



**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-Anwender**

**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>GRUNDLAGEN .....</b>	<b>10</b>
2.1	Betriebssystem .....	10
2.2	System-Zugriff .....	10
2.2.1	TELNET aktivieren.....	11
2.2.2	Verbindungstest .....	12
<b>3</b>	<b>MESS- UND STELLWERTE.....</b>	<b>13</b>
3.1	Sensoren, Eingänge.....	13
3.1.1	Filterung der digitalen Eingänge.....	14
3.1.2	Filterung der analogen Eingänge.....	16
3.1.3	Hysterese der analogen Eingänge.....	18
3.2	Aktoren, Ausgänge.....	18
3.2.1	Remanenz der Ausgänge.....	18
3.3	Daten-Organisation.....	19
<b>4</b>	<b>ANWENDUNGEN.....</b>	<b>23</b>
4.1	Vorbemerkungen zum SYSFS.....	23
4.2	HTTP-Server.....	23
4.2.1	Realisierung.....	23
4.2.2	Funktionsaufrufe (SSI, CGI).....	24
4.2.2.1	get_data.cgi.....	24
4.2.2.2	get_data_value.cgi.....	25
4.2.2.3	get_data_definition.cgi.....	26
4.2.2.4	set_data.cgi.....	28
4.3	UDP-Config-Server.....	29
4.3.1	Berechtigungen.....	29
4.3.2	Server Start.....	29
4.4	Daten-Server (TCP).....	30
4.4.1	Berechtigungen.....	30
4.4.2	Server Start.....	30
4.5	Daten-Server (UDP).....	31
4.5.1	Berechtigungen.....	31
4.5.2	Server Start.....	31
4.6	TCPCOMM Server.....	32
4.6.1	Berechtigungen.....	32
4.6.2	Server Start.....	32
4.6.3	Init-Skript init_tcpcomm.sh.....	32
4.6.4	Start-Skript start_tcpcomm.sh.....	33
4.6.5	Programm gen_tcpcomm.....	33
4.7	Daten-Client (TCP).....	34
4.7.1	Client Start.....	34
4.8	Daten-Client (UDP).....	34
4.8.1	Client Start.....	34
4.9	FTP-Server.....	35
4.10	TELNET-Server.....	35
4.11	NTP-Client.....	35

<b>5</b>	<b>SYSTEM-TOOLS.....</b>	<b>36</b>
5.1	Mess- und Stellwerte.....	36
5.1.1	get_data.....	36
5.1.2	get_data_dv.....	37
5.1.3	set_data.....	38
5.2	Konfiguration und Parameter.....	39
5.2.1	set_param, set_param_4102, set_param_4104.....	39
5.2.2	ssec3x.....	40
5.2.3	fpga_id.sh, fpga_rev.sh.....	40
5.2.4	rstout, rstout.sh.....	41
5.2.5	rstphy, rstphy.sh.....	41
5.3	System-Initialisierung.....	42
5.3.1	Systemstart eWebTest.....	43
5.3.2	gen_devs.....	44
5.3.2.1	Aufbau der Konfigurationsdateien.....	45
<b>6</b>	<b>ANHANG.....</b>	<b>48</b>
6.1	Definitionen.....	48
6.1.1	Allgemein.....	48
6.2	Auslieferungszustand.....	48
6.3	Rechtliche Bestimmungen.....	48
6.3.1	Lizenzierung.....	48

## Liste der Abbildungen

Abbildung 1: TELNET aktivieren.....	11
Abbildung 2: Verbindungstest.....	12
Abbildung 3: Blockschaltbild Sensoren.....	13
Abbildung 4: Formel arithmetische Mittelwertberechnung (DIN).....	14
Abbildung 5: DIN (Filter = 1, keine Filterung des Eingangssignals).....	14
Abbildung 6: DIN ( Filter = 3, Filterung der kurzen Störungen).....	15
Abbildung 7: DIN ( Filter = 4, Filterung aller Störungen).....	15
Abbildung 8: Formel arithmetische Mittelwertberechnung (AIN).....	16
Abbildung 9: AIN (Eingangssignal-Sprung von 0% auf 100%, Filter = 1, Filterung des Eingangssignals).....	16
Abbildung 10: AIN (Eingangssignal-Sprung von 0% auf 100%, Filter = 4, Filterung des Eingangssignals).....	17
Abbildung 11: Blockschaltbild Aktoren.....	18
Abbildung 12: Zuordnung interne I/Os zu Application-Devices.....	20
Abbildung 13: Zuordnung interne und externe I/Os zu Application-Devices.....	21
Abbildung 14: Beispiel Gesamt-System (intern).....	22
Abbildung 15: CGI –Tool-Verbindung.....	24
Abbildung 16: eWebTest : get_data.cgi.....	24
Abbildung 17: eWebTest : get_data_value.cgi.....	26
Abbildung 18: eWebTest : get_data_defintion.cgi.....	27
Abbildung 19: Berechtigungen UDP-Config-Server.....	29
Abbildung 20: Berechtigungen TCP-Server.....	30
Abbildung 21: Berechtigungen UDP-Server.....	31
Abbildung 22: Berechtigungen TCPCOMM.....	32
Abbildung 23: Beispiel get_data.....	37
Abbildung 24: Beispiel get_data_dv.....	37
Abbildung 25: System-Start.....	42
Abbildung 26: /etc/rc eWebTest.....	43
Abbildung 27: Programm gen_devs.....	44

## Liste der Tabellen

### Liste der Abkürzungen

AIN	Analoger Eingangs-Datenpunkt
AOT	Analoger Ausgangs-Datenpunkt
Application-Device	Eintrag eines (speziellen) Zeichen-orientierten Geräts (Character-Device)
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
FLASH	FLASH-EXTERN, Speicherbereich des NAND-FLASH Bausteins
ID	Interne Identifikationsnummer
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)

# 1 Einleitung

## 1.1 Zielsetzung

Dieses Dokument beschreibt die Nutzung, Anpassung und Administration der Software-Ausstattung, die auf dem eWebSrv-Gerät eingesetzt wird.

**Hinweis:** Sollten Sie die Handbücher „Inbetriebnahme“ und „Benutzer“ 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 Grundlagen

### 2.1 Betriebssystem

Als Betriebssystem wird uClinux ([www.uclinux.org](http://www.uclinux.org)) eingesetzt. Dabei handelt es sich um eine Distribution mit einem Standard-Kernel (angepasst auf Mikrocontroller ohne MMU).

Der „eWebSrv“ ist eine Modul-Baugruppe, welches überwiegend in einem Netzwerk eingesetzt wird. Es sind Anwendungen wie z.B. HTTP, FTP und TELNET implementiert, welche sich in einem Dateisystem (JFFS2) im NAND-FLASH des eWebSrv befinden.

Auf dem Modul werden eine Vielzahl von Programmen, Treibern und Diensten gestartet, diese können oder müssen im System miteinander kommunizieren. Um diese Kommunikation effizient zu gestalten sind zusätzlich zu der bekannten Linux IPC (Inter-Prozess-Kommunikation) weitere Schnittstellen von **mhc** implementiert.

**Hinweis:** In den folgenden Abschnitten werden die Begriffe 'Konfiguration', 'Parameter' und 'Messwerte' verwendet.

Unter dem Begriff 'Konfiguration' verstehen wir alle Kennwerte, welche die Hard- und Software eines auf dem eWebSrv basierenden Gerätes eindeutig beschreiben. Diese Kennwerte werden von **mhc** bei der Produktion festgelegt, protokolliert und auf dem Gerät gespeichert. Der Anwender kann diese Kennwerte lesen und verarbeiten, eine Änderung seitens des Anwenders ist ausgeschlossen.

'Parameter' sind Werte und Einstellungen, welche die Funktion des Gerätes beim Anwender beeinflussen. Diese Kennwerte werden von **mhc** bei der Produktion festgelegt, protokolliert und gespeichert. Der Anwender kann diese aber entsprechend seinen Anforderungen ändern.

Unter 'Mess- und Stellwerten' verstehen wir alle im Betrieb gemessen bzw. gesetzten Werte für die Peripherie (analoge/digitale Ein- bzw. Ausgänge). Diese werden unterteilt in einen Teil für die Definitionen und in einen Teil, welcher die reinen Mess-/Stellwerte enthält.

Alle gemachten Definitionen sind so organisiert, dass die oben beschriebene Unterteilung für die Konfiguration, die Parameter und die Messwerte gewährleistet ist. Die Definitionen selbst sind in den eigenständigen Dokumenten MkcDefCommonLib bzw. MkcDefUserLib spezifiziert.

Die notwendige **Konfiguration** (CONFIG) des Gerätes erfolgt durch **mhc** während der Produktion und wird im Testprotokoll dokumentiert. Jedes Gerät mit einem eWebSrv wird in einem definiertem Auslieferungszustand an den Anwender geliefert.

### 2.2 System-Zugriff

Die Kommunikation mit dem Gerät wird im Regelfall über ein Netzwerk erfolgen, sobald das Gerät gebootet ist und den definierten Betriebszustand erreicht hat. (Einzige Ausnahme ist der Zugriff über die serielle Schnittstelle für Service-Zwecke.)

Um Zugriff auf das Betriebssystem zu erhalten, muss daher eine Verbindung via TELNET möglich sein. Nach einem erfolgreichen Login stehen die bekannten Unix-Befehle (cp, rm, mv,...) über die System-Shell (msh) zur Verfügung. Es können dann die üblichen Datei-Operationen (Kopieren, Löschen, Umbenennen,...) vorgenommen werden.

Sollten Sie bereits den Zugriff für TELNET aktiviert haben, können Sie dieses Kapitel überspringen.

**Hinweis:** Veränderungen an System-Dateien können zu einem eingeschränkten bis nicht mehr lauffähigen System führen. Es ist dringend angeraten Sicherheitskopien zu erstellen bevor Änderungen vorgenommen werden.

## 2.2.1 TELNET aktivieren

Um den Zugriff zu aktivieren, wählen Sie bitte die Seite „Parameter“ an und stellen unter dem Menü-Punkt „TELNET-Server“ den Zustand : „JA“ ein und drücken den Knopf „Senden“.

Sie befinden sich hier: [Mess- und Stellwerte](#) / [Parameter](#)

Parameter

- [eWebSrv](#)
- [FPGA](#)
- [Extension](#)

eWebSrv

- [SYSTEM](#)
- [DIO](#)
- [SERIAL](#)

**eWebSrv**

HTTP Benutzer	Operator	<input type="text" value="user_rw"/>	<input type="text" value="pass_rw"/>	<input type="button" value="Senden"/>		
Netzwerk	IP-Adresse	<input type="text" value="192"/>	<input type="text" value="168"/>	<input type="text" value="015"/>	<input type="text" value="097"/>	<input type="button" value="Senden"/>
	IP-Maske	<input type="text" value="255"/>	<input type="text" value="255"/>	<input type="text" value="255"/>	<input type="text" value="000"/>	
	IP-Gateway	<input type="text" value="000"/>	<input type="text" value="000"/>	<input type="text" value="000"/>	<input type="text" value="000"/>	
	Verbindungsart	<input type="text" value="AUTO"/>				
Netzwerk (UDP)	Starten	<input type="text" value="Nein"/>				<input type="button" value="Senden"/>
	Port	<input type="text" value="49154"/>				
	Freigabe	<input type="text" value="255"/>	<input type="text" value="255"/>	<input type="text" value="255"/>	<input type="text" value="255"/>	
HTTP-Server	Port	<input type="text" value="80"/>				<input type="button" value="Senden"/>
FTP-Server	Starten	<input type="text" value="Nein"/>				<input type="button" value="Senden"/>
TELNET-Server	Starten	<input type="text" value="Ja"/>				<input type="button" value="Senden"/>
Daten-Server (TCP)	Starten	<input type="text" value="Nein"/>				<input type="button" value="Senden"/>
	Port	<input type="text" value="49152"/>				
	Freigabe	<input type="text" value="255"/>	<input type="text" value="255"/>	<input type="text" value="255"/>	<input type="text" value="255"/>	
	Zeit	<input type="text" value="2 s"/>				
Daten-Server (UDP)	Starten	<input type="text" value="Nein"/>				<input type="button" value="Senden"/>
	Port	<input type="text" value="49153"/>				
	IP-Adresse	<input type="text" value="192"/>	<input type="text" value="168"/>	<input type="text" value="015"/>	<input type="text" value="255"/>	
	Freigabe	<input type="text" value="255"/>	<input type="text" value="255"/>	<input type="text" value="255"/>	<input type="text" value="255"/>	
	Zeit	<input type="text" value="2 s"/>				
NTP-Client	Starten	<input type="text" value="Nein"/>				<input type="button" value="Senden"/>
	Server IP-Adresse	<input type="text" value="192"/>	<input type="text" value="168"/>	<input type="text" value="015"/>	<input type="text" value="090"/>	

**MKC Michels & Kleberhoff Computer GmbH**  
 42329 Wuppertal, Vohwinkel Str. 58  
 Tel.: 0202 / 27317 - 0, Fax: 0202 / 27317 - 49, [EMAIL-KONTAKT](#)

Abbildung 1: TELNET aktivieren

## 2.2.2 Verbindungstest

Um die Verbindung zu testen, öffnen Sie bitte eine Konsole auf ihrem PC und rufen das Programm TELNET mit der IP-Adresse des Geräts auf.

Im Auslieferungszustand ist die IP-Adresse 192.168.15.97 und nur der Benutzer „root“ mit dem Passwort „uClinux“ verfügbar.



```

c:\ Telnet 192.168.15.97
uClinux login: root
Password:
Welcome to

uClinux/MKc

For further information check:
http://www.uclinux.org/ and http://www.mkc-gmbh.de/

BusyBox v1.00 (2010.03.17-07:51+0000) Built-in shell (msh)
Enter 'help' for a list of built-in commands.
~#
```

*Abbildung 2: Verbindungstest*

### 3 Mess- und Stellwerte

Der eWebSrv selbst verfügt nur über DIO Datenpunkte, gleiches gilt für die Trägerkarte eWebTest. Durch die Einbindung externer Geräte können aber beliebig viele weitere Datenpunkte aller Datentypen vorhanden sein. Aus diesem Grund erfolgt hier auch eine Beschreibung der Behandlung aller anderen Datentypen. Prinzipiell könnte natürlich auch eine beliebige Trägerkarte zum Einsatz kommen, die ihrerseits verschiedene Datenpunkte bereit stellt.

Die analogen Eingänge und Ausgänge werden über eine einfache mehrstufige Operationskette betrieben. Von **mhc** werden bei der Produktion Korrekturfaktoren für die analogen Ein-/Ausgänge ermittelt und im CONFIG-Bereich gespeichert.

Die oben angesprochene Operationskette nutzt diese Informationen um die Temperaturabhängigkeit und Bauteiltoleranzen der analogen Beschaltung zu korrigieren.

Das Lesen von Messwerten bzw. das Schreiben von Sollwerten kann über die weiter unten angesprochenen Server erfolgen. Die Parametrierung erfolgt ausschließlich über das HTTP-Protokoll.

#### 3.1 Sensoren, Eingänge

Die Eingänge werden in festen zeitlichen Intervallen eingelesen. Dieses Intervall wird von **mhc** vorgegeben und kann nicht parametriert werden.

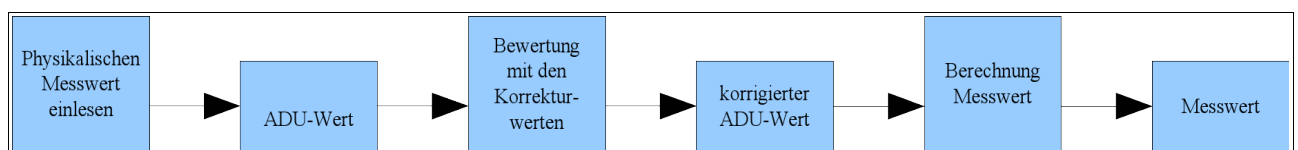
Der gemessene analoge Wandlerwert ADU wird mit den Korrekturfaktoren (Offset, Verstärkung) bewertet. Diese werden von **mhc** bei der Produktion eines Geräts festgelegt.

Durch diesen Schritt werden Temperaturabhängigkeiten und Bauteiltoleranzen der Eingangsstufen korrigiert. Diese Korrekturen entfallen bei digitalen Eingängen.

Danach werden die korrigierten Werte in Messwerte (Spannung, Temperatur, etc.) umgerechnet. Diese Messwerte entsprechen den möglichen Messbereichen der analogen Eingänge (Volt bei Spannungsmessung, Ampere bei Strommessung, Grad Celsius bei Temperaturmessung, etc.).

Bei allen Eingängen wird ein Faktor 'Filter' eingeführt. Je nach Zustand dieses Faktors wird eine Filterung der Eingänge vorgenommen.

Das folgende Diagramm stellt das prinzipielle Verfahren dar.



*Abbildung 3: Blockschaltbild Sensoren*

Die Messwerte werden zyklisch gelesen und mit den vorherigen Werten verglichen. Wenn hierbei Änderungen bei digitalen Eingängen bzw. Änderungen außerhalb der definierten Hysterese bei analogen Eingängen auftreten, wird ein Zustandswechsel erkannt.

### 3.1.1 Filterung der digitalen Eingänge

Um Störungen aus dem Eingangssignal zu filtern und um – von der Hardware gleichgerichtete – Wechselspannungen zu verarbeiten, wird dem Anwender nicht der unmittelbare aktuelle Messwert des digitalen Eingangs geliefert, sondern ein gewichtetes arithmetisches Mittel über alle Messungen. Hierzu wird die folgende Formel benutzt:

$$AM_n = AM_{n-1} - \frac{(AM_{n-1} + \frac{2^{Filter}}{2})}{2^{Filter}} + 2 * P_{Messwert} \quad ; P \in [0,1], AM \in \begin{cases} \mathbb{N} \text{ für } AM \geq 0 \\ \text{sonst } 0 \end{cases}$$

$$DIN = HIGH \text{ wenn } AM > (\frac{2^{Filter}}{2})$$

$$DIN = LOW \text{ wenn } AM \leq (\frac{2^{Filter}}{2})$$

Abbildung 4: Formel arithmetische Mittelwertberechnung (DIN)

Aufgrund dieser Formel wird die Bewertung der „neuen“ Messwerte immer niedriger, je größer der Parameter 'Filter' gewählt wird. Ein Wert von 1 für 'Filter' gibt den direkten digitalen Eingang zurück.

**Hinweis:** Der Formel-Faktor 'Filter' kann auf der Homepage auf der Seite 'Parameter' für jeden digitalen Eingang eingestellt werden. Der Wert 0 für 'Filter' ist nicht erlaubt.

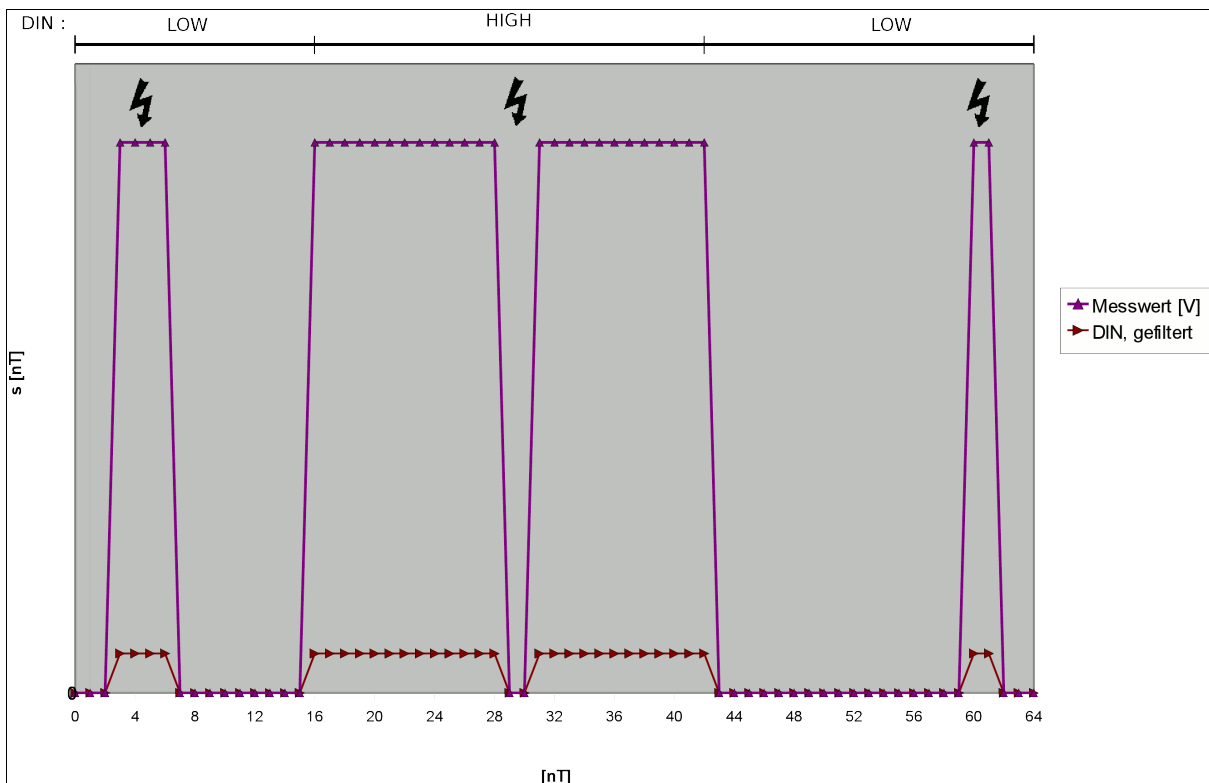


Abbildung 5: DIN (Filter = 1, keine Filterung des Eingangssignals)

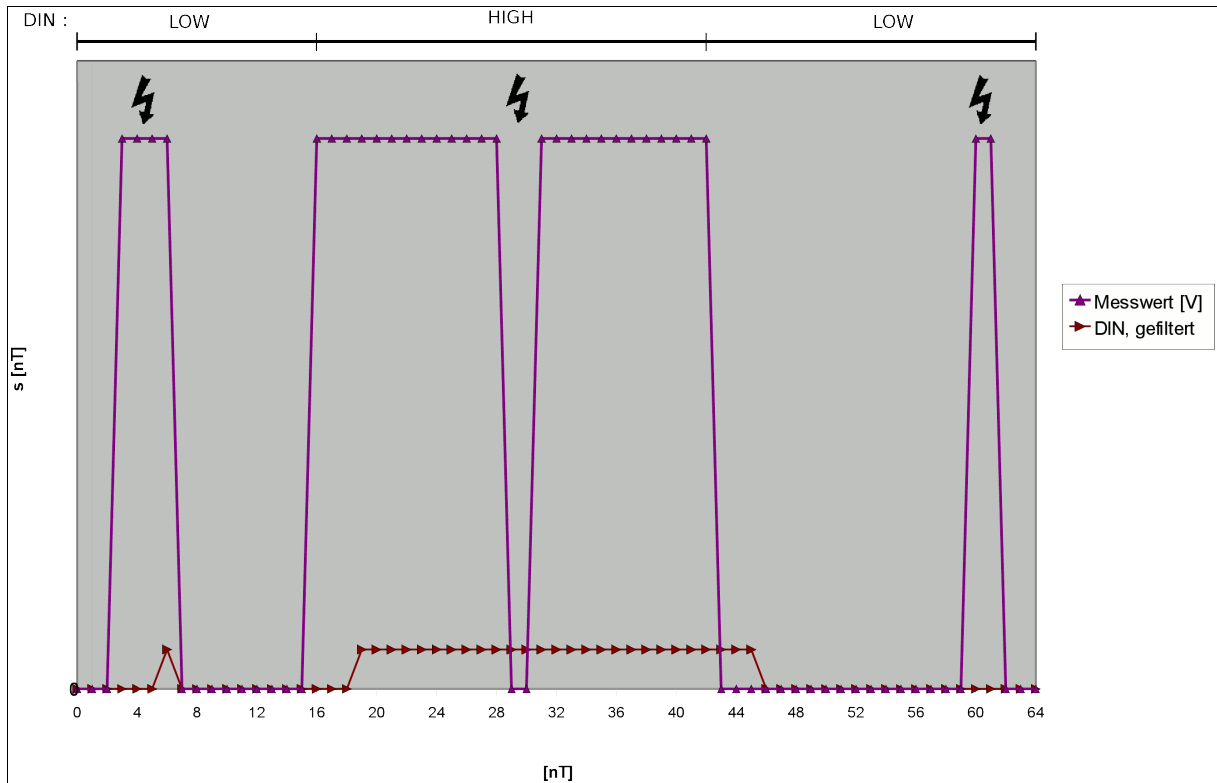


Abbildung 6: DIN ( Filter = 3, Filterung der kurzen Störungen)

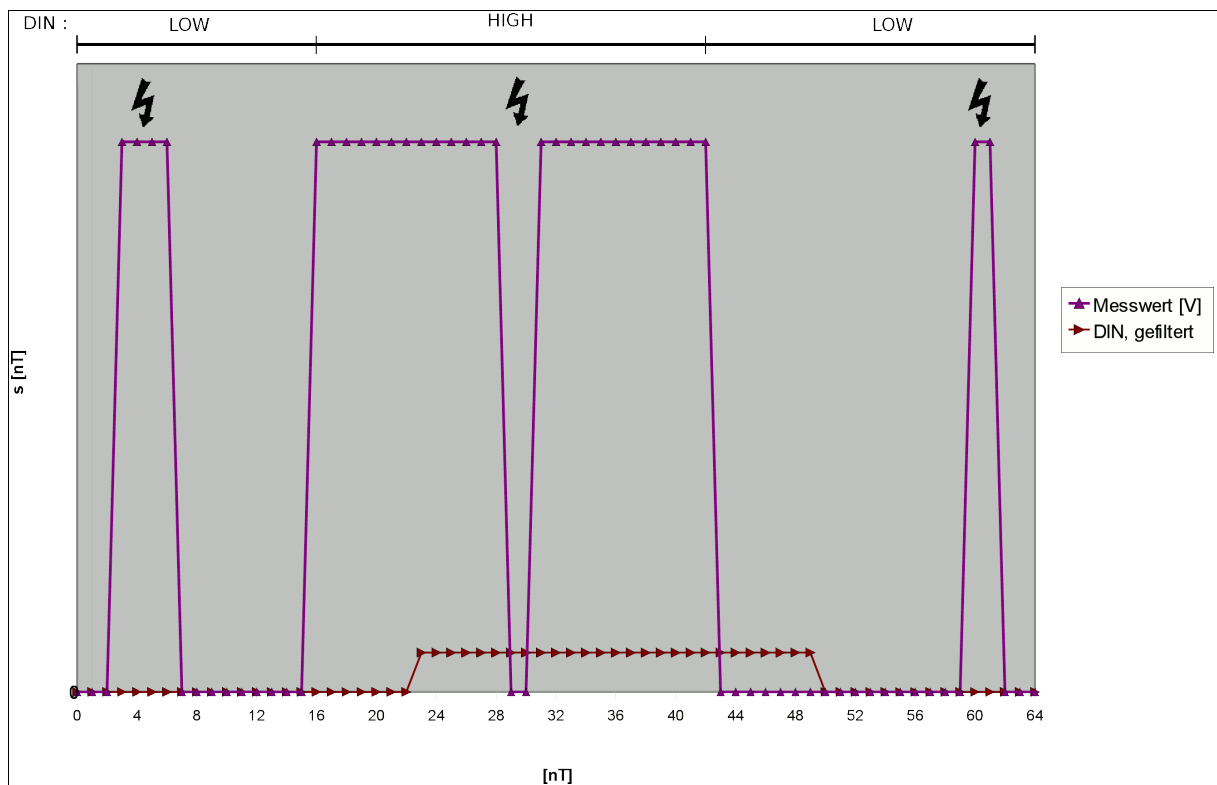


Abbildung 7: DIN ( Filter = 4, Filterung aller Störungen)

### 3.1.2 Filterung der analogen Eingänge

Um Störungen aus dem Eingangssignal zu filtern wird dem Anwender nicht das Ergebnis des gemessenen analogen Eingangs geliefert, sondern ein gewichtetes arithmetisches Mittel über alle Messungen. Hierzu wird die folgende Formel benutzt:

$$AM_n = AM_{n-1} - \left(\frac{AM_{n-1}}{2^{Filter}}\right) + Messwert$$

$$AIN = \left(\frac{AM}{2^{Filter}}\right)$$

Abbildung 8: Formel arithmetische Mittelwertberechnung (AIN)

Aufgrund dieser Formel wird die Bewertung der „neuen“ Messwerte immer niedriger, je größer der Parameter 'Filter' gewählt wird. Ein Wert von 0 für 'Filter' gibt den direkten analogen Eingang zurück.

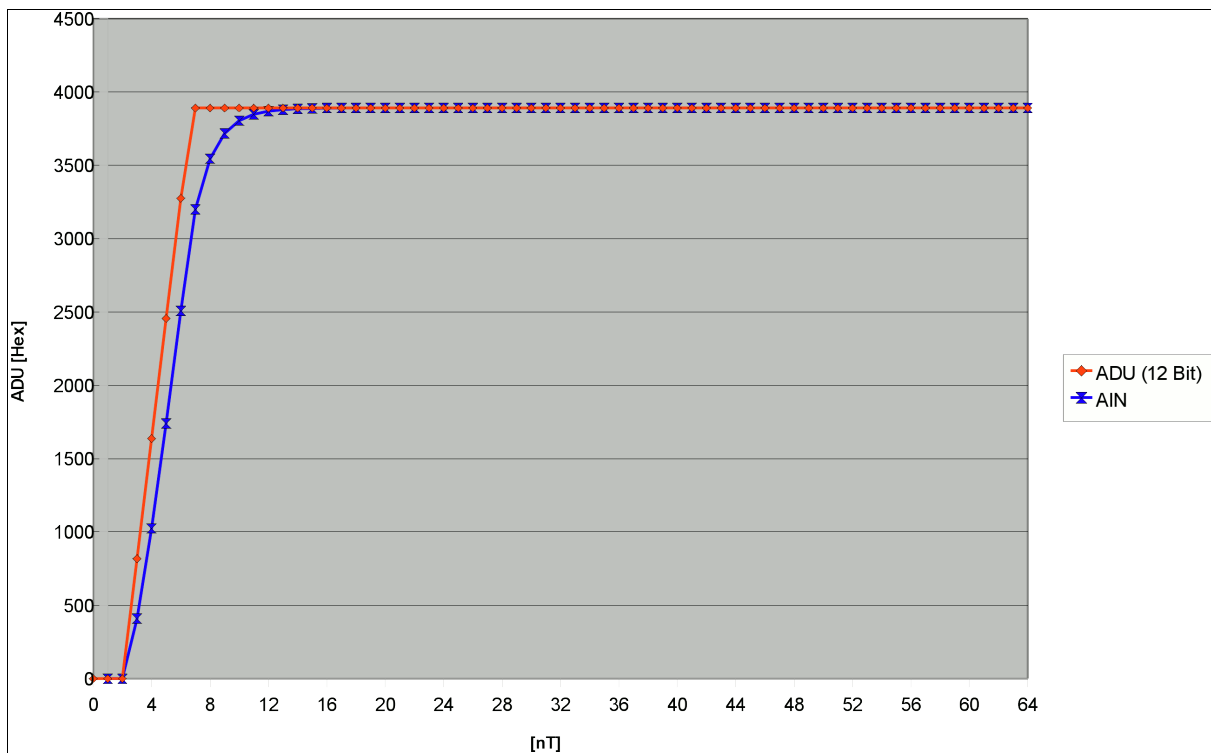


Abbildung 9: AIN (Eingangssignal-Sprung von 0% auf 100%, Filter = 1, Filterung des Eingangssignals)

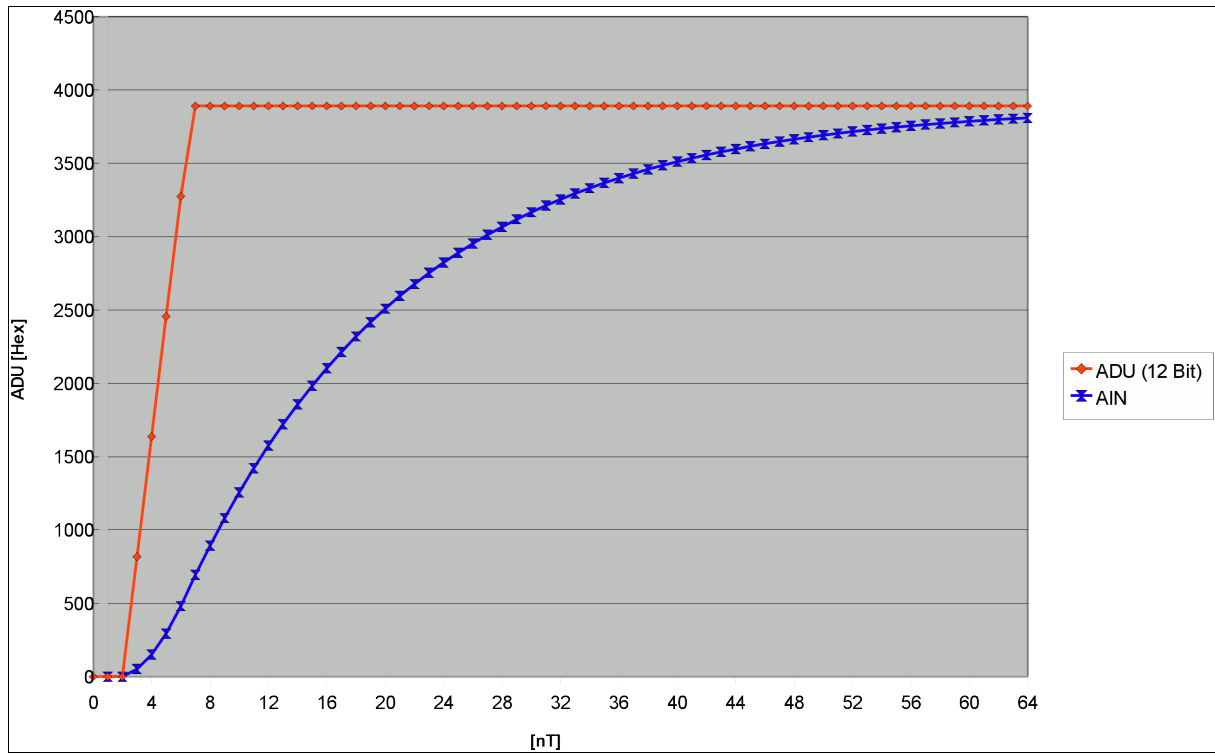


Abbildung 10: AIN (Eingangssignal-Sprung von 0% auf 100%, Filter = 4, Filterung des Eingangssignals)

### 3.1.3 Hysterese der analogen Eingänge

Bei der implementierten Messwerterfassung ist für jeden analogen Eingang ein Wert definiert, der eine Hysterese des Messwertes darstellt. Die Hysterese wird als Wertebereich angegeben und kann auf der Homepage auf der Seite 'Parameter' für die analogen Eingänge eingestellt werden.

Eine Hysterese von zum Beispiel 5 gibt an, dass der neue Messwert um  $\pm 5$  vom letzten vorherigen Messwert abweichen muss, damit dieser Eingang als geändert gilt.

### 3.2 Aktoren, Ausgänge

Bei den analogen Ausgängen wird der übergebene Sollwert (zum Beispiel 3.45 mA) auf den DAU-Wert umgerechnet. Bei digitalen Ausgängen entfallen diese Umrechnungen.

Der DAU-Wert wird anschließend mit den Korrekturfaktoren (Offset, Verstärkung), welche bei der Produktion eines Gerätes von **MHC** festgelegt wurden, bewertet. Durch diesen Schritt werden Temperaturabhängigkeiten und Bauteiltoleranzen der Ausgangsstufen korrigiert. Der aktuelle Zustand des physikalischen Analogausgangs wird unmittelbar mit dem Kommando geändert.

Die Ausgänge werden in festen zeitlichen Intervallen kontrolliert. Dieses Intervall wird von **MHC** vorgegeben und kann nicht parametrisiert werden. Hat sich die Temperatur geändert, wird der DAU-Wert erneut korrigiert und auf den physikalischen Ausgang geschrieben.

Das folgende Diagramm stellt das prinzipielle Verfahren dar.



Abbildung 11: Blockschaltbild Aktoren

Die Stellwerte können von mehreren Quellen, wie zum Beispiel Daten-Servern, gesetzt werden.

#### 3.2.1 Remanenz der Ausgänge

Für analoge und digitale Ausgänge ist eine sogenannte „Remanenz“ definiert. Diese kann auf der Homepage-Seite 'Parameter' für die analogen und digitalen Ausgänge eingestellt werden.

Ist der entsprechende Ausgang remanent definiert, so wird der aktuelle Stellwert dieses Ausgangs in einem internen nichtflüchtigen Datenspeicher (Flash) gespeichert. Bei einem Systemneustart werden alle remanenten Ausgänge mit diesen abgespeicherten Stellgrößen initialisiert. Somit liegen an den jeweiligen Klemmen die letzten übertragenen Stellwerte an.

### 3.3 Daten-Organisation

Um den I/O-Zugriff auf die Datenpunkte aller **MHC**-Geräte zu Vereinheitlichen, wird für alle Zugriffe ein einheitliches Datenmodell zur Verfügung gestellt. Dieses besteht aus einer Definition „decl\_data\_definition“ und den eigentlichen Daten „decl\_data\_value“. Bei „decl\_data\_value“ und „decl\_data\_definition“ handelt es sich um Strukturen die in der Programmiersprache „C“ definiert sind.

Alle im System vorhandenen Datenpunkte werden durch ein Speicherabbild repräsentiert, d.h. der Anwender greift immer auf dieses Speicherabbild zu und nicht direkt auf die Hardware. Schreib-/Lese-Zugriffe auf die Hardware befinden sich unter Kontrolle des Linux-Kernels und werden über interne Prozesse realisiert.

Datenpunkte werden von verschiedenen Hardware Modulen bereitgestellt. Da ist zunächst die CPU selbst, sie stellt auf dem eWebSrv 16 bidirektionale digitale I/Os zur Verfügung (in der Notation der **MHC** Definitionen DIO\_IN und DIO\_OUT). Das auf dem Modul befindliche FPGA kann weitere Datenpunkte zur Verfügung stellen. In der IO48 Variante z.B. 48 I/Os der Typen DIO\_IN, DIO\_OUT und DIO\_OUT\_OD. Prinzipiell könnte auch eine Trägerkarte weitere Datenpunkte liefern - ein eWebSrv Modul befindet sich immer auf einer Trägerkarte !

Aus diesem Grunde wird die Einheit aus CPU/FPGA/Trägerkarte als eine „Unit“ definiert und dem zufolge als 'Unit0' bezeichnet.

Wie schon erwähnt, werden die Datenpunkte der Unit0 im System abgebildet mittels eines Definitionsbereiches und dem dazugehörigen Datenbereich. Der Definitionsbereich beschreibt die Anzahl, Art und Position der Daten im Speicherabbild und gibt ihm einen eindeutigen Namen.

Weitere Datenpunkte kommen hinzu, wenn der eWebSrv über das Netzwerk mit zusätzlichen Datenservern (z.B. IONet) verbunden wird. Auf dem eWebSrv befinden sich Daten-Client Programme, die Verbindung mit den externen Servern aufnehmen. Jedes der verbundenen Geräte wird dann im Speicherabbild durch eine Unit repräsentiert, die über einen eigenen Definitions- und Datenbereich verfügt.

Im Verzeichnis-Baum des Betriebssystems ansprechbar (genauer: /dev/data) befindet sich für jede dieser Units ein Eintrag. Der Name ist beliebig und wird über eine Konfigurationsdatei festgelegt. Unit-Devices sind nur für Datenaustausch zwischen Client-Programmen und dem System (Kernel).

Für Anwendungsprogramme werden beim Systemstart über Konfigurationsdateien sog. Application-Devices generiert, in denen Datenpunkte aus beliebigen Units als Funktionsgruppen zusammengefasst werden. Auch diese Devices verfügen wieder jeweils über einen eigenen Definitions- und Datenbereich und werden durch einen Eintrag im Verzeichnis-Baum repräsentiert.

Grundsätzlich ist es möglich, dass Datenpunkte des eWebSrv direkt an die Hardware der Trägerkarte angeschlossen bzw. betrieben werden, beispielsweise für JTAG. Wenn diese Datenpunkte zur „freien“ Verfügung gestellt werden, ist eine korrekte Funktion der internen Hardware unter Umständen nicht mehr gewährleistet, weil beliebig viele Applikationen ohne zeitlichen Bezug auf einen Datenpunkt zugreifen könnten. Aus diesem Grund wurde das Flag „owner“ eingeführt. Dieses Flag definiert zwei Zustände:

- „private“ besagt, dass dieser Datenpunkt ausschließlich für interne Zwecke zur Verfügung steht und somit niemals von außen („extern“) sichtbar ist.
- „public“ besagt, dass dieser I/O nicht von interner Hardware benutzt wird.

Ist die Hardware (Trägerkarte) **MHC** bekannt, stellt die Firmware sicher, dass diese I/Os nicht „public“ werden. So werden zum Beispiel die Datenpunkte DIO[0..3] dem Device „JTAG“ zugeordnet. Ein Kunde kann für selbst erstellte Hardware (z.B. Extension beim eWebSrv) über den Parameterbereich zusätzliche Datenpunkte als „private“ kennzeichnen

Grundsätzlich gilt :

Alle als „public“ deklarierten Datenpunkte, welche keinem Device zugewiesen sind, werden automatisch dem Device **public** (Public-Device) zugeordnet.

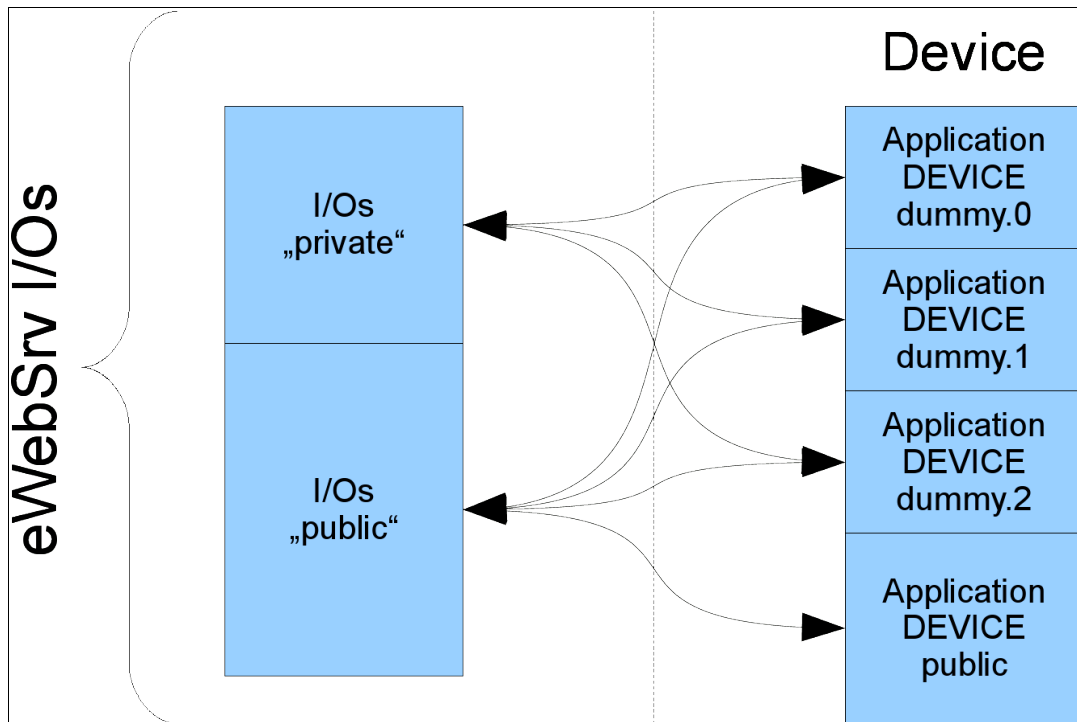


Abbildung 12: Zuordnung interne I/Os zu Application-Devices

Beim Systemstart werden die Konfigurationsdateien abgearbeitet, mit denen Application-Devices die zur Verfügung stehenden Datenpunkte zugeordnet werden. In Abbildung 12 werden I/Os den Devices dummy.0 bis dummy.2 zu geordnet.

Ein Gerät, das auf dem eWebSrv basiert, kann sowohl Daten-Server als auch Daten-Client sein. Ein oder mehrere Clients fügen Daten entfernter Geräte dem Datenpool hinzu, ein Daten-Server stellt Mess-/Stellwerte einem Hostrechner zur Verfügung. Über das Netzwerk verbundene externe Datenserver werden komplett als „private“ oder „public“ definiert.

Daraus ergibt sich folgende erweiterte Abbildung:

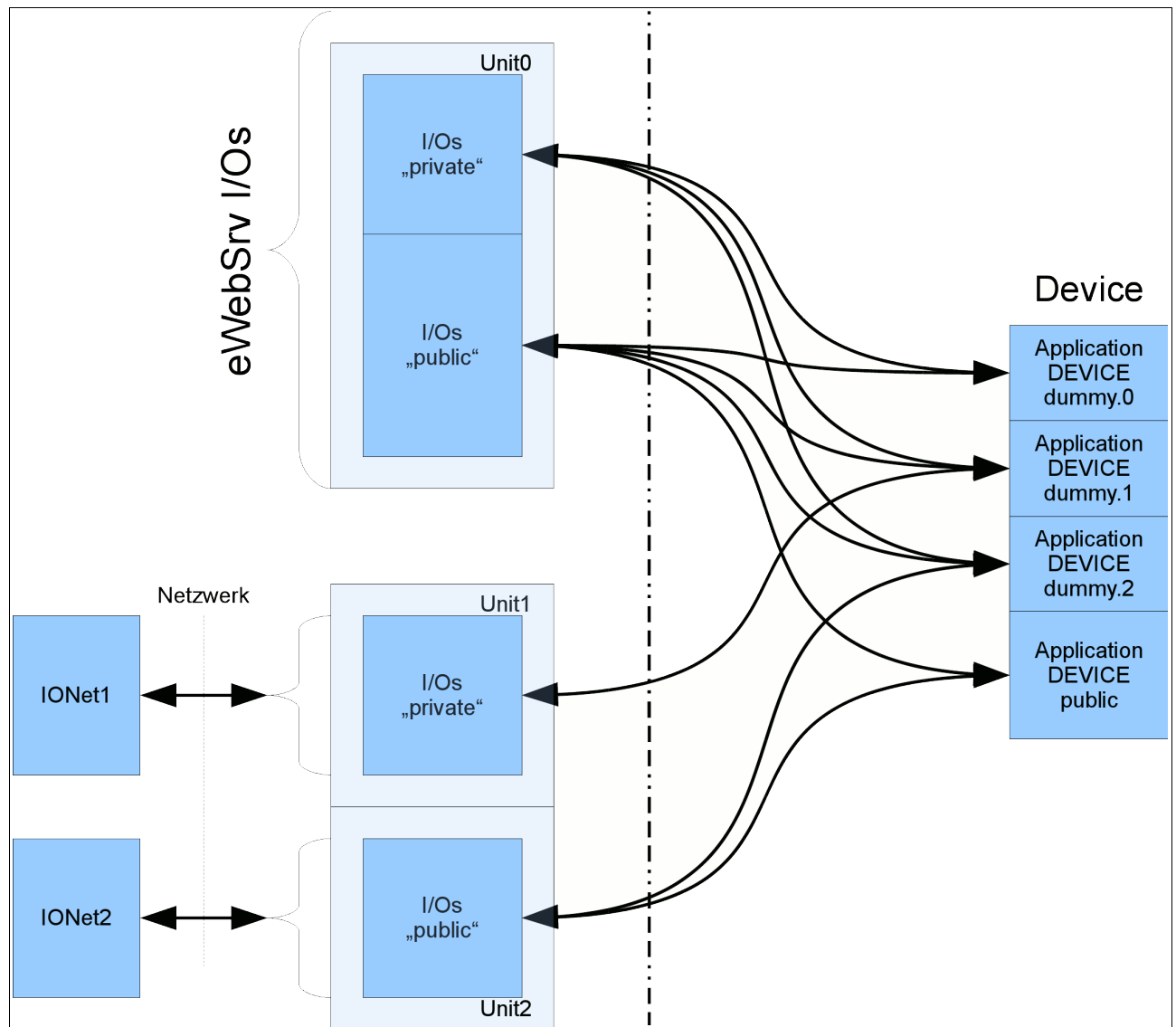


Abbildung 13: Zuordnung interne und externe I/Os zu Application-Devices

**Hinweis:** Der HTTP-Server hat nur Zugriff auf das Public-Device

Sowohl die Zugriffe von externer Hardware (IONet) zum Unit-Device, als auch die Verbindung Application-Device zu einem beliebigen Anwenderprogramm (z.B. Daten-Server) wird grundsätzlich über die von **mhc** standardisierten Interfaces bzw. Strukturen realisiert.

In Abbildung 14 sind die Geräte IONET 1...2 als Units „ionet.1“ bzw. „ionet.2“ innerhalb des eWebSrv-Systems eingebunden. Der Zugriff erfolgt über die Application-Devices **public**, **dummy.0**, **dummy.1**.

Die Programme A1,A4 und A5 können beliebige Anwendungen sein, A3 ist ein Daten-Server für das Public-Device.

Das Device **dummy.1** ist „private“ und somit exklusiv für die lokale Anwendung **A5**, d.h. **A1** hat keinen direkten Zugriff auf Device **dummy.1**. Soll diese Anwendung aber trotzdem z.B. den Status der Datenpunkte lesen können, muss **A5** eine Schnittstelle hierfür zur Verfügung stellen.

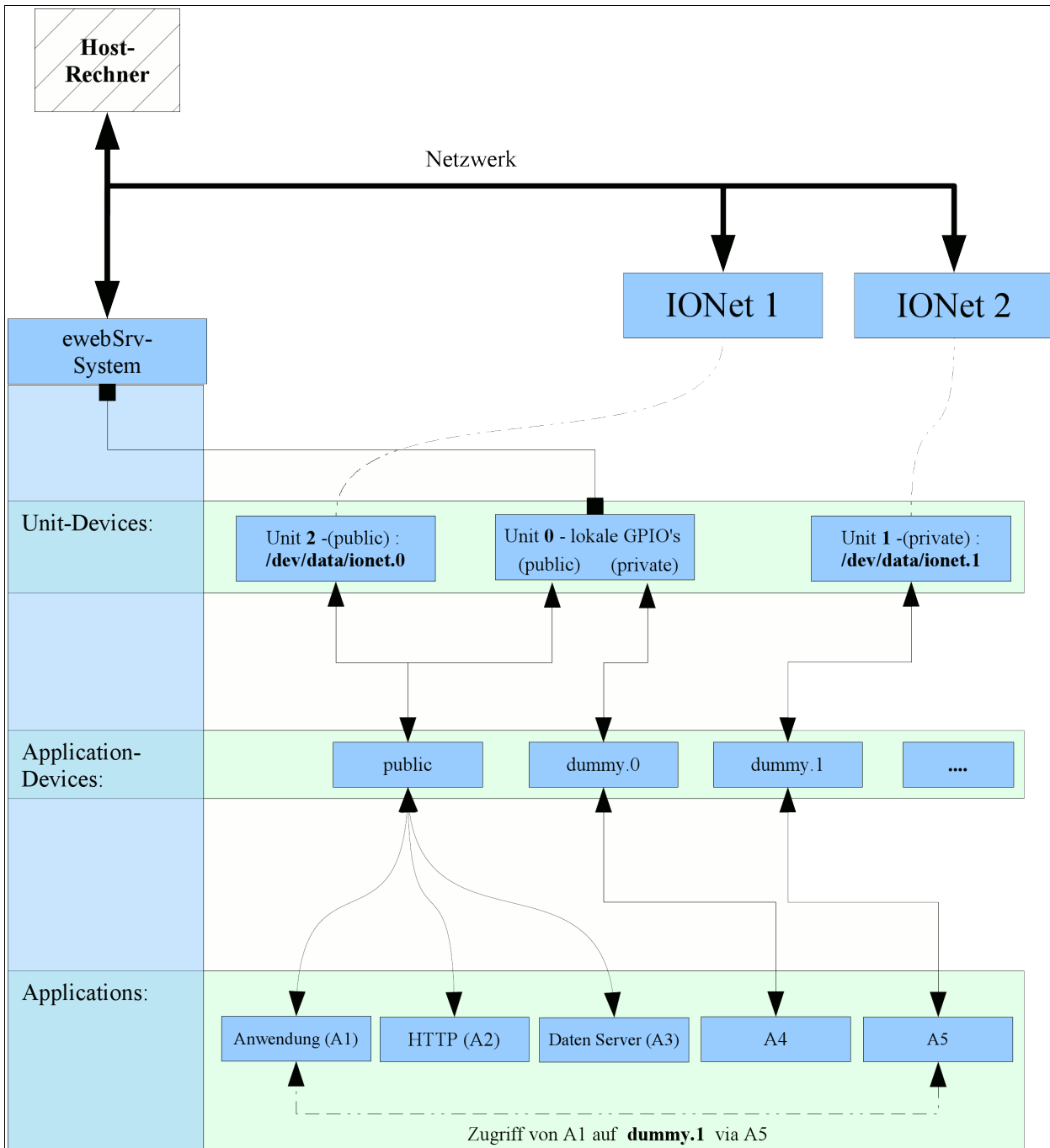


Abbildung 14: Beispiel Gesamt-System (intern)

Global stellt sich das Gesamt-System wie folgt dar. Wird der Zugriff auf die Mess- und Stellwerte bzw. Sensoren und Aktoren über das Netzwerk genutzt, kann der Datenaustausch ausschließlich zwischen dem Hostrechner und dem eWebSrv-System erfolgen. Die Daten-Server bzw. Daten-Clients werden im folgenden Kapitel erläutert.

## 4 Anwendungen

### 4.1 Vorbemerkungen zum SYSFS

Nur der System-Kernel kann auf den CONFIG- bzw. PARAM-Bereich direkt zugreifen. Um Informationen zwischen dem Kernel und einzelnen Programmen auszutauschen, existiert die standardisierte Schnittstelle SYSFS (genauer /sys/...). Aus diesem Grund werden Zugriffe z.B. auf den PARAM-Bereich über das SYSFS realisiert, in Form von sogenannten Attributen (Dateien). Der Attribut-Name leitet sich aus der Variablen-Bezeichnung (**MHC**-Strukturen) ab. Beispielsweise wird die Variable „*eth\_start*“ beim System-Start als auch zur Laufzeit benötigt um diverse Programme zu starten bzw. stoppen.

Detaillierte Informationen zum SYSFS und seiner Handhabung befinden sich im Handbuch „System-Programmierer“.

### 4.2 HTTP-Server

Als erste Schnittstelle zu dem eWebSrv ist das HTTP-Protokoll implementiert. Im Auslieferungszustand läuft der HTTP-Server auf Port 80. Eine Änderung ist per HTTP-Zugriff zu einem späterem Zeitpunkt möglich.

Die Benutzeroberfläche ist kompatibel zu den gängigen Browsern wie IE 8.0, Mozilla Firefox 3.x, etc. implementiert. Die Benutzer-Anmeldung und die HTML-Seiten wurden bereits im Handbuch „Benutzer“ beschrieben.

#### 4.2.1 Realisierung

Verwendet wird der 'thttpd' aus der uClinux Distribution.

Der thttpd Server wird als daemon im file '/etc/inittab' gestartet. Sein Home-Verzeichnis ist '/home/httpd/'.

Bei einem HTTP-Server können mit Hilfe von Server Side Includes direkt innerhalb von HTML-Dateien dynamische Informationen ausgegeben werden. Es können auch CGI-Programme gestartet und deren Ausgaben direkt in die HTML-Datei eingebunden werden. Beim eWebSrv sind CGI-Programme in der Regel Shell-Skripte die Systemprogramme aufrufen. Die Skripte 'übergeben' dabei die in der URL enthaltenen Parameter an das jeweilige Programm, welche mit den definierten Datenstrukturen arbeiten.

Der HTTP Server führt CGI-Programme aus Sicherheitsgründen nur aus, wenn sie sich in oder unter dem Verzeichnis 'cgi-bin' befinden. Dateien mit der Endung .cgi werden in diesem Unterverzeichnis gesucht, der Pfad muss in der URL nicht angegeben werden. Bei CGI-Programmen ohne die Endung .cgi ist die Pfadangabe erforderlich.

HTML Seiten sind im home (/home/thttpd) des HTTP Servers abgelegt. Der thttpd Server ist so modifiziert, dass Seiten die SSI Aufrufe enthalten an der Endung .shtml erkannt und verarbeitet werden.

Verzeichnisbaum des HTTP Servers:

http-root (/home/httpd)	
bin	Server spezifische SSI Programme
cgi-bin	CGI-Skripte und Programme
config	Konfiguration des Servers
var	auf eine RAM-Disk gemountet: von der Serverumgebung erzeugte Files, die nicht erhalten werden müssen, z.B. log-files

### 4.2.2 Funktionsaufrufe (SSI, CGI)

Im Verzeichnis /home/httpd/bin befinden sich Programme, die mit get\_ beginnen. Alle diese Programme bauen dynamisch HTML-Seiten auf und werden als SSI aufgerufen.

Eine ausführliche Erläuterung erfolgt hier nicht, der Aufruf mit -? als Parameter liefert eine kurze Erklärung

Die CGI-Skripte werden nur vom HTTP-Server benötigt, sind auf jedem ausgeliefertem Gerät vorhanden und befinden sich im Verzeichnis /home/httpd/cgi-bin. Sie stellen eine Verbindung zwischen HTTP-Server und den System-Tools aus Kapitel 5. dar.

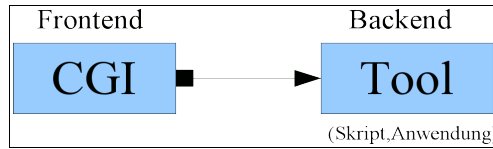


Abbildung 15: CGI-Tool-Verbindung

#### 4.2.2.1 get\_data.cgi

'get\_data.cgi' ist ein Skript, das den Query-String einer HTTP-Anforderung für das Programm 'get\_data' aufbereitet und dessen Ausgabe als 'text/plain' zurück liefert.

Die implementierte Funktion liefert bei einer HTTP-Anforderung alle Stell- und Messwerte in einem definierten Format, welches auf den definierten Datenstrukturen basiert. Diese Funktion darf von allen definierten Benutzern (siehe Handbuch Benutzer) aufgerufen werden.

Bei dem Aufruf ohne Parameter liefert der HTTP-Server alle definierten Werte in ihrem aktuellen Zustand:

Aufruf: get\_data.cgi

Ergebnis: ID\_REV\_NR:NAM[x]=Wert

Beispiel:

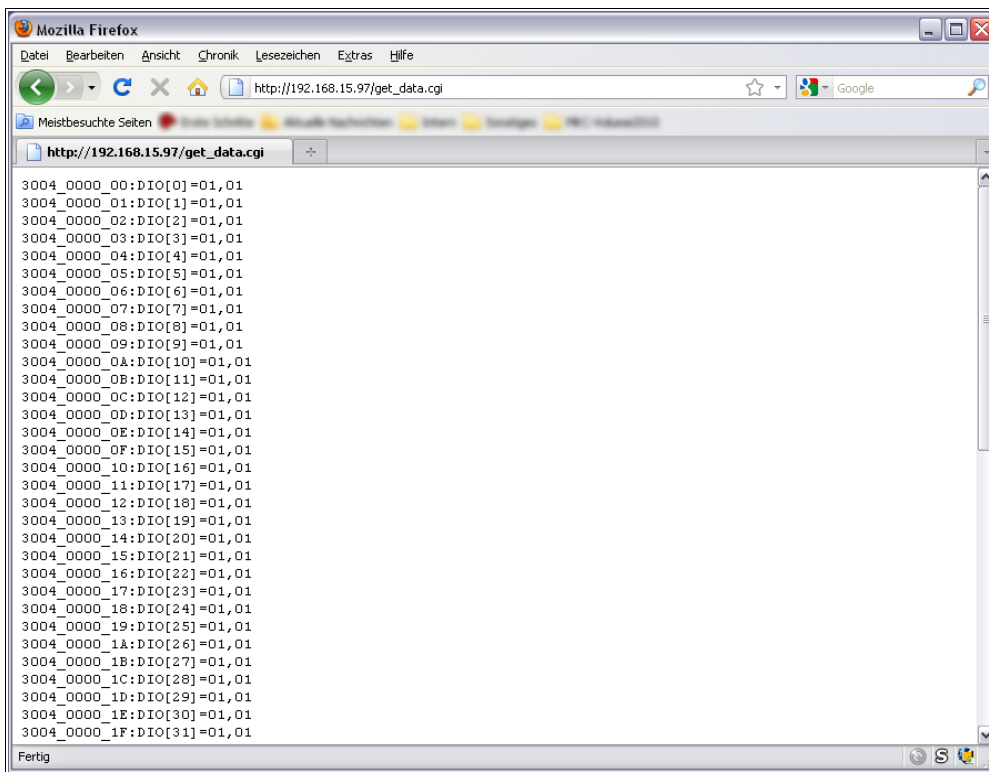


Abbildung 16: eWebTest : get\_data.cgi

Aufruf:           get\_data.cgi

Ergebnis:       3004\_0000\_00:DIO[0]=01,01     (Wert, Lese-Maske)

                  ...

Dieses Beispiel zeigt eine eWebTest. Die Ausgaben beinhalten folgende Informationen:

- ID der verwendeten Datendefinition für einen DIO : 3004,
- Die jeweilige Revision (0000) dieser Definitionen und die laufende Nummer (00, 01, 02, ...).
- Durch einen Doppelpunkt getrennt folgt der Name des entsprechende Ein- oder Ausgangs.
- Durch ein Gleichheitszeichen getrennt folgt der entsprechende Zustand oder Wert; bei einem DIO werden durch Komma getrennt Wert und Read-Maske der Struktur geliefert.

Bei dem Aufruf mit einem weiteren Parameter liefert der HTTP-Server genau einen definierten Wert:

Aufruf:           get\_data.cgi?arg=ID\_REV\_NR

Parameter:       arg=ID\_REV\_NR

                  mit     ID:     ID der verwendeten Datendefinition

                          REV:    Revision der verwendeten Datendefinition

                          NR:     Nummer des Ein/Ausganges

Ergebnis:       ID\_REV\_NR:NAM[x]=Wert

Beispiel:

Aufruf:           get\_data.cgi?arg=3004\_0000\_04

Ergebnis:       3004\_0000\_00:DIO[0]=01,01     (Wert, Lese-Maske)

In diesem Beispiel wird der aktuelle Messwert des Kanals DIO[0] zurückgegeben.

#### 4.2.2.2     get\_data\_value.cgi

'get\_data\_value.cgi' ist ein Skript, das den Query-String einer HTTP-Anforderung für das Programm 'get\_data\_dv' aufbereitet und dessen Ausgabe als 'text/plain' zurück liefert.

Die implementierte Funktion liefert bei einer HTTP-Anforderung die **Stell- und Messwerte** in einem definierten Format; pro Zeile einen Hex-Wert ohne führendes „0x“. Diese Funktion darf von allen definierten Benutzern (siehe Handbuch Benutzer) aufgerufen werden.

Bei dem Aufruf ohne Parameter liefert der HTTP-Server alle definierten Werte in ihrem aktuellen Zustand:

Aufruf:           get\_data\_value.cgi

Ergebnis:       Wert

                  Wert

                  Wert

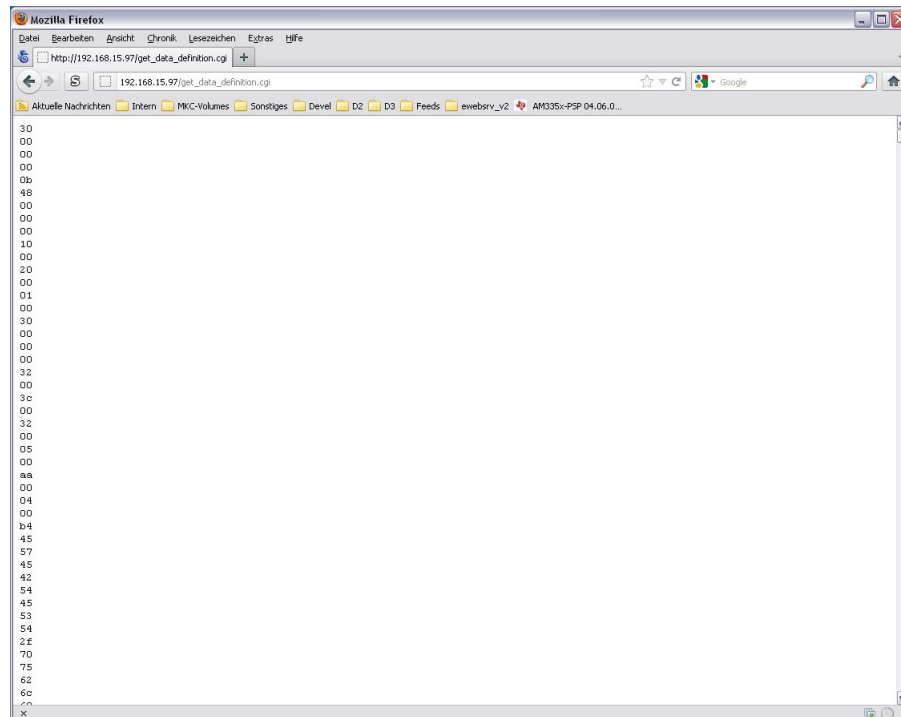
                  Wert

                  Wert

                  ...



Beispiel:



*Abbildung 18: eWebTest : get\_data\_defintion.cgi*

Aufruf:           get\_data\_definition.cgi

Ergebnis:       30  
                  00  
                  00  
                  00  
                  0b  
                  48  
                  ...

Dieses Beispiel zeigt eine eWebTest.

#### 4.2.2.4 set\_data.cgi

'set\_data.cgi' ist eine Shell-Skript, das den Query-String einer HTTP-Anforderung für das Programm 'set\_data' aufbereitet und dessen Ausgabe als 'text/plain' zurück liefert.

Die implementierte Funktion setzt den in der URL als Parameter angegebenen Stellwert. Das Format basiert hierbei auf den in get\_data.cgi übergebenen Stell- und Messwerten. Diese Funktion darf nur von den definierten Benutzern „Operator“ und Administrator“ aufgerufen werden.

Bei dem Aufruf mit den entsprechendem Parametern setzt HTTP-Server genau einen definierten Wert:

Aufruf: set\_data.cgi?arg=ID\_REV\_NR&val=Wert&mask=Schreib-Maske

Parameter: arg=ID\_REV\_NR  
&val= Trennung  
Wert der eigentliche Stellwert  
&mask= Trennung  
Schreib-Maske welche Bits bei DIO gesetzt werden sollen  
mit ID: ID der verwendeten Datendefinition  
REV: Revision der verwendeten Datendefinition  
NR: Nummer des Ein/Ausganges

Ergebnis: ID\_REV\_NR:NAM[x]=Wert

Beispiel:

Aufruf: set\_data.cgi?arg=3003\_0000\_00&val=1

Ergebnis: 3003\_0000\_00:DOT[0]=0

In diesem Beispiel wird der Stellwert des Kanals DOT[0] auf den Wert 1 gesetzt. Der Ausgang wird somit aktiv. Das Ergebnis zeigt den vorherigen Zustand des Kanals an. Bei dem Typ DIO wird zusätzlich die Lese-Maske zurück geliefert (wie beim get\_data.cgi) , um zu erkennen welche Bits gültig sind.

## 4.3 UDP-Config-Server

Wird dieser Server auf dem Gerät gestartet, kann die Netzwerkschnittstelle des **eWebSrv** über das Netzwerk (UDP-Protokoll) eingestellt werden.

Um dieses Verfahren zu verdeutlichen, kann das Beispielprogramm „**UdpConfigIpv4**“ von der **MHC** Homepage geladen und installiert werden.

Die prinzipielle Implementierung ist im Handbuch der Bibliothek „**MkcDefCommonLib**“ erläutert. Diese wird mit dem oben erwähnten Beispiel „**UdpConfigIpv4**“ installiert.

Der Server wird abhängig vom Inhalt der Variablen „**eth\_start**“ beim Booten des Systems gestartet. Diese Variable ist im Auslieferungszustand des Gerätes so gesetzt, dass der Server nicht gestartet wird. Auf welchem Port der Server arbeitet, ist in der Variablen „**udp\_config\_port**“ definiert. Diese Variable ist im Auslieferungszustand des Gerätes so gesetzt, dass der Server auf Port 49154 arbeitet.

### 4.3.1 Berechtigungen

Ob die Konfiguration durch zum Beispiel ein Programm auf dem PC möglich ist, wird durch eine Freigabe gesteuert. Auf der Seite Parameter unter dem Punkt „Netzwerk (UDP) / Freigabe“ wird festgelegt, welcher Rechner die Netzwerkeinstellungen ändern darf. Diese Einstellungen werden in der Variablen „**udp\_config\_enable**“ gespeichert. Diese ist im Auslieferungszustand so gesetzt, dass der Server jeden Verbindungsaufbau akzeptiert.

- |                                      |   |
|--------------------------------------|---|
| • 000.000.000.000                    | Nicht erlaubt.  |
| • 255.255.255.255                    | Jeder Rechner darf Stellwerte setzen.                                 |
| • 000.000.000.001 .. 255.255.255.254 | Nur der Rechner mit der eingestellten Adresse darf Stellwerte setzen. |

*Abbildung 19: Berechtigungen UDP-Config-Server*

Wird auf dem eingestellten Port ein Datenpaket empfangen, prüft der Server die eingestellte Freigabe. Entspricht die Absender-Adresse der Freigabe wird das Datenpaket angenommen.

### 4.3.2 Server Start

Der UDP-Config-Server wird über die Homepage gestartet. Grundsätzlich kann der Server auch manuell über die Konsole gestartet werden.

Aufruf: **udp\_config**

Optionen: keine

Der Config-Server versetzt sich selbstständig als Daemon in den Hintergrund.

Für den Server existiert zum Start/Stop ein Shell-Skript, das z.B. in der rc-Datei verwendet wird.

Aufruf für den Server:

`rc_mkc_config_server.sh`

### 4.4 Daten-Server (TCP)

Als Schnittstelle zu den Mess-/Stellwerten des eWebSrv's ist ein TCP-Server implementiert. Eine Änderung seiner Parameter ist per HTTP-Zugriff möglich.

Dieser Server realisiert die Übertragung von Mess-/Stellwerten über das Netzwerk mittels TCP/IP-Protokoll. Wobei die Daten in binärer Form übertragen werden. Damit kann das Gerät als Erweiterung mit externer Peripherie genutzt werden (Programm, Windows-DLL, LabVIEW, Widgets, usw.).

Um dieses Verfahren zu verdeutlichen, kann das Beispielprogramm „**TcpViewCon**“ von der **MHC** Homepage geladen und installiert werden.

Die prinzipielle Implementierung ist in den Handbüchern „**MkcDefUserLib**“/„**MkcDefCommonLib**“ erläutert. Diese wird mit dem oben erwähnten Beispiel „**TcpViewCon**“ installiert.

Der Server wird abhängig vom Inhalt der Variablen „*eth\_start*“ beim Booten des Systems gestartet. Diese Variable ist im Auslieferungszustand des Gerätes so gesetzt, dass der TCP-Server nicht startet. Auf welchem Port der Server arbeitet, wird mit der Variablen „*tcp\_data\_port*“ definiert. Diese Variable ist im Auslieferungszustand des Gerätes so gesetzt, dass der TCP-Server auf Port 49152 arbeitet.

Der Sender startet eine Übertragung, wenn sich die Mess-/Stellwerte geändert haben oder eine bestimmte Zeit – ohne Änderungen – überschritten worden ist. Diese einstellbare Zeit ist in der Variablen „*tcp\_data\_timeout*“ gespeichert. Die Variable „*tcp\_data\_timeout*“ ist im Auslieferungszustand so vorbesetzt, dass der Timeout 2 Sekunden beträgt.

#### 4.4.1 Berechtigungen

Der Server erlaubt eine Verbindung, welche durch einen zusätzlichen Parameter eingeschränkt werden kann. Hierzu wird die Variable „*tcp\_data\_enable*“ definiert. Diese ist im Auslieferungszustand so gesetzt, dass der TCP-Server jeden Verbindungsaufbau akzeptiert.

- 000.000.000.000 Keine Verbindung erlaubt.
- 255.255.255.255 Jeder darf sich mit dem TCP-Server verbinden.
- 000.000.000.001 .. 255.255.255.254 Nur der Rechner mit der in „*tcp\_data\_enable*“ gespeicherten Adresse darf sich mit dem TCP-Server verbinden.

*Abbildung 20: Berechtigungen TCP-Server*

#### 4.4.2 Server Start

Der TCP-Server für das Public-Device (/dev/data/public) wird – wie oben beschrieben - vollständig über die Homepage parametrisiert und gestartet. Für das Public-Device erfolgt der Programmaufruf daher ohne Optionen, das Programm liest seine Parameter selbständig aus dem Parameterbereich.

Zusätzlich kann jedes Application-Device mit einem TCP-Server verbunden werden; dazu müssen alle Optionen in einer Text-Datei (INI) angegeben werden. Der TCP-Server arbeitet als sog. Dämon-Prozess im Hintergrund.

```

Aufruf:      tcp_data_srv -f <INI-Datei>

Optionen:   device <Wert>           Name/Pfad des Application-Devices : /dev/data/NAME
            port <Wert>            Port Nummer des Servers
            enable_ip <Wert>       Berechtigungs-Maske : „tcp_data_enable“
            timeout <Wert>         Sendeintervall in Millisekunden; ohne Datenanforderung

```

Für die Datenserver des Public-Devices existiert zum Start/Stop ein Shell-Skript, das z.B. in der rc-Datei verwendet wird.

Aufruf für den TCP-Server:  
rc\_mkc\_data\_server.sh tcp

## 4.5 Daten-Server (UDP)

Als weitere Schnittstelle zum eWebSrv ist ein UDP-Server implementiert. Eine Änderung aller Parameter ist auch hier per HTTP-Zugriff möglich.

Das grundsätzliche Verfahren ist ähnlich dem TCP-Server, mit der Ausnahme, dass hier keine Verbindung aufgebaut wird, sondern die Daten einfach in das Netzwerk geschrieben bzw. vom Netzwerk gelesen werden.

Dieser Server realisiert die Übertragung von Mess-/Stellwerten über das Netzwerk mittels UDP/IP-Protokoll. Wobei die Daten in binärer Form übertragen werden. Damit kann das Gerät als Erweiterung mit externer Peripherie genutzt werden (Programm, Windows-DLL, LabVIEW, Widgets, usw.).

Um dieses Verfahren zu verdeutlichen, kann das Beispielpogramm „**UdpViewCon**“ von der **mhc** Homepage geladen und installiert werden. Die prinzipielle Implementierung ist in den Handbüchern „**MkcDefUserLib**“/ „**MkcDefCommonLib**“ erläutert. Diese wird mit dem oben erwähnten Beispiel „**UdpViewCon**“ installiert.

Der Server wird abhängig von Inhalt der Variablen „**eth\_start**“ beim Booten des Systems gestartet. Diese Variable ist im Auslieferungszustand des Gerätes so gesetzt, dass der Server nicht startet. Zu welchem Port die Übertragung gesendet wird, wird in der Variablen „**udp\_data\_port**“ definiert. Diese Variable ist im Auslieferungszustand des Gerätes so gesetzt, dass auf dem Port 49153 gearbeitet wird.

Der Sender überträgt an die in der Variablen „**udp\_data\_ip**“ gespeicherten Adresse als erstes die Definition der Stell- und Messdaten. Diese beschreiben die IO-Schnittstellen (Digital/Analog, Genauigkeit, Anzahl, etc.) und den Aufbau der eigentlichen Messdaten. Im Anschluss daran werden die Daten der Mess-/Stellwerte gesendet. Der Sender startet eine Übertragung, wenn sich die Mess-/Stellwerte geändert haben oder eine bestimmte Zeit – ohne Änderungen – überschritten worden ist. Diese einstellbare Zeit ist in der Variablen „**udp\_data\_timeout**“ gespeichert. Die Variable „**udp\_data\_timeout**“ ist im Auslieferungszustand so vorbesetzt, dass der Timeout 2 Sekunden beträgt.

### 4.5.1 Berechtigungen

Der Empfänger erwartet neue Stellwerte indem UDP-Pakete auf dem Netzwerk „mitgehört“ werden. Werden die UDP-Pakete an das Gerät und mit der in „**udp\_data\_port**“ gespeicherten Port-Nummer gerichtet, werden die Stellwerte interpretiert. Dies kann durch einen zusätzlichen Parameter eingeschränkt werden kann. Hierzu wird die Variable „**udp\_data\_enable**“ definiert. Diese ist im Auslieferungszustand so gesetzt, dass der Server jeden Verbindungsaufbau akzeptiert.

- |                                      |   |
|--------------------------------------|---|
| • 000.000.000.000                    | Keine Stellwerte erlaubt.   |
| • 255.255.255.255                    | Jeder darf Stellwerte senden.   |
| • 000.000.000.001 .. 255.255.255.254 | Nur der Rechner mit der in „ <b>udp_data_enable</b> “ gespeicherten Adresse darf Stellwerte senden. |

*Abbildung 21: Berechtigungen UDP-Server*

### 4.5.2 Server Start

Der UDP-Server für das Public-Device wird – wie oben beschrieben - vollständig über die Homepage parametrisiert und gestartet. Für das Public-Device erfolgt der Programmaufruf daher ohne Optionen, das Programm liest seine Parameter selbständig aus dem Parameterbereich.

Zusätzlich kann jedes Application-Device mit einem UDP-Server verbunden werden; dazu müssen alle Optionen in einer Text-Datei (INI) angegeben werden. Der UDP-Server arbeitet als sog. Dämon-Prozess im Hintergrund.

Aufruf:	<b>udp_data_srv -f &lt;INI-Datei&gt;</b>	
Optionen:	device <Wert>	Name/Pfad des Application-Devices : /dev/data/NAME
	port <Wert>	Port Nummer des Servers
	enable_ip <Wert>	Berechtigungs-Maske : „ <b>udp_data_enable</b> “
	data_ip <Wert>	Zieladresse für den Datenversand : „ <b>udp_data_ip</b> “
	timeout <Wert>	Sendintervall in Millisekunden; ohne Datenanforderung



#### 4.6.4 Start-Skript `start_tcpcomm.sh`

`start_tcpcomm.scr` realisiert die Funktionen für den Start., es modifiziert die Datei `/etc/inittab` und veranlasst somit den `init`-prozess den `TCPCOMM`-Server zu starten/stoppen.

Das Programm `tcpcomm` wird mit seinen Optionen gestartet und in den Hintergrund verlegt.

Aufruf: `start_tcpcomm.sh <Name>`

Optionen: Name der INI-Datei

#### 4.6.5 Programm `gen_tcpcomm`

Dieses Programm generiert anhand von INI-Dateien im Verzeichnis `/etc/tcpcomm.init` die `SYSFS`-Einträge unter `/sys/plattform/devices/mkc/tcpcomm`. Der Dateiname der INI-Datei legt den untergeordneten Verzeichnisnamen fest.

Beispiel :

Datei `/etc/tcpcomm.init/tcp0`

```
# Demo Server
ip=255.255.255.255      # Berechtigung, hier: Jeder darf sich mit dem Server verbinden
port=5557              # Port-Nummer des Servers
available=ek0808.1/din0 # Eingangs-Datenpunkt zur Erkennung der seriellen Gegenstelle
enable=ek0808.1/dot0  # Ausgangs-Datenpunkt zum Steuern der seriellen Gegenstelle
comm=mkcs3x0          # Device-Name der seriellen Gegenstelle
```

## 4.7 Daten-Client (TCP)

Das Programm fungiert als Daten-Client für ein externes Gerät (z.B. IONet) und stellt eine Verbindung über das TCP-Protokoll zwischen Gerät und Unit-Device her. Für das Gerät **muss** eine entsprechende INI-Datei im Verz. **/etc/unit.init** vorhanden sein um die Einbindung in das Gesamt-System zu realisieren.

Diese INI-Datei beschreibt einige Eigenschaften des Unit-Devices und beim Systemstart wird hieraus automatisch das zugehörige Unit-Device unter **/dev/data** erzeugt.

Das Format dieser INI-Datei wird in Kapitel 5.3.2.1 beschrieben.

### 4.7.1 Client Start

Zum Start des Clients müssen immer alle Optionen angegeben werden.

Aufruf: **tcp\_data\_client** -d <device> -p <port> -h <IP-Adresse>

Optionen:	device	Unit-Devices
	port	Port Nummer des TCP-Servers
	IP-Adresse	IP des TCP-Servers

Der TCP-Client arbeitet als Dämon-Prozess im Hintergrund.

Um den Start/Stop eines Clients zu vereinfachen sind verschiedene Shell-Skripte implementiert.

Start-Skript:	<b>rc_data_client.sh</b> <Unit-Device>
Stop-Skript:	<b>stop_data_client.sh</b> <Unit-Device>

Diese Skripte werten Einträge aus der INI-Datei aus, insbesondere die Angabe tcp/udp, und starten (stoppen) dementsprechend den Client.

## 4.8 Daten-Client (UDP)

Das Programm fungiert als Daten-Client für ein externes Gerät (z.B. IONet ) und stellt eine Verbindung über das UDP-Protokoll zwischen Gerät und Unit-Device her. Für das Gerät **muss** eine entsprechende INI-Datei im Verz. **/etc/unit.init** vorhanden sein um die Einbindung in das Gesamt-System zu realisieren.

Diese INI-Datei beschreibt einige Eigenschaften des Unit-Devices und beim Systemstart wird hieraus automatisch das zugehörige Unit-Device unter **/dev/data** erzeugt.

Das Format dieser INI-Datei wird in Kapitel 5.3.2.1 beschrieben.

### 4.8.1 Client Start

Zum Start des Clients müssen immer alle Optionen angegeben werden.

Aufruf: **udp\_data\_client** -d <device> -p <port> -h <IP-Adresse>

Optionen:	device	Unit-Devices
	port	Port Nummer des TCP-Servers
	IP-Adresse	IP des UDP-Servers

Der UDP-Client arbeitet als Dämon-Prozess im Hintergrund.

Um den Start/Stop eines Clients zu vereinfachen sind die Shell-Skripte, wie in Kapitel 4.7.1, zu benutzen.

## 4.9 FTP-Server

Auf dem eWebSrv ist ein FTP Server implementiert, der Dateiübertragungen von/zum eWebSrv ermöglicht. Für eine Verbindung ist eine Anmeldung mit einem in der /etc/passwd eingetragem User erforderlich. Der Server arbeitet auf Port 21.

Der Server wird abhängig vom Inhalt der Variablen „*eth\_start*“ beim Booten des Systems aktiviert. Diese Variable ist im Auslieferungszustand des Gerätes so gesetzt, dass der Server nicht startet.

Über die Parameter-Seite kann der Server mit einem HTTP-Browser aktiviert werden.

Gestartet wird der Server durch den Daemon „inetd“. Ein Shell-Skript trägt den Server in der Konfigurationsdatei /etc/inetd.conf) ein und aus.

**rcftp.sh**

## 4.10 TELNET-Server

Auf dem eWebSrv ist ein TELNET Server implementiert, der dem Anwender eine Konsole zum Betriebssystem über das Netzwerk zur Verfügung stellt. Für eine Verbindung ist eine Anmeldung mit einem in der /etc/passwd eingetragem User erforderlich. Der Server arbeitet auf Port 23.

Der Server wird abhängig vom Inhalt der Variablen „*eth\_start*“ beim Booten des Systems gestartet. Diese Variable ist im Auslieferungszustand des Gerätes so gesetzt, dass der Server nicht startet.

Über die Parameter-Seite kann der Server mit einem HTTP-Browser aktiviert werden.

Gestartet wird der Server durch den Daemon „inetd“. Ein Shell-Skript trägt den Server in der Konfigurationsdatei /etc/inetd.conf) ein und aus.

**rcctelnet.sh**

## 4.11 NTP-Client

Es besteht die Möglichkeit, den eWebSrv beim Systemstart mit der Internet Zeit zu synchronisieren. Hierzu ist ein NTP-Client implementiert. Wird dieser Client aktiviert, wird nur noch die IP-Adresse des NTP-Servers benötigt um einen Zeit-Abgleich durchzuführen.

Der Client wird abhängig vom Inhalt der Variablen „*eth\_start*“ beim Booten des Systems gestartet. Diese Variable ist im Auslieferungszustand des Gerätes so gesetzt, dass der Client nicht startet.

Über die Parameter-Seite kann der Server mit einem HTTP-Browser aktiviert werden.

Ein Shell-Skript ruft „ntpdate“ mit den passenden Parametern auf.

**set\_ntp.sh**

## 5 System-Tools

Im Folgenden werden die System-Tools beschrieben, die zusätzlich zu den Standard Linux Programmen für die eWebSrv-Plattform realisiert wurden. Diese Programme befinden sich in den Verzeichnissen /usr/bin oder /usr/sbin.

### 5.1 Mess- und Stellwerte

#### 5.1.1 get\_data

Das Programm liefert die Stell- und Messwerte in dem Format, welches auf den definierten Datenstrukturen basiert. Bei dem Aufruf ohne Parameter liefert get\_data alle definierten Werte des Public-Device in ihrem aktuellen Zustand:

Aufruf: get\_data [option] [ID\_REV\_NR] oder  
get\_data [option] [PORT [INDEX]]

Ergebnis: ID\_REV\_NR:NAM[x]=Wert

Beispiel:

Aufruf: get\_data  
Ergebnis: 3002\_0000\_00:DIN[0]=00  
... (weitere DIN's)  
oder/und  
3003\_0000\_00:DOT[0]=01  
... (weitere DOT's)  
oder/und  
3004\_0000\_00:DIO[0]=00,02  
... (weitere DIO's)  
oder/und  
3005\_0000\_00:AIN[0]=-1.938  
... (weitere AIN's)

ID der verwendeten Datendefinition für DIN (3002), DOT (3003), DIO (3004), AIN (3005), die jeweilige Revision (0000) dieser Definitionen und die laufende Nummer (00, 01, 02, ...).  
Durch einen Doppelpunkt getrennt folgt der Name des entsprechendem Ein- oder Ausganges.  
Durch ein Gleichheitszeichen getrennt folgt der entsprechende Zustand oder Wert.  
Bei einem DIO werden durch ein Komma getrennt Wert und Lese-Maske der Struktur geliefert.

Bei dem Aufruf mit dem Parameter „ID\_REV\_NR“ liefert get\_data genau einen definierten Wert:

Aufruf: get\_data ID\_REV\_NR  
Parameter: arg=ID\_REV\_NR  
mit ID: ID der verwendeten Datendefinition  
REV: Revision der verwendeten Datendefinition  
NR: Nummer des Ein/Ausganges

Ergebnis: ID\_REV\_NR:NAM[x]=Wert

Beispiel:

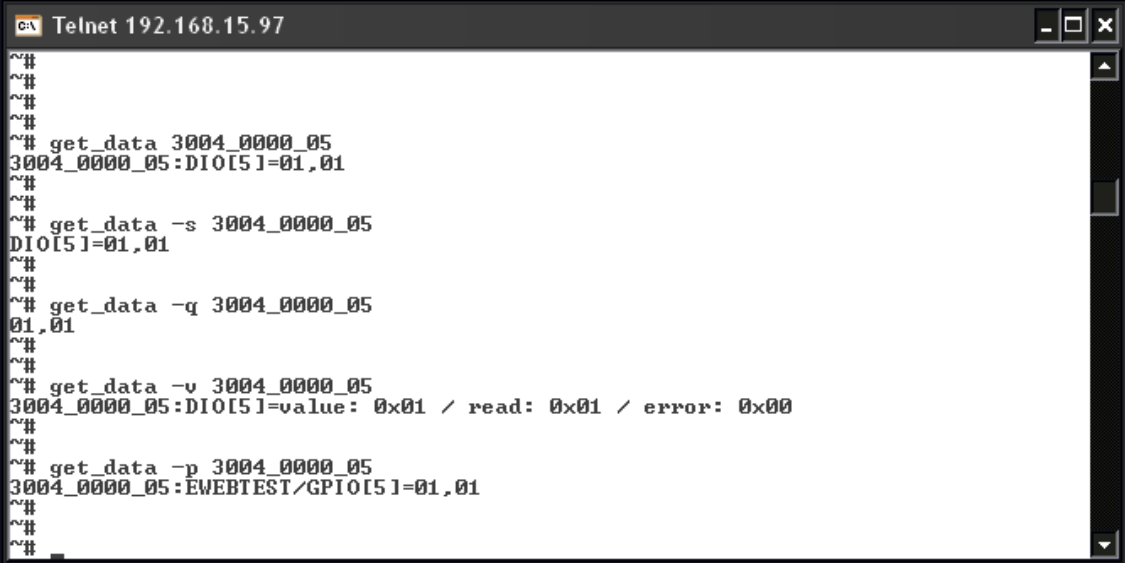
Aufruf: get\_data 3005\_0000\_04  
Ergebnis: 3005\_0000\_04:AIN[4]=1.182

Es wird der aktuelle Messwert des Kanals AIN[4] zurückgegeben.

Aufruf: get\_data ain 4 2>/dev/null liefert das gleiche Ergebnis  
(2>/dev/null : stderr Ausgabe umlenken).

Mittels „[option]“ lässt sich die Ausgabe steuern:

Beispiel:



```

c:\ Telnet 192.168.15.97
^#
^#
^#
^#
^# get_data 3004_0000_05
3004_0000_05:DIO[5]=01,01
^#
^#
^# get_data -s 3004_0000_05
DIO[5]=01,01
^#
^#
^# get_data -q 3004_0000_05
01,01
^#
^#
^# get_data -v 3004_0000_05
3004_0000_05:DIO[5]=value: 0x01 / read: 0x01 / error: 0x00
^#
^#
^# get_data -p 3004_0000_05
3004_0000_05:EWEBTEST/GPIO[5]=01,01
^#
^#

```

Abbildung 23: Beispiel `get_data`

Mit der OPTION `-d <Pfad>` wird das zu lesende Device bestimmt. Ohne diese Option wird immer das Public-Device gelesen.

Mit der OPTION `-l <Pfad>` kann ein log-file angegeben werden, in das `get_data` interne Ausgaben schreibt.

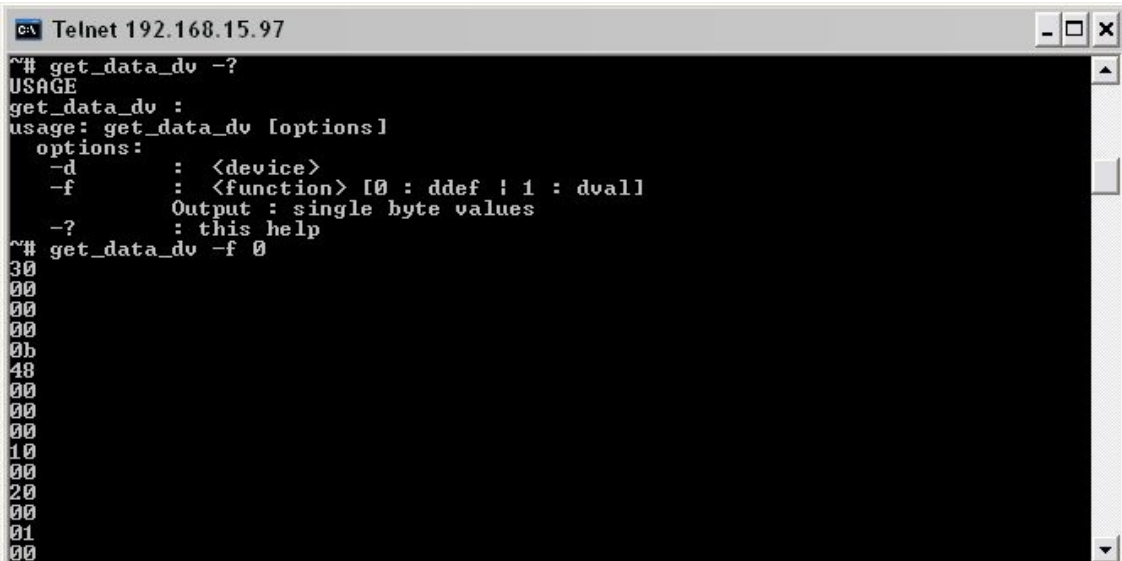
### 5.1.2 `get_data_dv`

Das Programm liefert die Stell- und Messwerte oder die Daten-Definitionen in Hex-Format ohne führendes '0x'.

Bei dem Aufruf ist der Parameter `-f` eine Pflichtangabe. Bei `-f 1` liefert `get_data` alle definierten Werte des Public-Device in ihrem aktuellen Zustand, bei `-f 0` die zugrundeliegende Daten-Definition.

Aufruf: `get_data_dv -f 0 [weitere Optionen]` oder  
`get_data_dv -f 1 [weitere Optionen]`

Beispiel:



```

c:\ Telnet 192.168.15.97
^# get_data_dv -?
USAGE
get_data_dv :
usage: get_data_dv [options]
options:
-d      : <device>
-f      : <function> [0 : ddef | 1 : dval]
        Output : single byte values
-?      : this help
^# get_data_dv -f 0
30
00
00
00
0b
48
00
00
00
10
00
20
00
01
00

```

Abbildung 24: Beispiel `get_data_dv`

### 5.1.3 set\_data

Das Programm setzt einen Stellwert.

Das Format basiert hierbei auf den in get\_data übergebenen Stell- und Messwerten.

Bei dem Aufruf mit den entsprechenden Parametern setzt set\_data genau einen definierten Wert:

```
Aufruf:      set_data [option] [ID_REV_NR] WERT [MASKE]  oder
              set_data [option] [PORT [INDEX]] [MASKE]

Ergebnis:   ID_REV_NR:NAM[x]=Wert
```

Beispiel:

```
Aufruf:      set_data 3003_0000_00 1

Ergebnis:   3003_0000_00:DOT[0]=0
```

In diesem Beispiel wird der Stellwert des Kanals DOT[0] auf den Wert 1 gesetzt. Der Ausgang wird somit aktiv. Das Ergebnis zeigt den vorherigen Zustand des Kanals an. Durch die Rückgabe des vorherigen Zustandes ist erkennbar, ob eine Änderung stattgefunden hat, d.h. der Kanal an anderer Stelle oder von einer anderen Quelle bereits verändert wurde.

Die Maske kommt bei DIO Kanälen zum Einsatz. Je nach definiertem Ausgangs-Typ (totem-pole, open-drain oder tristate) gibt es ein oder zwei setzbare Register, ein Ausgangsregister und ein enable Register. Mit der Maske ist es möglich nur ein Register zu setzen. Diese Maske entspricht dem Schreib-Eintrag in den Datenstrukturen.

Beispiel:

```
Aufruf:      set_data dio 5 6 6

Ergebnis:   3004_0000_00:DIO[5]=04,06
```

In diesem Beispiel werden Ausgangsregister und enable-Register des Kanals DIO[5] auf den Wert 1, also aktiv gesetzt. Das Ergebnis zeigt den vorherigen Zustand des Kanals an. Der Rückgabewert zeigt, dass der Kanal bereits enabled war, das Ausgangsregister inaktiv und der Pin des Kanals ebenfalls inaktiv war.

Mittels OPTION lässt sich die Ergebnis-Ausgabe steuern:

Beispiel:

```
Aufruf:      set_data [option] 3004_0000_05 2 2
mit Option:
  ohne      3004_0000_05:DIO[5]=04,06
  -s       DIO[5]=04,06
  -q       04,06
  -v       3004_0000_05:DIO[5]=value:0x04/read:0x06/write:0x00/error:0x00
  -p       3004_0000_05:eWebSrv/GPIO[9]=02,02
```

Die weiteren Optionen „-d“ und „-l“ sind analog zu der Funktionalität bei dem Programm „get\_data“.

**Hinweis:** Alle Datenpunkte der eWebTest sind im Auslieferungszustand als Eingang definiert.

## 5.2 Konfiguration und Parameter

### 5.2.1 set\_param, set\_param\_4102, set\_param\_4104

Die Programme setzen Geräte Parameter.

set\_param bedient die Basiseinstellungen, set\_param\_41xx die Systemerweiterungen (FPGA oder Trägerkarte)

set\_param setzt immer genau einen Wert im Parameter Bereich:

```
Aufruf:      set_param [option] ID_REV_NR [FUNKTION] [EIGENSCHAFT] WERT      oder
             set_param [option] FUNKTION [INDEX EIGENSCHAFT] WERT
```

```
Ergebnis:   ID_REV_NR:NAM[x]=Wert
```

Beispiel:

```
Aufruf:      set_param 4101_0000_00 i 192.168.15.97
```

```
Ergebnis:   4101_0000_00:IP=192.168.15.97
```

Die Funktionen und Eigenschaften sind die gleichen wie bei den get\_param Programmen, mit einem Unterschied. Während get\_param beim Aufruf mit dem Parameter 'eth\_s' die komplette Bitmaske liefert, ist es hier möglich alle oder auch nur einzelne Bits zu setzen.

Beispiel:

```
Aufruf:      set_param 4101_0000_00 eth_s 2 6
```

In diesem Beispiel wird der TCP Datenserver gestartet und der UDP Datenserver gestoppt.

Mittels OPTION lässt sich die Ausgabe steuern:

Beispiel:

```
Aufruf:      set_param [option] 2002_0000_05 t 1
mit Option:
ohne        2002_0000_05:DIO[5]=1
-s          DIO[5]=1
-q          1
-v          2002_0000_05:DIO[5]=DIO Port Type: 1
```

Mit der OPTION -l <Pfad> kann ein log-file angegeben werden, in das get\_data interne Ausgaben schreibt.

set\_param\_4102 setzt analog zur obigen Beschreibung die DIO Parameter der DIO Ports, die im FPGA mit der ID 4102 (FPGA IO48) realisiert sind.

Mit set\_param\_4104 werden die Parameter-Bereiche der SERC3X Schnittstellen gesetzt, die ID des FPGA ist 4104.

**Hinweis:** set\_param setzt lediglich die Werte im Parameter-Bereich der Gerätekonfiguration, die aktuellen Zustände in der Hardware oder dem System werden nicht modifiziert! Hierzu sind die entsprechenden Linux Programme aufzurufen. In den setf\_param\_xxx Skripten zum Setzen von Formulardaten über HTTP wird dieses Verfahren verwendet. Eine Ausnahme hierzu sind lediglich die digitalen und analogen Datenports, die vom Treiber intern beim Schließen des Parameter Devices aktualisiert werden.

### 5.2.2 **sserc3x**

Programm zum Setzen/Lesen der Parameter der serc3x-Schnittstelle (FPGA 4104).

Aufruf: `sserc3x [option] device`

option:

ohne	Gibt die aktiven Parameter aus
-s	FIFO Status lesen
-b baud	Baudrate setzen
-d delay	Receive Delay setzen
-x handshake	Handshake-Mode setzen

Ergebnis: `Baudrate=Wert; Delay=Wert`

Ergebnis: `Status=Wert; Baudrate=Wert; Delay=Wert`

### 5.2.3 **fpga\_id.sh, fpga\_rev.sh**

Liefert die ID bzw. die Revision des FPGA auf dem eWebSrv am stdout Device.

## 5.2.4 rstout, rstout.sh

Setzt und liest den RSTOUT Pin des FPGA.

Beide Programme arbeiten gleich, rstout ist eine Binary-Programm, rstout.sh ist eine Shell-Skript.

Aufruf:           rstout [Option] [Parameter]

Option:           -q : keine Ausgaben (quiet)  
                  -v : ausführliche Ausgaben (verbose)  
                  -? : Hilfe

Parameter:       '0' , 'aus' oder 'off':       RSTOUT inaktiv  
                  '1' , 'an' oder 'on':       RSTOUT aktiv  
                  kein Parameter:           aktueller Zustand wird ausgegeben

Ergebnis:       der aktuelle Zustand wird als '0' oder '1' ausgegeben

## 5.2.5 rstphy, rstphy.sh

Setzt und liest den RSTPHY Pin des FPGA.

Beide Programme arbeiten gleich, rstphy ist eine Binary-Programm, rstphy.sh ist eine Shell-Skript.

Aufruf:           rstphy [Option] [Parameter]

Option:           -q : keine Ausgaben (quiet)  
                  -v : ausführliche Ausgaben (verbose)  
                  -? : Hilfe

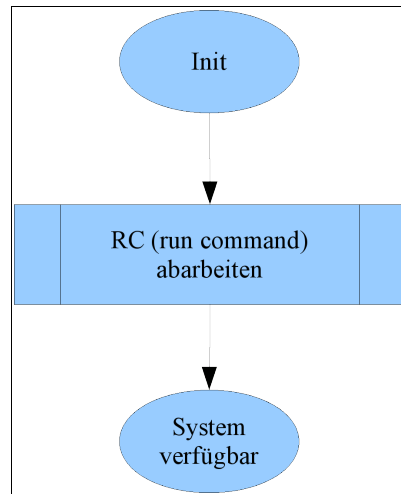
Parameter:       '0' , 'aus' oder 'off':       RSTPHY inaktiv  
                  '1' , 'an' oder 'on':       RSTPHY aktiv  
                  kein Parameter:           aktueller Zustand wird ausgegeben

Ergebnis:       der aktuelle Zustand wird als '0' oder '1' ausgegeben

**Warnung:** Das Setzen (Aktivieren) von RSTPHY im laufenden Betrieb kann zu Instabilitäten führen (Verlust der Netzwerk-Verbindung).

### 5.3 System-Initialisierung

Während des Systemstarts wird die plattformbezogene RC (run-command) durchlaufen. Dabei handelt es sich um eine Systemdatei in der verschiedene Dienste und Voreinstellungen für den Betrieb gestartet oder gesetzt werden.



*Abbildung 25: System-Start*

**Warnung:** Veränderungen an dieser Datei können zu einem eingeschränkten bis nicht mehr lauffähigen System führen. Es ist dringend angeraten von dieser Datei eine Sicherheitskopie zu erstellen bevor Änderungen vorgenommen werden

### 5.3.1 Systemstart eWebTest

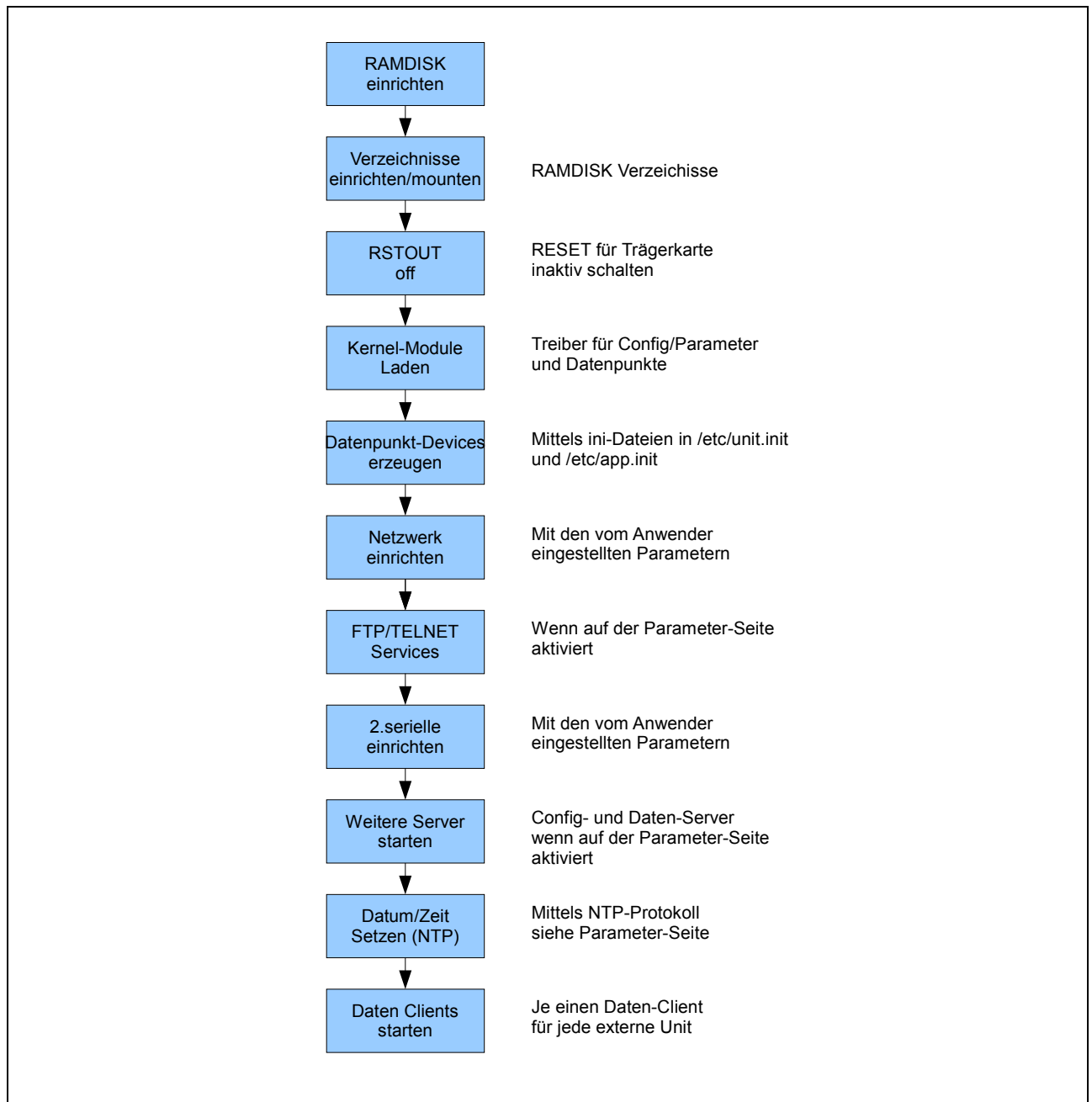


Abbildung 26: /etc/rc eWebTest

### 5.3.2 gen\_devs

Das Programm baut beim Systemstart alle Unit- und Application-Devices des Systems und deren Verknüpfungen untereinander auf.

Die Device-Deskriptoren in /dev/data werden erzeugt.

Das Programm wird in /etc/rc (runcommand) gestartet und bedient sich dabei mehrerer Konfigurationsdateien.

Für jedes Device gibt es eine Konfigurationsdatei. Die Dateien für Unit-Devices befinden sich in /etc/unit.init, die für Application-Devices in /etc/app.init.

Zu beachten ist, dass mit Abschluss dieses Programms die externen Devices mit Default-Werten aus den Dateien in /etc/default besetzt sind. Durch die Kommunikation mit einem externen Gerät können sich Werte und insbesondere die Terminal-Namen ändern.

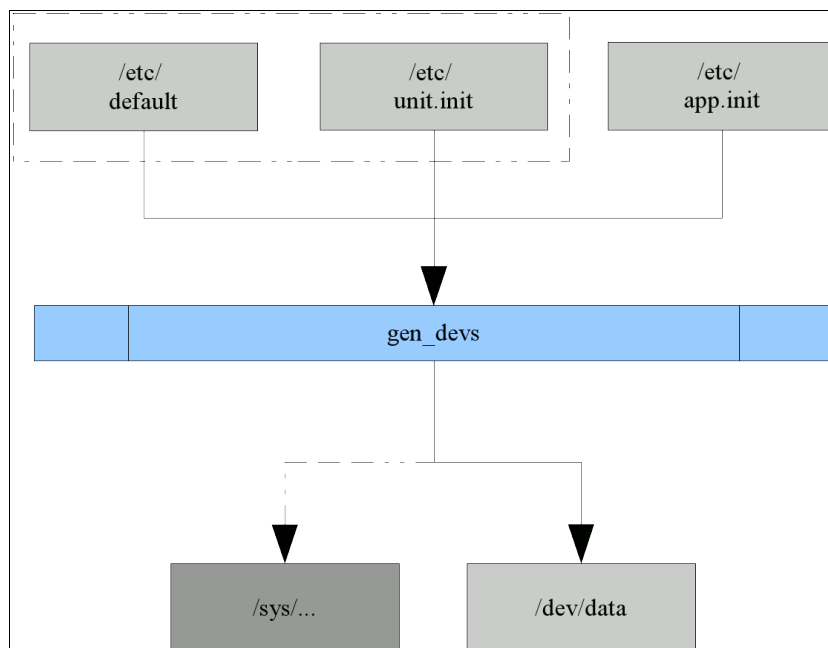


Abbildung 27: Programm gen\_devs

### 5.3.2.1 Aufbau der Konfigurationsdateien

Die Konfigurationsdateien werden zeilenweise abgearbeitet.

Ein '#' leitet einen Kommentar bis zum Zeilenende ein. Leere Zeilen werden ignoriert.

Whitespace (Blank oder Tab) werden ignoriert.

Ein Parametereintrag hat (mit einer Ausnahme) immer die Form „Name=Wert“.

Der Name der Konfigurationsdatei ist, mit einer Ausnahme, der Name des Devices. Die Konfigurationsdatei „LOCAL“ (in Großbuchstaben) bezeichnet das lokale Unit-Device 'Unit0', daher wird der Name des Devices aus dem CONFIG-Bereich übernommen.

Für **alle Devices** sind folgende Parameter definiert:

- owner=
- group=
- mode=

Für die Devices LOCAL und Application-Device public (Public-Device) sind **nur** diese Parameter definiert.

Sie bestimmen die Zugriffsrechte der Datei im Device Verzeichnis (/dev/data). Zur Zeit sind hier nur numerische Werte gültig, die den Einträgen in den Dateien /etc/passwd bzw. /etc/group entsprechen. Werden diese Parameter nicht gesetzt, wird allen Usern volles Zugriffsrecht gewährt.

#### Parameter Unit-Devices

- private=
- ip=
- port=
- protocol=
- *File-Name*

**private** bestimmt, ob Datenpunkte dieser Unit dem Public-Device zugeordnet werden dürfen. Ist **private=1**, werden Datenpunkte dieses Devices nicht im Public-Device zugeordnet.

**ip** und **port** definieren die entsprechenden Parameter des verbundenen externen Gerätes.

**protocol** definiert das Netzwerk Protokoll TCP oder UDP.

**File-Name** ist der Name des Binary-File mit der default data-definition des externen Gerätes, das an dieser ip/port Kombination angebunden werden soll. Es können nur bekannte Geräte verbunden werden. Die Beschreibungsdateien befinden sich mit ihrer Verkaufsbezeichnung im Verzeichnis /etc/default.

#### Parameter Application-Devices

- din=
- dot=
- dio=
- ain=
- aot=

Hier werden die Zuordnungen 'virtueller' Ports von Application-Devices zu 'physikalischen' Ports von Unit-Devices getroffen, und zwar in der Form:

*din=unit-device/port*

Die (zeilenweise) Reihenfolge der Definitionen bestimmt hierbei den Index des Ports im Application-Device.

Es können immer nur gleiche Datenpunkt Typen zugewiesen werden, die Zuweisung DIN=unit:DIO4 ist nicht zulässig.

Alle Unit-Ports, die nicht auf diese Art einer Applikation zugewiesen wurden, und die nicht als private gekennzeichnet sind, werden dem Public-Device zugeordnet. Zuerst werden die Ports des LOCAL Devices eingetragen, danach die anderen Units in alphabetischer Reihenfolge.

**BEISPIEL:**Datei /etc/unit\_init/**LOCAL:**

```
owner=0          # user root
group=0         # group root
mode=0644
```

Datei /etc/unit\_init/**ionet.0**

```
owner=0
group=0
mode=0644       # alle können lesen, nur owner schreiben
private=0       # Datenpunkte für public zugelassen
ip=192.168.15.99
port=5777
protocol=tcp
IONET-4 aebe    # Datenpunkte
```

Datei /etc/unit\_init/**ionet.1**

```
owner=0
group=0
mode=0644       # alle können lesen, nur owner schreiben
private=1       # Datenpunkte erscheinen nicht im public
ip=192.168.15.87
port=5766
protocol=udp
IONET-2 ag--    # Datenpunkte
```

Datei /etc/app\_init/**public**

```
owner=0         # root
group=0         # root
mode=0666       # jeder kann lesen und schreiben
```

Datei /etc/app\_init/**jtag**

```
owner=501       # irgendein user
group=100       # irgendeine group
mode=0600       # nur owner kann lesen und schreiben
dio=LOCAL/dio0 # dio0 ist dio0 des eWebSrv
dio=LOCAL/dio1 # dio1 ist dio1 des eWebSrv
dio=LOCAL/dio2 # dio2 ist dio2 des eWebSrv
dio=LOCAL/dio3 # dio3 ist dio3 des eWebSrv
```

Datei /etc/app\_init/**app.0**

```
owner=502
group=101
mode=0600       # nur owner kann lesen und schreiben
dio=LOCAL/dio6 # dio0 ist dio6 des eWebSrv
dio=LOCAL/dio4 # dio1 ist dio4 des eWebSrv
din=ionet.0/din1 # din0 ist din1 des externen device ionet.0
din=ionet.0/din0 # din1 ist din0 des externen device ionet.0
dot=ionet.0/dot0 # dot0 ist dot0 des externen device ionet.0
ain=ionet.0/ain2 # ain0 ist ain2 des externen device ionet.0
```

Datei /etc/app\_init/**slave**

```
owner=503
group=102
mode=0600       # nur owner kann lesen und schreiben
din=LOCAL/din0 # din0 ist din0 des eWebSrv
dot=LOCAL/dot0 # dot0 ist dot0 des eWebSrv
din=ionet.0/din6 # din1 ist din6 des externen device ionet.0
dot=ionet.0/dot0 # dot1 ist dot0 des externen device ionet.0
din=ionet.1/din0 # din2 ist din0 des externen device ionet.1
dot=ionet.1/dot0 # dot2 ist dot0 des externen device ionet.1
aot=ionet.1/aot1 # aot00 ist aot1 des externen device ionet.1
```

Im Verzeichnis /dev/data werden folgende Dateien erzeugt:

- **eWebTest** (das Unit-Device „LOCAL“ der eWebTest)
- **public**
- ionet.0
- ionet.1
- jtag
- app.0
- slave

**Hinweis:** Für jede Datei wird jeweils ein Application-Device oder Unit-Device angelegt. Aus diesem Grunde dürfen sich auch keine Sicherheitskopien von INI-Dateien im Verzeichnis „/etc/unit\_init“ bzw. „/etc/app\_init“ befinden.

## 6 ANHANG

### 6.1 Definitionen

#### 6.1.1 Allgemein

Die allgemeinen Software-Definitionen für **MHC** befinden sich in den Dokumenten „MKCDefUserLib“ und „MKCDefCommonLib“.

### 6.2 Auslieferungszustand

Sämtliche Einstellungen für den Betrieb des **eWebSrv** erfolgen über das „Web Management Interface“. Im Auslieferungszustand wird der **eWebSrv** mit folgenden Parametern geliefert:

IP-Adresse:	192.168.015.097	kann mit dem Konfigurationsassistenten oder “zu Fuß” adaptiert werden,
IP-Maske:	255.255.255.000	siehe hierzu das Handbuch “Inbetriebnahme”.

Es ist kein Router/Gateway eingetragen, das Netzwerk wird im Auto Negotiation Modus betrieben.  
Der TCP-Server und der UDP-Server ist deaktiviert.  
Der UDP-Config-Server ist deaktiviert.  
FTP- und TELNET Server sind deaktiviert.  
HTTP-Server ist aktiv.

Sämtliche GPIO sind als Eingang definiert.  
Serielle Konsole 19200 Baud, 8 Bit, keine Parität, 1 Stopbit.

### 6.3 Rechtliche Bestimmungen

#### 6.3.1 Lizenzierung

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