1.html.tar

Im dem Verzeichnis 'system\_rescue' des SDK von **MKC** befindet sich der Original-Datensatz mit dem der eWebSrv ausgeliefert wurde inklusive Tools für eine Installation/Wiederherstellung mittels einem Windows XP Arbeitsrechner.

Kopieren Sie das Verzeichnis 'system\_rescue' auf Ihren Arbeitsrechner und ersetzen Sie ggf. die 5 Installationsdateien durch Ihre selbst erstellten Dateien. Passen Sie in den Dateien **init\_nand.bat** und **load\_nand.bat** die IP-Zieladresse des eWebSrvs mit einem Editor an. Starten Sie einen FTP-Server.

### 5.3.2 Prozedur

An den eWebSrv eine Konsole anschließen und eWebSrv starten.

Ist das installierte Kernel-Image nicht mehr ladbar meldet sich automatisch der Bootloader (dBUG). Falls sich nach dem Einschalten noch das Linux meldet, muss ein Reset des Systems ausgelöst werden, eine beliebige Taste (z.B. die Leertaste) gedrückt und solange gehalten werden bis sich der Bootloader mit dem Prompt: **'dBUG>'** meldet.

Eingabe/Aktion	auf Target (Konsole) auf eWebSrv
1.	Das Kommando <b>'node s'</b> eingeben. dbug lädt das Service-System (image.bin).
2.	Mit dem Kommando <b>'boot'</b> wird dieses Image gebootet.

Nach dem Boot-Vorgang läuft jetzt im DRAM ein komplettes Linus System und es kann mit der Installation bzw. Wiederherstellung des Systems begonnen werden. Die IP-Adresse ist defaultmäßig 192.168.15.90 und kann mit den bekannten Linux-Tools ggf. geändert werden.

Eingabe/Aktion	auf Arbeitsrechner	auf Target (Konsole) auf eWebSrv
1.	Aufruf von <b>init_nand.bat</b>	
2.		Wechseln Sie in das Verzeichnis tmp: <b>cd /tmp</b>
3.		Aufruf von <b>partition_nand</b>
4.		Aufruf von <b>format_nand 2 3</b>
5.	Aufruf von <b>load_nand.bat</b>	
6.		Aufruf von <b>update_nand sys=EWEBTEST rev=2.0.1.1</b>
7.		Aufruf von <b>reboot</b>
		Jetzt bootet das <u>neue</u> Linux aus dem NAND-FLASH

## 6 Anwendungs-Entwicklung

In diesem Kapitel werden alle Schritte erklärt, um eine Anwendung für ein eWebSrv-System zu erstellen und zu installieren.

### 6.1 Grundlagen

Für das Entwickeln von Anwendungen werden ausreichende Kenntnisse über das Programmieren unter Linux (z.B. make, gcc, usw. ) und der Programmier-Sprache C vorausgesetzt. Für das Erstellen von Quell-Dateien kann ein beliebiger Editor benutzt werden. Alle relevanten Dateien von MKC sind in dem SDK zu finden, siehe folgende Tabelle.

<i>AV/linux-2.6.x/drivers/mkc/</i>	
...	Hier befinden sich die Quell-Dateien für die MKC-Treiber.
<i>AV/users/mkc/</i>	
...	In den weiteren Unterverzeichnissen befinden sich die Quell-Dateien für die Anwendungen und Tools von MKC.
<i>AV/include/mkc/</i>	
...	Hier befinden sich die Header-Dateien von MKC.

*Tabelle 17: MKC-Verzeichnisse*

### 6.2 Kompletter Entwicklungs-Zyklus

Das Verzeichnis für Ihre Quell-Dateien befindet sich unter 'AV/users/**oem**'. Damit wird sichergestellt, dass alle gewünschten Programme automatisch erstellt werden können.

#### 6.2.1 Vorbereitungen

Von der Kommandozeile im AV aufrufen:

- **make xconfig**  
 Menüpunkt „**Kernel/Library/Defaults Selection**“ anwählen und „**Customize Vendor/User Settings**“ mit 'y' aktivieren. Mit „**Main Menu**“ → „**Save and Exit**“ dieses Modus verlassen. Das gewünschte Einstellungs-Menü wird angezeigt.  
 Unter dem Menüpunkt „**Miscellaneous Applications**“ anwählen und „**OEM programs**“ mit 'y' aktivieren.  
 Mit „**Main Menu**“ → „**Save and Exit**“ dieses Modus verlassen. Die Entwicklungsumgebung ist jetzt so Konfiguriert, dass Programme unter dem Verzeichnis oem im „Build“-Prozess mit erstellt werden.

In das Top-Level „Makefile“ unter „oem“ wird nur der Eintrag 'dirs=' um den Verzeichnis-Namen der Anwendungs-Quellen erweitert. Das Anwendungs-Makefile wird im folgenden Abschnitt beschrieben.

## 6.2.2 Erstellen einer neuen Anwendung

Ein Beispiel Makefile für Anwendungen :

```
EXEC = Anwendungs-Name
OBJS = Anwendungs-Name.o

all: $(EXEC)

$(EXEC): $(OBJS)
$(CC) $(LDFLAGS) -o $@ $(OBJS) $(LDLIBS)

clean:
-rm -f $(EXEC) *.elf *.gdb *.o

romfs:
$(ROMFSINST) /bin/$(EXEC)

%.o: %.c
$(CC) -c $(CFLAGS) -o $@ $<
```

*Abbildung 4: Makefile*

Die Anwendung wird analog wie im Kapitel 5.2.2 „Build“-Prozess beschrieben durch den Aufruf „**make**“ generiert.

Alternativ kann durch den Aufruf „**make user/oem\_only**“ die Bearbeitungszeit verkürzt werden, wenn nur die Anwendung neu erzeugt werden soll (muss). Dabei werden die Anwendungen in diesem Verzeichnis neu übersetzt, aber **nicht** in das „romfs“ kopiert.

## 6.2.3 Installation der neuen Anwendung

Abhängig davon welches Zielverzeichnis (Ziel: **romfs: ff.**) im Makefile angegeben wurde, befindet sich die Anwendung durch das Erstellen automatisch im richtigen Verzeichnis (File-System des Geräts), wenn das System vollständig neu erzeugt wird.

## 7 ANHANG

### 7.1 Definitionen

#### 7.1.1 Allgemein

Die allgemeinen Software-Definitionen für **mhc** befinden sich in den Dokumenten „MkcDefUserLib“ und „MkcDefCommonLib“.

### 7.2 Beschreibung des SDK

Verzeichnis:	Beschreibung
<b>distribution</b>	
binaries	Kernel-, System- und HTML-Dateien mit denen der eWebSrv ausgeliefert wurde.
oem	Verzeichnis für OEM-Dateien
source	Komprimierte uClinux-Distribution mit Modifikationen von <b>mhc</b>
toolchain	Die Utilities zum Erstellen der Images
<b>Dokumentation</b>	Die Dokumentation von <b>mhc</b>
<b>HTML-Seiten</b>	Die HTML-Dateien von <b>mhc</b>
<b>RAM-System</b>	Das Linux Download Image mit einem komplett im RAM laufenden System.(*,**)

*Tabelle 18: SDK-Inhalt*

(\*) ein frei verfügbarer TFTP-Server kann z.B. von <http://tftp32.jounin.net/> geladen werden.

(\*\*) ggf. muss das Image noch umbenannt werden in 'image.bin'

### 7.3 Weiterführende Informationen

#### 7.3.1 MKC-Handbücher

Für ein WebSrv-System existieren die Handbücher Hardware, Benutzer, System-Anwender und System-Programmierer (dieses Handbuch).

#### 7.3.2 Online-Material

Aktuelle Software und Dokumentation finden Sie auf der Produkt-Seite des eWebSrv im Internet :

<http://www.mkc-gmbh.de/produkte/ewebsrv.php>

### 7.4 Rechtliche Bestimmungen

#### 7.4.1 Lizenzierung

Soweit nicht explizit anders angegeben unterliegen alle von **mhc** entwickelten und veröffentlichten Programme, Treiber und Skripte für den eWebSrv (Modul / Systeme) der GNU Public Licence in Version 2 (GPLv2) oder höher .