

Gebrauchsanleitung

PLMaster / WEB

GA-E035

ETHERNET Modul

14597a



Inhaltsverzeichnis

1 Einführung	3
2 Optionsspezifikation	3
2.1 Technische Beschreibung	3
2.2 Eigenschaften	3
3 Installation	4
3.1 Einbau in den PLMaster	4
3.2 Anschluß an ein Netzwerk	4
4 Konfiguration	5
4.1 Aufbau der Konfigurationsparameter	5
4.2 Parameter der Protokollebene	5
4.2.1 IP-Adresse	5
4.2.2 Netzmaske	5
4.2.3 Gateway / Router	5
4.2.4 Namensdienste	6
4.2.5 Mail Host	6
4.3 Email Parameter	6
4.3.1 Authentifizierung	6
4.3.1 Adressentabelle	6
4.4 Ereignismeldungen / Aktionen	6
4.4.1 Hauptempfänger	7
4.4.2 Kopieempfänger	7
4.4.3 Ereignisquelle	7
4.4.4 Ereignistyp	8
4.5 Bericht / Report	8
4.5.1 Hauptempfänger	8
4.5.2 Kopieempfänger	8
5 Webserver	9
6 Die Programmierschnittstelle	13
6.1 Eingangsdaten	13
6.2 Ausgangsdaten	13
6.3 Client-Server Kommunikation	13
7 Aufbau der Datenstrukturen	14
7.1 Allgemeiner Datenpaket Aufbau	14
7.2 Das Request Paket (Client)	14
7.2.1 Request Identifikation	14
7.3 Das Response Paket (Server)	15
7.3.1 Response Identifikation	15
7.3.2 Fehler Codes	15
7.4 Detaillierter Datenpaket Aufbau	15
7.4.1 Operation: Lesen der aktuellen Meßdaten	15
7.4.2 Operation: Lesen der Konfigurationsdaten	19
Anhang A, Beispiel eines Client Programms	25

1 Einführung

Die vorliegende Dokumentation beschreibt die technischen Fähigkeiten, die Konfiguration, die Bedienung und die Programmierung der Anwenderschnittstelle der PLMaster Option PLMaster / WEB (Ethernet Modul). Weiterhin wird der Aufbau der über Ethernet zur Verfügung gestellten Meßdaten erläutert.

Um das PLMaster System über dieses Ethernet Modul mit anderen Rechnersystemen zur Datenübermittlung zu koppeln, sind Kenntnisse und Grundlagen bezüglich Ethernet, Ethernet Installationstechnik, TCP/IP und UNIX- bzw. Windows-Socket Programmierung notwendig. Diese werden als bekannt vorausgesetzt und sind nicht Bestandteil dieser Dokumentation.

2 Optionsspezifikation

2.1 Technische Beschreibung

Die Option PLMaster / WEB stellt eine Ethernet LAN Schnittstelle nach dem Standard IEEE802.3 zur Verfügung mit einer Datenrate von 10 Mbps (10BASE-T).

Die Software der PLMaster / WEB umfasst folgende Bestandteile:

- grundlegende Protokolle wie UDP, TCP, HTTP, SMTP
- einen Webserver für die Online Datenvisualisierung über jeden PC-üblichen Webbrowser wie Microsoft Internet Explorer oder Netscape Navigator. Dies ist sowohl im lokalen Netzwerk als auch über das Internet möglich. Die Visualisierung der Daten erfolgt grafisch mit Hilfe von Java Applets, die von PLMaster / WEB zum Browser übertragen werden, dadurch ist keine zusätzliche Software auf dem PC erforderlich
- einen Ereignisserver zum automatischen Versenden von elektronischer Post als Email oder SMS (kurze Textmeldungen an ein Mobiltelefon) beim Auftreten von konfigurierbaren Ereignissen
- einen Datenserver, welcher über eine TCP/IP Verbindung Client-Anfragen entgegennimmt und beantwortet. Die zur Verfügung gestellten Daten werden in Pakete definierter Länge mit definiertem Aufbau verpackt. Bei den Daten handelt es sich um aktuelle Meßgrößen der am PLMaster angeschlossenen Sensoren, um Statusflags, Zählerstände sowie Konfigurationsdaten.

2.2 Eigenschaften

Hardware: Ethernet Schnittstelle nach IEEE802.3, 10BASE-T (Twisted Pair).

Software: Webserver, Datenserver, Ereignisserver, SMTP Client, unterstützte Protokolle UDP, TCP, HTTP, SMTP.

3 Installation

3.1 Einbau in den PLMaster

Der PLMaster wird mit werkseitig bereits eingebauter Option PLMaster / WEB geliefert.

3.2 Anschluß an ein Netzwerk

Der Anschluss an ein vorhandenes LAN sowie eine eventuell vorzunehmende Konfiguration des LAN entsprechen internationalem Standard für Ethernet Netzwerke. Soll Zugang zum Internet möglich sein (z.B. um Email zu versenden), so muß ein Router bzw. ein Gateway im lokalen Netzwerk vorhanden und entsprechend konfiguriert sein.

Die Beschreibung der Konfiguration von Ethernet Netzwerken, Routern und Gateways ist nicht Bestandteil dieser Dokumentation.

4 Konfiguration

Die Konfiguration der Option PLMaster / WEB sowie der einzelnen Softwarepakete ist relativ komplex und sollte mit dem PC-gestützten PLMaster Konfigurationsprogramm durchgeführt werden.

Alle Konfigurationsparameter befinden sich in einer Datei auf der ATA Flash Speicherkarte des PLMaster Gerätes. Diese Datei enthält sowohl die Konfiguration des PLMaster selbst als auch die der PLMaster / WEB Option und ist recht umfangreich. Zum Verständnis der einzelnen Parameter sind diese nachfolgend aufgeführt und erläutert, soweit sie die Option PLMaster / WEB betreffen.

4.1 Aufbau der Konfigurationsparameter

Alle PLMaster Konfigurationsparameter setzen sich zusammen aus 5 dreistelligen Nummernblöcken und einem Erläuterungstext. Die Nummernblöcke und der Text sind durch Unterstriche verbunden. Unmittelbar anschließend folgt ein Gleichheitszeichen. Alles, was rechts vom Gleichheitszeichen steht, entspricht dem Parameterwert. Die Nummernblöcke identifizieren den Parameter eindeutig, wobei der Erläuterungstext nur als Kommentar zur besseren Lesbarkeit dient.

Beispiel:

```
100_315_600_001_003_IPADDRESS=172.20.100.100
```

Dieser Parameter legt z.B. die IP Adresse der PLMaster / WEB Option fest, sie steht rechts vom Gleichheitszeichen bis zum Zeilenende.

Der erste Nummernblock, hier „100“, bezeichnet die Jean Müller Produktfamilie (PLMaster). Der zweite Nummernblock, hier „315“, kennzeichnet die Versionsnummer des Parameters. Der dritte Nummernblock kennzeichnet die Parametergruppe innerhalb der Konfigurationsdatei. Die letzten 2 Nummernblöcke dienen zur eindeutigen Identifikation des Parameters.

4.2 Parameter der Protokollebene

Zu dieser Gruppe zählen alle Parameter, die zum Grundbetrieb des Netzwerks notwendig sind.

4.2.1 IP-Adresse

Beispiel:

```
100_315_600_001_003_IPADDRESS=172.20.100.100
```

Bestimmt die IP Adresse dieses PLMaster Gerätes innerhalb des lokalen Netzwerks. Dieser Parameter ist zum korrekten Betrieb notwendig.

4.2.2 Netzmaske

Beispiel:

```
100_315_600_001_004_NETMASK=255.255.0.0
```

Bestimmt die Netzmaske zur IP Adresse. Dieser Parameter ist zum korrekten Betrieb notwendig.

4.2.3 Gateway / Router

Beispiele:

```
100_315_600_001_005_GATEWAY=172.20.19.99
```

oder

```
100_315_600_001_005_GATEWAY=
```

Legt die IP Adresse eines Gateways oder Routers fest um z.B. ins Internet oder in ein anderes Netzwerk zu gelangen. Dieser Parameter ist optional. Falls kein Gateway vorhanden ist, endet die Zeile nach dem Gleichheitszeichen.

4.2.4 Namensdienste

Beispiel:

```
100_315_600_001_006_NAMESERVA=194.25.2.129
100_315_600_001_007_NAMESERVB=
100_315_600_001_008_NAMESERVC=
```

Legt die IP Adressen von bis zu 3 sogenannten Nameservern fest. Diese sind notwendig, wenn mit symbolischen IP Adressen wie „mail.online.de“ gearbeitet wird, ansonsten optional. Falls kein Nameserver vorhanden ist, endet die Zeile nach dem Gleichheitszeichen.

4.2.5 Mail Host

Beispiel:

```
100_315_600_001_011_SMTPHOST=mail.onlinehome.de
```

Legt den symbolischen Namen des Servers fest, der eingehende Email nach dem SMTP Protokoll entgegennimmt. Dieser Parameter ist notwendig, falls automatisch Reports oder Ereignismeldungen als Email verschickt werden sollen, ansonsten optional. Falls kein SMTP Host vorhanden ist, endet die Zeile nach dem Gleichheitszeichen.

4.3 Email Parameter

Zu dieser Gruppe zählen Parameter, die zusätzlich zu denjenigen der Protokollebene zum Versenden von Email benötigt werden.

4.3.1 Authentifizierung

Beispiel:

```
100_315_500_001_001_AUTH=rl@rl-systemintegration.de
```

Legt den Authentifizierungsstring für die ausgehende Email fest (sozusagen den „Absender“). Dieser Parameter ist Provider-abhängig. Es kann eine zulässige Email Adresse sein, eine beliebige Zeichenfolge oder überhaupt nicht benötigt werden.

4.3.1 Adressentabelle

Beispiel:

```
100_315_500_001_002_ADDR1=benutzername@online.de
100_315_500_001_003_ADDR2=01771234567@smsmail.eplus.de
100_315_500_001_004_ADDR3=benutzername@d2mail.de
100_315_500_001_005_ADDR4=01701234567@T-D1-SMS.de
100_315_500_001_006_ADDR5=rl@rl-systemintegration.de
100_315_500_001_008_ADDR6=01731234567@d2-message.de
100_315_500_001_007_ADDR7=
100_315_500_001_009_ADDR8=
100_315_500_001_010_ADDR9=
100_315_500_001_011_ADDR10=
```

Legt die Empfänger Email Adressen für Reports und Ereignismeldungen fest. Bis zu 10 Adressen können vergeben werden. Dies können „normale“ Email Adressen sein, oder spezielle, von den Telefonprovidern zur Verfügung gestellte Adressen, um eine Email als SMS (kurze Textnachricht) zu einem Mobiltelefon zu senden. Diese speziellen Adressen enthalten oft die Telefonnummer des Mobiltelefons. Falls automatisch Reports oder Ereignismeldungen als Email verschickt werden sollen, muß mindestens eine Adresse konfiguriert werden.

4.4 Ereignismeldungen / Aktionen

Diese Parametergruppe spezifiziert bis zu 100 sogenannte Aktionen, d.h. auslösende Ereignisse und die zugehörigen Reaktionen, die im Absenden von Emails bzw. SMS bestehen. Eine Aktion wird durch jeweils 4 zusammengehörende Parameter beschrieben: Hauptempfänger, Kopieempfänger, Ereignisquelle und Ereignistyp.

4.4.1 Hauptempfänger

Beispiele:

`100_315_510_001_001_ACMAILTO=2`

oder

`100_315_510_002_001_ACMAILTO=0`

Der Hauptempfänger für die Email bzw. SMS wird aus Platzgründen und um Redundanz zu vermeiden als Index in die Adressentabelle von Kapitel 4.3.1 gespeichert.

Dies bedeutet im obigen Beispiel:

Die Empfängeradresse `100_315_510_001_001_ACMAILTO=2`

bezieht sich auf die konfigurierte Emailadresse 2:

`100_315_500_001_003_ADDR2=01771234567@smsmail.eplus.de`

4.4.2 Kopieempfänger

Beispiel:

`100_315_510_001_002_ACMAILCC=1`

oder

`100_315_510_002_002_ACMAILCC=0`

Jede gesendete Email kann auf Wunsch zusätzlich in Kopie einem zweiten Empfänger zugestellt werden. Dieser Parameter wirkt sich nur dann aus, wenn auch ein entsprechender Hauptempfänger konfiguriert ist.

Der Kopieempfänger für die Email bzw. SMS wird aus Platzgründen und um Redundanz zu vermeiden als Index in die Adressentabelle von Kapitel 4.3.1 gespeichert.

Dies bedeutet im obigen Beispiel:

Die Empfängeradresse `100_315_510_001_002_ACMAILCC=1`

bezieht sich auf die konfigurierte Emailadresse 1:

`100_315_500_001_002_ADDR1=benutzername@online.de`

4.4.3 Ereignisquelle

Beispiel:

`100_315_510_001_003_ACSOURCE=5`

oder

`100_315_510_002_003_ACSOURCE=6`

Dieser Parameter legt denjenigen Sensor am PLMaster Gerät fest, der auf Ereignisse überwacht werden soll. Angegeben wird die laufende Nummer des Sensors, zwischen 1 und 60. Wird hier eine Null eingetragen, findet keine Überwachung statt.

4.4.4 Ereignistyp

Beispiel:

100_315_510_001_004_ACEVTYPE=3

oder

100_315_510_002_004_ACEVTYPE=5

Dieser Parameter legt den Ereignistyp, der an einem Sensor eines PLMaster Gerätes überwacht werden soll, fest.

Die Definition der Ereignistypen ist wie folgt:

1 ---> Einschaltvorgang

2 ---> Ausschaltvorgang

3 ---> Sicherungsfall

4 ---> Sicherung OK

5 ---> Überstrom

6 ---> Überstrom OK

7 ---> Unterstrom

8 ---> Unterstrom OK

9 ---> Überspannung

10 ---> Überspannung OK

11 ---> Unterspannung

12 ---> Unterspannung OK

13.---> Leistungsschalter ausgelöst

14 ---> Leistungsschalter thermisch ausgelöst

15 ---> Leistungsschalter magnetisch ausgelöst

16 ---> Leistungsschalter EIN

4.5 Bericht / Report

Diese Parametergruppe konfiguriert die Reportfunktion, die täglich automatisch einen Bericht als Email versenden kann. Der Versand erfolgt dabei immer kurz nach 24.00 Uhr.

4.5.1 Hauptempfänger

Beispiel:

100_315_520_001_001_REPMAILTO=1

Der Hauptempfänger für die Email wird aus Platzgründen und um Redundanz zu vermeiden als Index in die Adressentabelle von Kapitel 4.3.1 gespeichert.

Dies bedeutet im obigen Beispiel:

Die Empfängeradresse *100_315_520_001_001_REPMAILTO=1*

bezieht sich auf die konfigurierte Emailadresse 1:

100_315_500_001_002_ADDR1=benutzername@online.de

4.5.2 Kopieempfänger

Beispiel:

100_315_520_001_002_REPMAILCC=2

oder

100_315_520_001_002_REPMAILCC=0

Jede gesendete Email kann auf Wunsch zusätzlich in Kopie einem zweiten Empfänger zugestellt werden. Dieser Parameter wirkt sich nur dann aus, wenn auch ein entsprechender Hauptempfänger konfiguriert ist.

Der Kopieempfänger für die Email wird aus Platzgründen und um Redundanz zu vermeiden als Index in die Adressentabelle von Kapitel 4.3.1 gespeichert.

Dies bedeutet im obigen Beispiel:

Die Empfängeradresse *100_315_520_001_002_REPMAILCC=2*

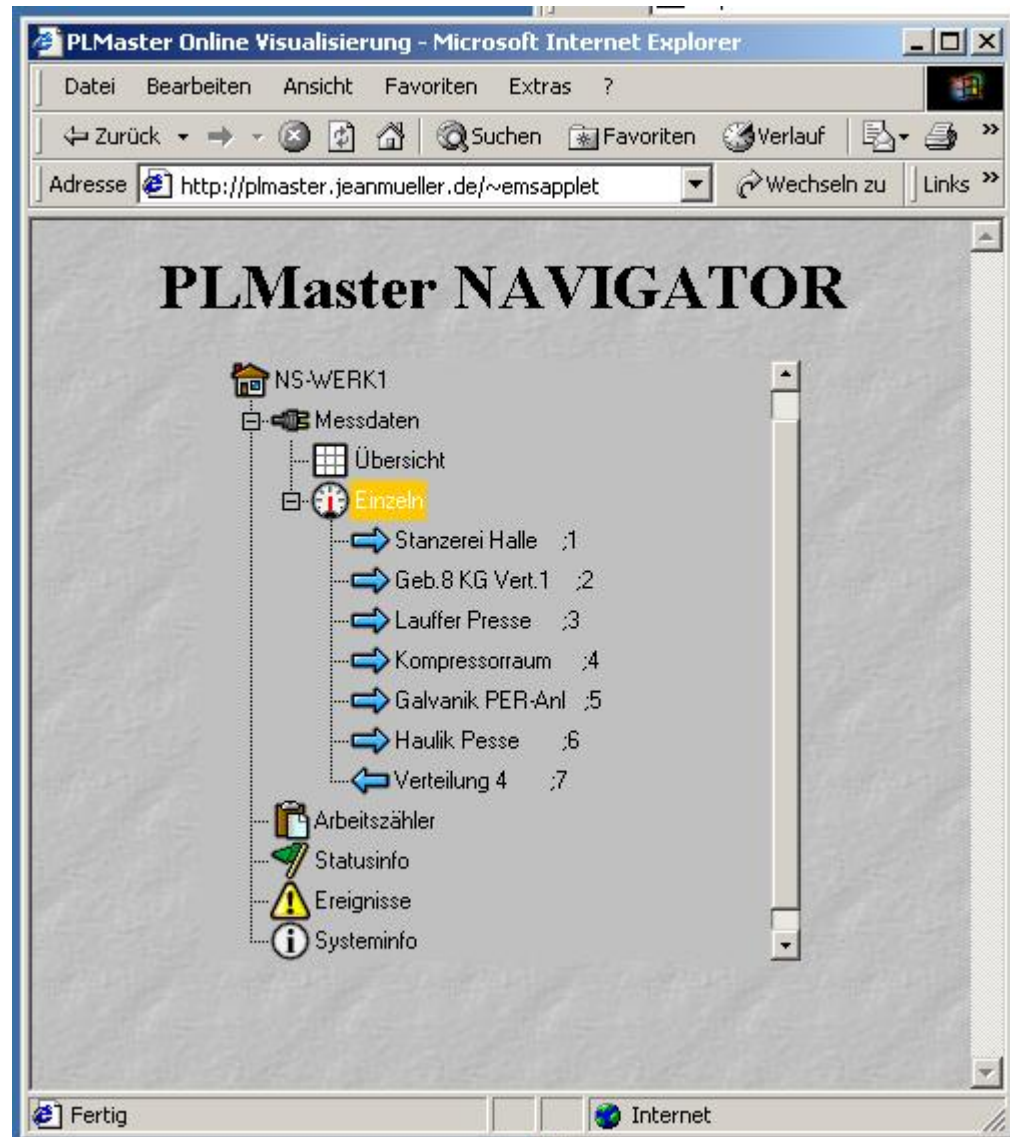
bezieht sich auf die konfigurierte Emailadresse 2:

100_315_500_001_003_ADDR2=01771234567@smsmail.eplus.de

5 Webserver

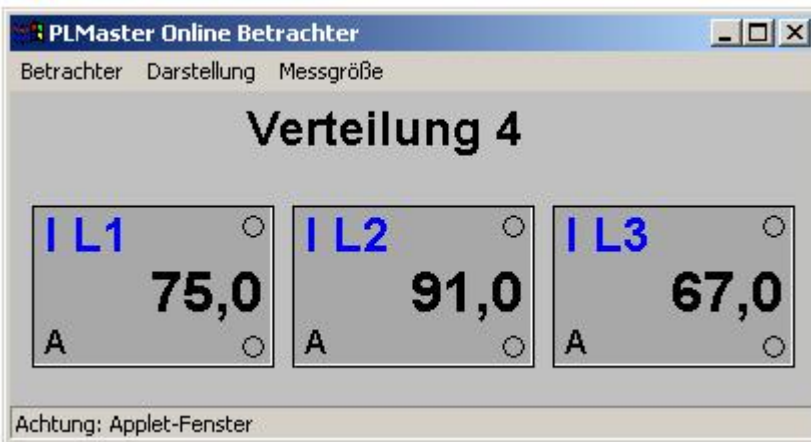
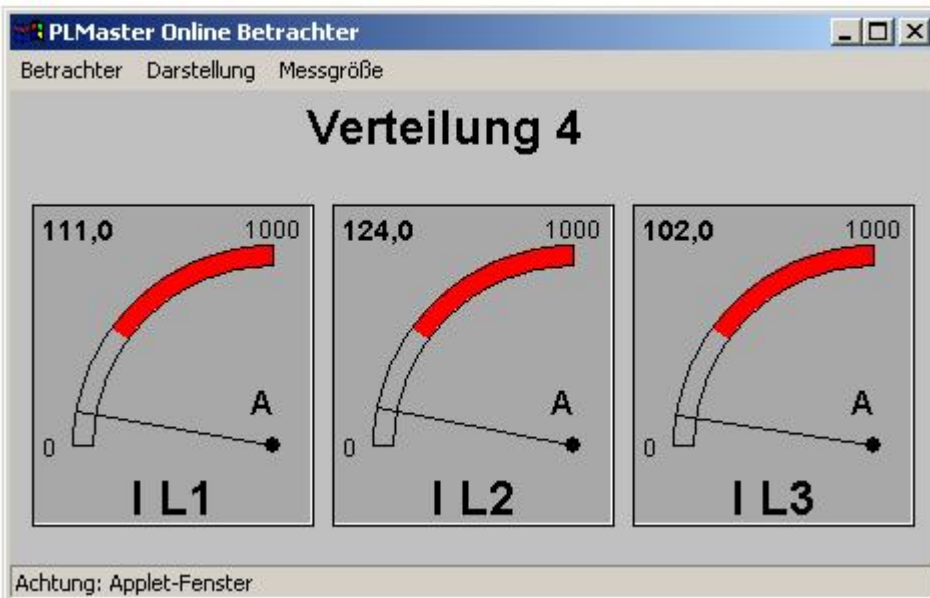
Die Software der Option PLMaster / WEB enthält einen Webserver plus spezielle JAVA Applets, die jeden Web-Browser wie z.B. Microsoft Internet Explorer oder Netscape Navigator in ein Online-Visualisierungsprogramm verwandeln. Die Funktion steht nach erfolgter Konfiguration der PLMaster / WEB und des Browsers sofort zur Verfügung.

Nach dem Verbindungsaufbau und der Initialisierung des Systems wird das PLMaster System übersichtlich in einer Baumstruktur dargestellt. Nicht angeschlossene bzw. nicht konfigurierte Sensoren werden in der Darstellung unterdrückt. Durch einfachen Mausklick ist direkter Zugriff auf die aktuellen Meßdaten, Zählerstände, Ereignisse und Statusinformationen möglich.



Die Darstellung der Informationen kann sowohl grafisch in Form von Analoginstrumenten, Digitalinstrumenten, Balken und Kurvenschreibern als auch tabellarisch erfolgen.

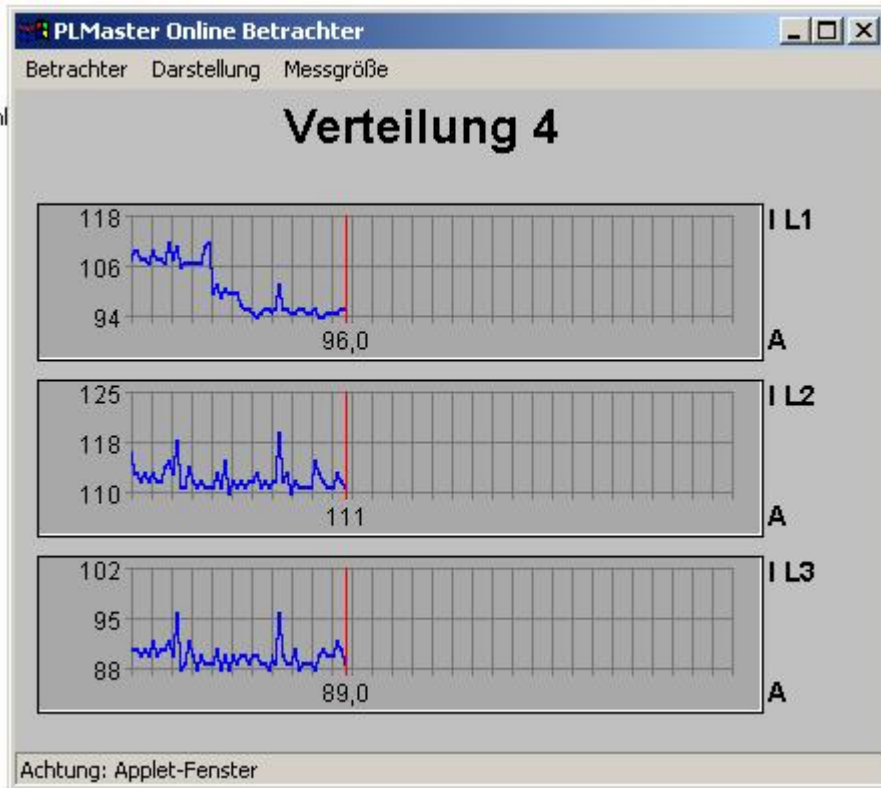
Nachfolgend eine Auswahl aus den möglichen Darstellungsarten.



Verteilung 4

	L1	L2	L3	
U	234,0	234,0	234,0	V
I	111,0	128,0	106,0	A
PF	0,45	0,42	0,33	-
P	11,0	12,0	8,00	kW
Q	23,0	27,0	23,0	kVAr

Achtung: Applet-Fenster



PLMaster Status Betrachter

Betrachter

	Sicherung			Stromrichtung			Sensor-	Schalt-
	L1	L2	L3	L1	L2	L3	status	stellung
Stanzerei Halle	OK	OK	OK	Ab	Ab	Ab	OK	Ein
Geb.8 KG Vert.1	OK	OK	OK	Ab	Ab	Ab	OK	Ein
Lauffer Presse	OK	OK	OK	Ab	Ab	Ab	OK	Ein
Kompressorraum	OK	OK	OK	Ab	Ab	Ab	OK	Ein
Galvanik PER-Anl	OK	OK	OK	Ab	Ab	Ab	OK	Ein
Haulik Presse	OK	OK	OK	Ab	Ab	Ab	OK	Ein
Verteilung 4	OK	OK	OK	Ab	Ab	Ab	OK	Ein

Achtung: Applet-Fenster

	Wirkarbeit [kWh]		Blindarbeit [kVAhr]	
	Bezug	Lieferung	Bezug	Lieferung
	Stanzerei Halle	8896	0	17169
Geb.8 KG Vert.1	15890	0	20829	0
Lauffer Presse	702	0	1154	0
Kompressorraum	16	0	18	0
Galvanik PER-Anl	10499	0	5347	0
Haulik Pesse	2359	1	7496	31
Verteilung 4	37497	0	54718	61

Achtung: Applet-Fenster

Die Visualisierung wird realisiert durch JAVA Applets, die in den Web-Browser geladen und dort ausgeführt werden. Diese JAVA Applets holen sich die aktuellen Messdaten ihrerseits über eine Netzwerkverbindung zu dem PLMaster System, von dem sie geladen wurden. Hierfür wird der spezielle **TCP Port 11278** benutzt, der im Netzwerk entsprechend zugänglich und freigeschaltet sein muß.

6 Die Programmierschnittstelle

6.1 Eingangsdaten

Keine.

6.2 Ausgangsdaten

Der PLMaster stellt folgende Daten über die Option PLMaster / WEB zur Verfügung:

Messdaten:

- aktuelle Messwerte wie Ströme, Spannungen, Leistungsfaktoren, Leistungen
- Statusinformationen
- Arbeitszähler

Konfigurationsdaten:

- konfigurierte Namen
- Minimum- und Maximumgrenzwerte
- Modultyp
- Nominalstrom und –spannung
- PLMaster Gruppen
- PLMaster Felder

Der detaillierte Aufbau der Datenstrukturen sowie das Kommunikationsprotokoll ist im Kapitel 7 beschrieben.

6.3 Client-Server Kommunikation

Die Option PLMaster / WEB stellt aus Sicht der Programmierschnittstelle einen Datenserver dar, der an dem definierten **Port 11278** und dem Protokoll TCP auf eingehende Client Anfragen wartet, diese entgegen nimmt, bearbeitet und die aufgebaute TCP Verbindung danach wieder schliesst. Auf diese Art ist es einerseits möglich, viele verschiedene Clients zu bedienen, andererseits gestaltet sich durch den Standard TCP/IP die Programmierung eines Clients recht einfach. Es können beliebige netzwerkfähige Betriebssysteme mit beliebiger Programmiersprache eingesetzt werden, z.B. C, C++, Java, Visual Basic, oder auch Scriptsprachen wie Tcl oder Perl.

7 Aufbau der Datenstrukturen

Der PLMaster stellt über die Option PLMaster / WEB folgende Informationen, eingebettet in Datenpakete für den Anwender zur Verfügung:

- aktuelle Messwerte
- Statusinformationen
- Zählerstände
- Konfigurationsdaten

Die Kodierung der übertragenen Werte erfolgt im "big endian" bzw. Motorola Format, d.h. bei den INT16, UINT16, INT32 bzw. UINT32 Typen werden die höherwertigen Bytes zuerst übertragen.

Die Statusinformationen werden bitweise kodiert übermittelt.

Bedingt durch Funktionserweiterungen des Systems unterscheiden sich die Datenstrukturen der jeweiligen Firmwarestände der Geräte.

Diese Unterschiede sind ab Kapitel 7.4 firmwarespezifisch dokumentiert.

7.1 Allgemeiner Datenpaket Aufbau

Client und Server kommunizieren nach aufgebauter Verbindung ("CONNECT") über Datenpakete. Der Client sendet ein Request Paket mit einer Request Identifikation, der Server antwortet mit einem Response Paket und den angeforderten Daten bzw. mit einer Fehlermeldung. Danach wird die Verbindung wieder abgebaut ("CLOSE"). Beide Pakettypen besitzen einen Kopfteil (Header) und einen optionalen Datenteil (Body).

7.2 Das Request Paket (Client)

Das Request Paket wird vom Client zum Server gesendet und besitzt folgenden Aufbau:

Byte-Pos.	Bezeichnung	Datentyp	Inhalt, Beschreibung
0..1	request_id	UINT16	Siehe Beschreibung
2..5	data_length	UINT32	Datenlänge, immer Null

Der Parameter **request_id** spezifiziert die Client Anforderung, der Parameter **data_length** enthält die Länge des optionalen Datenblocks im Paket nach dem Header. Es ist kein Client Request mit optionalen Daten definiert, deshalb ist **data_length** immer Null (0). Die Länge dieses Pakets ist 6 Bytes.

7.2.1 Request Identifikation

Die definierten Kennungen für den Parameter **request_id** sind:

Bezeichnung	Wert	Beschreibung
TCP_GET_CONFIG	2	Konfigurationsdaten des PLMaster
TCP_GET_DATA	1	Aktuelle Meßdaten aller Sensoren

7.3 Das Response Paket (Server)

Das Response Paket wird vom Server zum Client gesendet (als Antwort auf ein Request Paket) und besitzt folgenden Aufbau:

Byte-Pos.	Bezeichnung	Datentyp	Inhalt, Beschreibung
0..1	response_id	UINT16	Siehe Beschreibung
2..5	data_length	UINT32	Datenlänge
6..n	error_code / data	UINT16 / VAR	Fehlercode oder Daten

Der Parameter **response_id** spezifiziert die Server Antwort, der Parameter **data_length** enthält die Länge des Datenblocks im Paket nach dem Header. Der Datenblock ist in der Länge variabel und muß abhängig von **response_id** interpretiert werden, er enthält entweder die Daten des angeforderten Typs oder einen Fehlercode vom Datentyp UINT16. Die minimale Länge eines Response Paketes ist deshalb 8 Bytes, die maximale Länge abhängig von der Art der angeforderten Daten.

7.3.1 Response Identifikation

Die definierten Kennungen für den Parameter **response_id** sind:

Bezeichnung	Wert	Beschreibung
TCP_ERROR	0	Fehlerhafter Request
TCP_CONFIG	3	Konfigurationsdaten des PLMaster
TCP_DATA	2	Aktuelle Meßdaten aller Sensoren

7.3.2 Fehler Codes

Die definierten Kennungen für den Fehlercode **error_code** sind:

Bezeichnung	Wert	Beschreibung
TCP_ERR_MSGLEN	1	Falsche Längenangabe
TCP_ERR_UNKNOWN	2	Request unbekannt

7.4 Detaillierter Datenpaket Aufbau

7.4.1 Operation: Lesen der aktuellen Meßdaten

Die Prozedur zum Lesen / Holen der aktuellen Meßdaten aller 60 PLMaster Sensoren ist wie folgt:

Client Request:

Byte-Pos.	Bezeichnung	Datentyp	Inhalt, Beschreibung
0..1	request_id	UINT16	TCP_GET_DATA
2..5	data_length	UINT32	0

Server Response bei korrektem Request:

Byte-Pos.	Bezeichnung	Datentyp	Inhalt, Beschreibung
0..1	response_id	UINT16	TCP_DATA
2..5	data_length	UINT32	3840
6..3845	data		siehe Beschreibung unten

Bei einem korrekten Request liefert der Server ein Datenpaket mit den aktuellen Meßdaten aller 60 Sensoren des PLMaster zurück. Die Beschreibung dieser Struktur folgt weiter unten.

Server Response bei fehlerhaftem Request:

Byte-Pos.	Bezeichnung	Datentyp	Inhalt, Beschreibung
0..1	response_id	UINT16	TCP_ERROR
2..5	data_length	UINT32	2
6..7	error_code	UINT16	siehe Beschreibung

Bei einem fehlerhaften Request gibt der Server einen Fehlercode zurück. Die möglichen Fehlercodes sind weiter oben beschrieben.

Aufbau des Datenbereiches (data):

Byte-Pos.	Bezeichnung	Datentyp	Inhalt, Beschreibung
6..69	module_1		Meßdaten Sensor 1, siehe unten
70..133	module_2		Meßdaten Sensor 2
134..197	module_3		Meßdaten Sensor 3
...
...
3718..3781	module_59		Meßdaten Sensor 59
3782..3845	module_60		Meßdaten Sensor 60

Der Datenbereich setzt sich zusammen aus den jeweils 64 Bytes Meßdaten aller 60 Sensoren eines PLMaster. Die Daten der einzelnen Sensoren sind hintereinanderliegend im Datenbereich angeordnet. Die gesamte Länge beträgt demnach $60 * 64 = 3840$ Bytes.

Aufbau der Meßdaten eines Sensors (module_n):

Pos.	Bezeichnung	Datentyp	Inhalt, Beschreibung
0..3	status	Bitfeld32	Siehe Beschreibung unten
4..7	wp_f	INT32	Wirkarbeit Lieferung [kWh]
8..11	wq_f	INT32	Blindarbeit Lieferung [kVAhr]
12..15	wp_r	INT32	Wirkarbeit Bezug [kWh]
16..19	wq_r	INT32	Blindarbeit Bezug [kVAhr]
20..21	u_l1	INT16	Phase L1 Spannung [V]
22..23	u_l2	INT16	Phase L2 Spannung [V]
24..25	u_l3	INT16	Phase L3 Spannung [V]
26..27	i_l1	INT16	Phase L1 Strom [A]
28..29	i_l2	INT16	Phase L2 Strom [A]
30..31	i_l3	INT16	Phase L3 Strom [A]
32..33	p_l1	INT16	Phase L1 Wirkleistung [kW]
34..35	p_l2	INT16	Phase L2 Wirkleistung [kW]
36..37	p_l3	INT16	Phase L3 Wirkleistung [kW]
38..39	q_l1	INT16	Phase L1 Blindleistung [kVAr]
40..41	q_l2	INT16	Phase L2 Blindleistung [kVAr]
42..43	q_l3	INT16	Phase L3 Blindleistung [kVAr]
44..45	pf_l1	INT16	Phase L1 Leistungsfaktor [%]
46..47	pf_l2	INT16	Phase L2 Leistungsfaktor [%]
48..49	pf_l3	INT16	Phase L3 Leistungsfaktor [%]
50..63	reserved		Reserviert

Die Meßdaten eines Sensors besitzen eine Länge von 64 Bytes. Die Spalte Position in obiger Tabelle ist als relativer Offset innerhalb der Sensorposition im Datenpaket zu verstehen.

Aufbau des Status Bitfelds (status):

Pos.	Bezeichnung	Datentyp	Inhalt, Beschreibung
0	Switch_State	Bit	Schaltzustand 0 = Aus, 1 = Ein
1	Fuse_State	Bit	Sicherungszustand L1,L2,L3 (Sammelmeldung) 0 = Si. Ok, 1 = Si. Defekt
2	Res_2	Bit	Reserviert
3	Res_3	Bit	Reserviert
4	Fuse1_State	Bit	Sicherungszustand L1 0 = Si. Ok, 1 = Si. Defekt
5	Fuse2_State	Bit	Sicherungszustand L2 0 = Si. Ok, 1 = Si. Defekt
6	Fuse3_State	Bit	Sicherungszustand L3 0 = Si. Ok, 1 = Si. Defekt
7	Res_7	Bit	Reserviert
8	Cur1_Dir	Bit	Stromflußrichtung L1 0 = abgehend, 1 = zugehend
9	Cur2_Dir	Bit	Stromflußrichtung L2 0 = abgehend, 1 = zugehend
10	Cur3_Dir	Bit	Stromflußrichtung L3 0 = abgehend, 1 = zugehend
12	Res_12	Bit	Reserviert
13	Res_13	Bit	Reserviert
14	Res_14	Bit	Reserviert
15	Res_15	Bit	Reserviert
16	Res_16	Bit	Reserviert
17	Res_17	Bit	Reserviert
18	Res_18	Bit	Reserviert
19	Res_19	Bit	Reserviert
20	Res_20	Bit	Reserviert
21	Res_21	Bit	Reserviert
22	Res_22	Bit	Reserviert
23	Res_23	Bit	Reserviert
24	Res_24	Bit	Reserviert
25	Res_25	Bit	Reserviert
26	Res_26	Bit	Reserviert
27	Res_27	Bit	Reserviert
28	Res_28	Bit	Reserviert
29	Res_29	Bit	Reserviert
30	Mod_Active	Bit	Modul Aktivität 0 = nicht in Betrieb, 1 = Aktiv
31	Mod_Config	Bit	Modul Konfiguration 0 = undefiniert, 1 = konfiguriert

Die Spalte Position in obiger Tabelle ist als Bitposition innerhalb eines Statusworts eines Sensors zu verstehen. Bit 0 bezeichnet das niedrigstwertige Bit, Bit 31 das höchstwertige.

7.4.2 Operation: Lesen der Konfigurationsdaten

Die Prozedur zum Lesen / Holen der Konfigurationsdaten eines PLMaster Systems ist wie folgt:

Client Request:

Byte-Pos.	Bezeichnung	Datentyp	Inhalt, Beschreibung
0..1	request_id	UINT16	TCP_GET_CONFIG
2..5	data_length	UINT32	0

Server Response bei korrektem Request: Firmwarestand EMS 4.13/CMU 4.11

Byte-Pos.	Bezeichnung	Datentyp	Inhalt, Beschreibung
0..1	response_id	UINT16	TCP_CONFIG
2..5	data_length	UINT32	3708
6..3713	data		Siehe Beschreibung unten

Server Response bei korrektem Request: Firmwarestand EMS 4.15/CMU 4.12

Byte-Pos.	Bezeichnung	Datentyp	Inhalt, Beschreibung
0..1	response_id	UINT16	TCP_CONFIG
2..5	data_length	UINT32	3828
6..3713	data		Siehe Beschreibung unten

Bei einem korrekten Request liefert der Server ein Datenpaket mit den Konfigurationsdaten des PLMaster zurück. Die Beschreibung dieser Struktur folgt weiter unten.

Server Response bei fehlerhaftem Request:

Byte-Pos.	Bezeichnung	Datentyp	Inhalt, Beschreibung
0..1	response_id	UINT16	TCP_ERROR
2..5	data_length	UINT32	2
6..7	error_code	UINT16	siehe Beschreibung

Bei einem fehlerhaften Request gibt der Server einen Fehlercode zurück. Die möglichen Fehlercodes sind weiter oben beschrieben.

**Aufbau des Datenbereiches (data):
Firmwarestand EMS 4.13/CMU 4.11**

Byte-Pos.	Bezeichnung	Datentyp	Inhalt, Beschreibung
6..19	cfg_master		Konfiguration PLMaster, siehe unten
20 ..69	cfg_module_1		Konfiguration Sensor 1, siehe unten
70..	cfg_module_2		Konfiguration Sensor 2
...
...
..	cfg_module_59		Konfiguration Sensor 59
..3019	cfg_module_60		Konfiguration Sensor 60
3020..	cfg_field_1		Konfiguration Feld 1, siehe unten
...	cfg_field_2		Konfiguration Feld 2
...
...	cfg_field_9		Konfiguration Feld 9
..3239	cfg_field_10		Konfiguration Feld 10
3240..	cfg_group_1		Konfiguration Gruppe 1, siehe unten
	cfg_group_2		Konfiguration Gruppe 2

	cfg_group_9		Konfiguration Gruppe 9
..3409	cfg_group_10		Konfiguration Gruppe 10

Der Datenbereich setzt sich zusammen aus 16 Bytes Konfiguration für den PLMaster selbst, den jeweils 50 Bytes Konfiguration der 60 Sensoren, den jeweils 22 Bytes Konfiguration der 10 Felder sowie den jeweils 17 Bytes Konfiguration der 10 Gruppen. Die Daten sind hintereinanderliegend im Datenbereich angeordnet. Die gesamte Länge beträgt demnach:

$$14 + (60 * 50) + (10 * 22) + (10 * 17) + 304[\text{reserviert f. IOM}] = 3708 \text{ Bytes.}$$

**Aufbau des Datenbereiches (data):
Firmwarestand EMS 4.15/CMU 4.12**

Byte-Pos.	Bezeichnung	Datentyp	Inhalt, Beschreibung
6..19	cfg_master		Konfiguration PLMaster, siehe unten
20 ..71	cfg_module_1		Konfiguration Sensor 1, siehe unten
72..	cfg_module_2		Konfiguration Sensor 2
...
...
..	cfg_module_59		Konfiguration Sensor 59
..3139	cfg_module_60		Konfiguration Sensor 60
3140..	cfg_field_1		Konfiguration Feld 1, siehe unten
...	cfg_field_2		Konfiguration Feld 2
...
...	cfg_field_9		Konfiguration Feld 9
..3359	cfg_field_10		Konfiguration Feld 10
3360..	cfg_group_1		Konfiguration Gruppe 1, siehe unten
	cfg_group_2		Konfiguration Gruppe 2

	cfg_group_9		Konfiguration Gruppe 9
..3529	cfg_group_10		Konfiguration Gruppe 10

Der Datenbereich setzt sich zusammen aus 16 Bytes Konfiguration für den PLMaster selbst, den jeweils 52 Bytes Konfiguration der 60 Sensoren, den jeweils 22 Bytes Konfiguration der 10 Felder sowie den jeweils 17 Bytes Konfiguration der 10 Gruppen. Die Daten sind hintereinanderliegend im Datenbereich angeordnet. Die gesamte Länge beträgt demnach:

$$14 + (60 * 52) + (10 * 22) + (10 * 17) + 304[\text{reserviert f. IOM}] = 3828 \text{ Bytes.}$$

Aufbau der Konfigurationsdaten des PLMaster (cfg_master):

Pos.	Bezeichnung	Datentyp	Inhalt, Beschreibung
0..8	name	STRING	PLMaster Name als Zeichenkette nach „C“-Konvention
9	daylight_saving	INT8	Automatische Sommer-/Winterzeitumstellung 0 = inaktiv, 1 = aktiv
10..11	hw_version	UINT16	Hardware Versionsnummer
12..13	sw_version	UINT16	Software Versionsnummer

Die Konfigurationsdaten des PLMaster besitzen eine Länge von 14 Bytes. Die Spalte Position in obiger Tabelle ist als relativer Offset innerhalb der genauen Position im Datenpaket zu verstehen. Die Kodierung der Strings erfolgt nach „C“-Konvention, d.h. jeder String ist mit einer Null (0) terminiert.

Aufbau der Konfigurationsdaten eines Sensors (cfg_module_n):
 Firmwarestand EMS 4.13/CMU 4.11

Pos.	Bezeichnung	Datentyp	Inhalt, Beschreibung
0..1	f_nom	INT16	Sicherungsnennwert (A)
2..3	l_nom	INT16	Wandlernennwert (A)
4..7	i_max	INT32	Maximalstrom [A]
8..11	i_min	INT32	Minimalstrom [A]
12..15	u_max	INT32	Maximalspannung [V]
16..19	u_min	INT32	Minimalspannung [V]
20	Type	INT8	Sensortyp, siehe unten
21	Direc	INT8	Richtungsdefinition: 0 = Abgehend, 1 = Zugehend
22	Bus	INT8	Lokaler Bus
23	f_ref	INT8	Feldindex
24	g_ref	INT8	Gruppenindex
25	a_time	INT8	Mittelungszeit, siehe unten
26	i_log	INT8	Strom Aufzeichnung 0 = aus, 1 = ein
27	u_log	INT8	Spannung Aufzeichnung 0 = aus, 1 = ein
28	pf_log	INT8	Leistungsfaktor Aufzeichnung 0 = aus, 1 = ein
29	w_log	INT8	Arbeit Aufzeichnung 0 = aus, 1 = ein
30	dw_log	INT8	Arbeit/Intervall Aufzeichnung 0 = aus, 1 = ein
31	p_log	INT8	Wirkleistung Aufzeichnung 0 = aus, 1 = ein
32	q_log	INT8	Blindleistung Aufzeichnung 0 = aus, 1 = ein
33..49	name	STRING	Sensor Name als Zeichenkette nach „C“- Konvention

Die Konfigurationsdaten eines Sensors besitzen eine Länge von 50 Bytes. Die Spalte Position in obiger Tabelle ist als relativer Offset innerhalb der Sensorposition im Datenpaket zu verstehen. Die Kodierung der Strings erfolgt nach „C“-Konvention, d.h. jeder String ist mit einer Null (0) terminiert.

Aufbau der Konfigurationsdaten eines Sensors (cfg_module_n):
Firmwarestand EMS 4.15/CMU 4.12

Pos.	Bezeichnung	Datentyp	Inhalt, Beschreibung
0..1	f_nom	INT16	Sicherungsnennwert (A)
2..3	i_nom	INT16	Wandlernennwert (A)
4..5	u_nom	INT16	Spannungsmessbereich [V]
6..9	i_max	INT32	Maximalstrom [A]
10..13	i_min	INT32	Minimalstrom [A]
14..17	U_max	INT32	Maximalspannung [V]
18..21	U_min	INT32	Minimalspannung [V]
22	Type	INT8	Sensortyp, siehe unten
23	Direc	INT8	Richtungsdefinition: 0=Abgehend, 1=Zugehend
24	Bus	INT8	Lokaler Bus
25	f_ref	INT8	Feldindex
26	g_ref	INT8	Gruppenindex
27	a_time	INT8	Mittelungszeit, siehe unten
28	i_log	INT8	Strom Aufzeichnung 0=aus, 1=ein
29	u_log	INT8	Spannung Aufzeichnung 0=aus, 1=ein
30	pf_log	INT8	Leistungsfaktor Aufzeichnung 0=aus, 1=ein
31	w_log	INT8	Arbeit Aufzeichnung 0=aus, 1=ein
32	dw_log	INT8	Arbeit/Intervall Aufzeichnung 0=aus, 1=ein
33	p_log	INT8	Wirkleistung Aufzeichnung 0=aus, 1=ein
34	q_log	INT8	Blindleistung Aufzeichnung 0=aus, 1=ein
35..51	name	STRING	Sensor Name als Zeichenkette nach „C“- Konvention

Die Konfigurationsdaten eines Sensors besitzen eine Länge von 52 Bytes. Die Spalte Position in obiger Tabelle ist als relativer Offset innerhalb der Sensorposition im Datenpaket zu verstehen. Die Kodierung der Strings erfolgt nach „C“-Konvention, d.h. jeder String ist mit einer Null (0) terminiert.

Aufbau der Konfigurationsdaten eines Feldes (cfg_field):

Pos.	Bezeichnung	Datentyp	Inhalt, Beschreibung
0..3	rat_curr	INT32	Bemessungsstrom [A]
4..20	name	STRING	Feld Name als Zeichenkette nach „C“-Konvention
21	reduc	INT8	Reduktionsfaktor [%]

Die Konfigurationsdaten eines Feldes besitzen eine Länge von 22 Bytes. Die Spalte Position in obiger Tabelle ist als relativer Offset innerhalb der genauen Position im Datenpaket zu verstehen. Die Kodierung der Strings erfolgt nach „C“-Konvention, d.h. jeder String ist mit einer Null (0) terminiert.

Aufbau der Konfigurationsdaten einer Gruppe (cfg_group):

Pos.	Bezeichnung	Datentyp	Inhalt, Beschreibung
0..16	name	STRING	Gruppen Name als Zeichenkette nach „C“-Konvention

Die Konfigurationsdaten einer Gruppe besitzen eine Länge von 17 Bytes. Die Spalte Position in obiger Tabelle ist als relativer Offset innerhalb der genauen Position im Datenpaket zu verstehen. Die Kodierung der Strings erfolgt nach „C“-Konvention, d.h. jeder String ist mit einer Null (0) terminiert.

Kodierung des Sensortyps:

Wert	Inhalt, Beschreibung
1	EMDE
2	SASIL 00 EMDE
3	SASIL 1 EMDE
4	SASIL 2 EMDE
5	SASIL 3 EMDE
6	SASIL-MOT 00 EMDE
7	SASIL-MOT 1 EMDE
8	SASIL-MOT 2 EMDE
9	SASIL-MOT 3 EMDE
10	SASIL-MOT 00
11	SASIL-MOT 1
12	SASIL-MOT 2
13	SASIL-MOT 3
14	Leistungsschalter EMDE

Kodierung der Mittelungszeit:

Wert	Inhalt, Beschreibung
0	1 Minute
1	5 Minuten
2	8 Minuten
3	10 Minuten
4	15 Minuten
5	30 Minuten
6	60 Minuten

Anhang A, Beispiel eines Client Programms

Das folgende einfache Programm zeigt beispielhaft die Implementierung eines Clients in der Programmiersprache Java.

Der Client sendet ein Request Paket zur PLMaster / WEB mit der IP-Adresse 192.168.200.12, empfängt das Response Paket und schickt die Werte an die Konsole.

```
import java.util.*;
import java.net.*;
import java.io.*;

public class ExampleClient {

    public static void main(String[] args) throws IOException {
        final short TCP_ERROR = 0;
        final short TCP_DATA = 2;
        final short TCP_GET_DATA = 1;
        final int DATA_LENGTH = 0;
        String hostname = "192.168.200.12";
        int port = 11278;
        short ShortVal;
        int IntVal;

        Socket s = new Socket(hostname, port);

        InputStream sin = s.getInputStream();
        DataInputStream fromServer = new DataInputStream(sin);

        OutputStream sout = s.getOutputStream();
        DataOutputStream toServer = new DataOutputStream(sout);

        toServer.writeShort(TCP_GET_DATA);
        toServer.writeInt(DATA_LENGTH);

        try {
            ShortVal = fromServer.readShort();
            System.out.println("Response ID: " + ShortVal);
            IntVal = fromServer.readInt();
            System.out.println("Length:      " + IntVal);
            for (int i=0; i<60; i++) {
                System.out.println("Sensor:      " + (i+1));
                IntVal = fromServer.readInt();
                System.out.println("Status:      " + IntVal);
                IntVal = fromServer.readInt();
                System.out.println("WP_F:        " + IntVal);
                IntVal = fromServer.readInt();
                System.out.println("WQ_F:        " + IntVal);
                IntVal = fromServer.readInt();
                System.out.println("WP_R:        " + IntVal);
                IntVal = fromServer.readInt();
                System.out.println("WQ_R:        " + IntVal);
            }
        }
    }
}
```

```

ShortVal = fromServer.readShort();
System.out.println("U_L1:      " + ShortVal);
ShortVal = fromServer.readShort();
System.out.println("U_L2:      " + ShortVal);
ShortVal = fromServer.readShort();
System.out.println("U_L3:      " + ShortVal);
ShortVal = fromServer.readShort();
System.out.println("I_L1:      " + ShortVal);
ShortVal = fromServer.readShort();
System.out.println("I_L2:      " + ShortVal);
ShortVal = fromServer.readShort();
System.out.println("I_L3:      " + ShortVal);
ShortVal = fromServer.readShort();
System.out.println("P_L1:      " + ShortVal);
ShortVal = fromServer.readShort();
System.out.println("P_L2:      " + ShortVal);
ShortVal = fromServer.readShort();
System.out.println("P_L3:      " + ShortVal);
ShortVal = fromServer.readShort();
System.out.println("Q_L1:      " + ShortVal);
ShortVal = fromServer.readShort();
System.out.println("Q_L2:      " + ShortVal);
ShortVal = fromServer.readShort();
System.out.println("Q_L3:      " + ShortVal);
ShortVal = fromServer.readShort();
System.out.println("PF_L1:     " + ShortVal);
ShortVal = fromServer.readShort();
System.out.println("PF_L2:     " + ShortVal);
ShortVal = fromServer.readShort();
System.out.println("PF_L3:     " + ShortVal);
IntVal = fromServer.readInt();
IntVal = fromServer.readInt();
IntVal = fromServer.readInt();
ShortVal = fromServer.readShort();
    }
} catch (EOFException e) {}

toServer.close();
fromServer.close();
s.close();
}
}

```