Modhub Entwickler-Referenz [INTERN]

Martin Strubel

20. Mai 2025

Revision:
`v0.2-rc1`

1 Funktionsumfang

Abb. 1.1 zeigt die Frontansicht des modhub/modnet Geräts.

1.1 Stromversorgung

Die Stromversorgung geschieht über die Pins GND und 24V+. Betriebsparameter siehe Tabelle 1.1.

24V+	12-26V (Plus) Gleichspannung, maximal 30V, 1A
GND	Masse (Minus)

Tabelle 1.1: Stromversorgung

Keine Wechselspannungen anschliessen

1.2 Status-Lichtsignale

Anordnung der Lichtsignale auf der Frontscheibe:

```
(1)  (2)
(3)  (4)
(5)  (6)
(7)
```

1. GFG Bus Aktivität (gelb) Blinkt langsam, wenn Bus aktiv. Bei fehlenden Daten ist die LED aus
2. GFG Bus Error (rot) Bei fehlenden Daten blinkt diese LED rot. Falls die Daten nicht geloggt werden können, ist die LED dauernd an. In diesem Fall muss die Konfiguration der Sensoren neu erzeugt werden.
3. Modbus Slave Aktivität (gelb) Bei Zugriff auf den Modbus-Slave blitzt diese LED kurz auf.
4. Modbus Slave Error (rot) Bei einem Bus-Fehler (CRC-Error, etc.) blinkt diese LED
5. Mode-LED (grün) Bei laufender Datenerfassung leuchtet die LED konstant. Während der Sensor-Probe blinkt sie beim Aufstarten mit `Autoprobe`=1 oder bei expliziter Probe-Anweisung während der Zeit der Erfassung (normalerweise 5s). Falls sie fortwährend blinkt, ist das System im Simulations-Modus (bei nicht angeschlossenen Sensoren).
6. Ernsthafter Betriebs-Fehler (rot) Bei einem permanenten Fehler, der eine Neukonfiguration oder Neustart des Systems erfordert, ist diese LED dauernd an. Falls das Datum nicht korrekt gesetzt ist, blinkt diese LED.

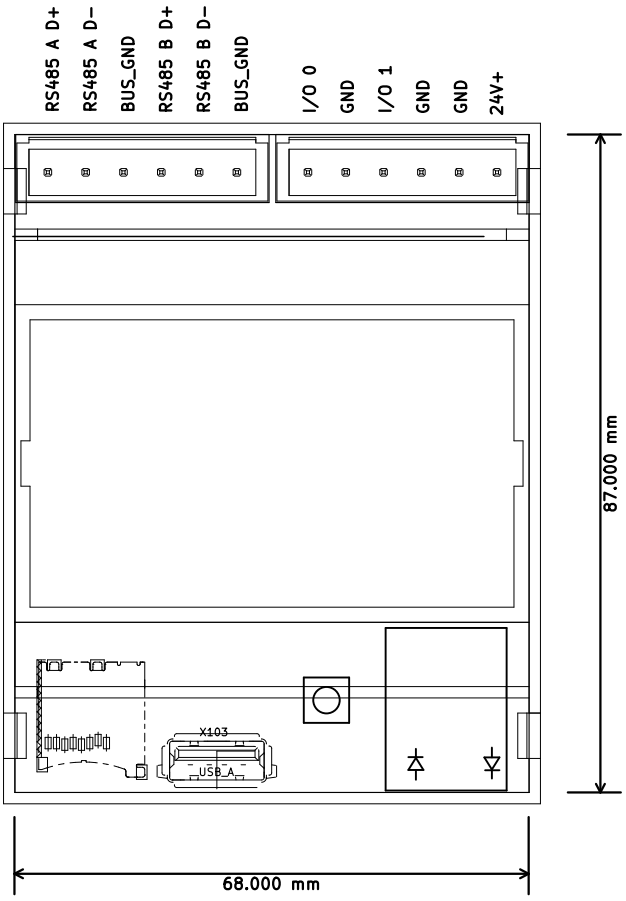

Abbildung 1.1: modhub Front-Ansicht

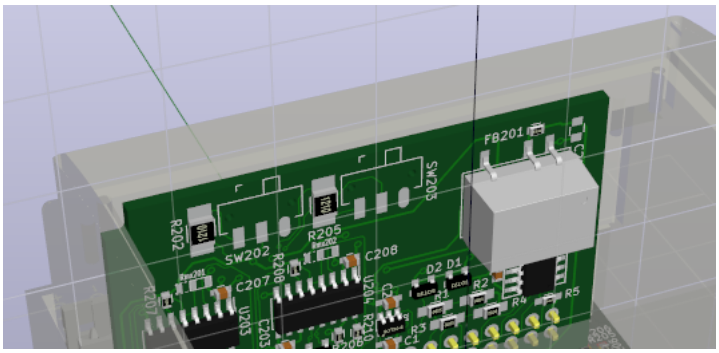

Abbildung 1.2: Schalter-Schema

7. WLAN-Aktivität (blau) Wenn das WLAN aktiv ist, blinkt diese LED im Herzklopfen-Modus. Bei ausgeschaltetem WLAN ist die LED aus.

Zu den möglichen Fehlern siehe auch Appendix 7.

1.3 Taster

Der Taster erfüllt momentan nur die einfache Funktion der Sensorkonfiguration bzw. Aenderung der Betriebsart:

1. Betriebsart LOGGING (grüne LED (5) an): Langer Druck schaltet in DEFAULT-Betriebsart
2. Betriebsart DEFAULT: Kurzer Druck löst Sensor-Probe aus

1.4 RS485-Ports

Zwei Ports A und B stehen für RS485-Anschlüsse zur Verfügung. Die Anschlüsse sind belegt wie folgt:

Port A
: GfG-Sensorbus

Port B
: RS485 Modbus kompatibler Slave-Bus für externe Master-Geräte

Die Bus-Terminierung wird für beide Busse auf dem I/O-Board vorkonfiguriert, siehe Abb. 1.2 (Front-Abdeckung entfernt). In der in der Grafik notierten Schalterstellung ist die Terminierung nicht aktiv (‘Aus’). Die empfohlene Einstellung:

BUS A (SW202)
: Aus

BUS B (SW203)
: An

1.4.1 Konfiguration

Die Standardkonfiguration des Modbus-Ports ist bei Auslieferung wie folgt:

Baudrate
: 19200

Parity
Even

Bits
8 Daten, 1 Stop

1.5 TCP-Interface

Das TCP-Interface bietet eine optionale Möglichkeit für die Integration in ein bestehendes Ethernet-Netzwerk. Momentan ist es nur für den Service-Zugang vorgesehen.

1.6 USB-Port

Der USB-Port bietet eine optionale Möglichkeit des System-Update wie auch das Auslesen der Log-Dateien. Momentan nicht unterstützt.

1.7 WLAN-Funktion

Das WLAN-Interface bietet die Möglichkeit der drahtlosen Konfiguration des Geräts. In der momentanen Konfiguration ist es dauernd aktiviert und bietet einen Accesspoint mit entsprechenden Anmeldeparametern in Tabelle 1.2.

BSSID-Kennung	modhub
Passwort	modhub2016
Server-Adresse (Webserver)	http://192.168.12.1

Tabelle 1.2: WLAN-Parameter (Nur Technikerzugang)

2

Hardware

2.1 Einrichten

- Allenfalls Batterie einlöten
- Kartenrohling einsetzen
- System hochstarten
- Per WLAN von einer Linux-Plattform aus Datum setzen, z.B.:

  ```
  ssh modhub date -s \’‘date‘\’ \; hwclock -w
  ```

- Systemtest ausführen (TBD). Geplant ist, dass der Systemtest das Datum setzt.

2.1.1 Testsystem

Anschluss per Ethernet-Kabel.
Zu testen:

1. Beide RS485 Busse
2. Ethernet und WLAN
3. Abschliessendes Setzen des Datums
4. Vergabe einer Seriennummer

2.2 Wartung

Nach jedem Wartungsvorgang muss der Systemtest (TBD) neu ausgeführt werden!

2.2.1 WLAN-Karte ersetzen

Bei Austauschen der WLAN-Karte müssen folgende Schritte vorgenommen werden, ansonsten wird das WLAN nicht aktiviert:

1. SD-Karte auf PC mounten
2. Auf rootfs-Partition das File `/etc/udev/rules.d/70-persistent-net.rules` löschen
3. System neu starten

2.2.2 I/O Board ersetzen

1. Unbedingt sicherstellen, dass das Gerät ausgeschaltet ist
2. Darauf achten, dass eingefräste Nut am Board links bei Pin 1 (Marke) liegt
3. I/O Board einstecken

3 System

Das System wird auf der SD-Karte geliefert.

3.1 Einrichtung SD-Karte

3.1.1 SD-Karte klonen

Die "goldene" SD-Karte mit einem Clone-Tool oder Linux dd auf eine frische kopieren:

```
dd if=<golden card> of=<virgin card> bs=1048576
```

Die Quell- und Zielkarte muss vom Typ absolut identisch sein. Ansonsten können später im Betrieb Fehler auftreten.

3.1.2 SD-Karte einrichten

1. Partitionen entsprechend Tabelle 3.1 einrichten.
2. Tar-Files entpacken (Tabelle 3.2)

Name	Filesystem	Grösse in 1k blocks
boot	vfat	31168
rootfs	ext4	1998672
data	ext4	beliebig gross, min 1 GB

Tabelle 3.1: Partitionen

Filename	Zielpartition
modhub-root.tgz	rootfs
modhub-boot.tgz	boot

Tabelle 3.2: Tar-Archive

Auf der /data-Partition muss allenfalls ein leerer Ordner `log` eingerichtet werden.

3.2 System-Wartung

Login ins System geschieht per ssh entsprechend der Login-Daten in Tabelle 3.3.

3.3 Software

Wichtige Systempfade für installierte Software sind in Tabelle 3.4 aufgeführt.

Login	Passwort
root	modhub

Tabelle 3.3: Login-Daten

Paket	Pfad
Modhub netpp slave	/usr/local/netpp/slave
Webserver-Daten	/var/www

Tabelle 3.4: Pfade

4

Konfiguration/Eigenschaften

Die Konfigurationsparameter werden im Folgenden unter abstrakten Eigenschaften (engl. 'Properties') abstrahiert.
Die Properties sind typischerweise als Betriebsparameter anzusehen. Per netpp (network property protocol) können die einzelnen Properties von aussen per Netzwerk angesprochen werden. Zudem sind Interfaces vorgesehen, die eine Fernkonfiguration wie folgt erlauben:

- Webserver (TBD)
- pvserver (Prozesskontroll-Software)

Prinzipiell gilt für das netpp-Protokoll eine einfaches Schema:

1. Property-Wert auslesen (Laufzeit-Datentyp wird zurückgegeben)
2. Property-Wert setzen (Datentyp muss gesetzt und zur Laufzeit übergeben werden)
3. Property-Hierarchie abfragen (Min/Max-Wert, etc.)

Für die Nutzer-Standardeinstellung wird zusätzlich ein INI-File im Verzeichnis des modhub-servers abgelegt (`default.ini`). Diese Datei kann mit einem Texteditor bearbeitet werden und enthält die Konfiguration der relevanten (und schreibbaren) Properties. Bei Aufruf von `Store` wird die aktuelle Konfiguration in dieser Datei gespeichert. Ein `FactoryReset` löscht diese Datei.

4.1 Konfiguration Werteaufzeichnung

Normalerweise wird die Konfiguration der Sensoren automatisch beim Start vorgenommen. Sofern eine manuelle Neukonfiguration erforderlich ist, müssen die RRDLog-Parameter (Tabelle 4.14) gesetzt werden.

Enable
: Wenn 1, ist das Logging aktiv

Frequency
: Messintervallzeit (in s). Erst nach Neuerstellung des Logfiles (RRDLog.Configure) aktiv.

HeartBeat
: Maximale Ausfallzeit [s] eines Sensorwerts bevor ein Wert "UNDEFINIERT" aufgezeichnet wird

Configure
: Wenn auf 1 gesetzt, wird die Log-Datei neu erzeugt. Achtung: Die bisher aufgezeichneten Werte werden gelöscht!

Filename
: Pfadname des RRD-Logfiles (RRDLog.Configure)

SimFilename
: Pfadname des Simulations-Logfiles (erst nach Neustart des Systems aktiv)

4.2 Dynamische Eigenschaften

Im Falle der angeschlossenen Sensoren, die sich allenfalls nach einer Neukonfiguration ändern können, existieren dynamische Property-Container, z.B. `Sensors` (siehe Tabelle 4.12)
Innerhalb dieses Containers erzeugen die Properties New und Probe neue 'Kinder' wie folgt:

New <String>
Erzeugt einen neuen Sensor-Knoten mit Namen <String>

Probe
Löscht alle bestehenden Sensoren-Knoten und sucht nach Sensoren-Nachrichten auf dem Sensoren-Bus

Innerhalb dieses Containers werden die Sensoren als Properties angesprochen. Eine Auflistung aller Container erfolgt z.B. über die netpp-Abfrage:

```
$ netpp TCP:modhub Sensors
Type : {Struct} [RW.]
Child: (s) [80000004] 'New'
Child: (c) [80000005] 'Probe'
Child: (S) [80000006] 'Channel0'
Child: (S) [80000015] 'Channel1'
Child: (S) [80000024] 'Channel2'
Child: (S) [80000033] 'Channel3'
Child: (S) [80000042] 'Channel4'
Child: (S) [80000051] 'Channel5'
Child: (S) [80000060] 'Channel6'
```

Ein einzelner Sensor wird somit wie folgt angesprochen:

```
$ netpp TCP:modhub Sensors.Channel0
Type : {Struct} [RW.]
Child: (m) [80000007] 'Type'
Child: (m) [8000000a] 'Datatype'
Child: (u) [8000000e] 'Address'
.
.
.
Child: (c) [80000010] 'Delete'
Child: (c) [80000011] 'Dump'
Child: (A) [80000012] 'Value'
```

Die Parameter-Beschreibung eines einzelnen Sensors ist Tabelle 4.1 zu entnehmen.

4.2.1 Detaillierte Sensorkonfiguration

Je nach Modus (Mode) enthält der Sensors-Container beim Start des Systems unterschiedliche Konfigurationen, d.h. die Sensor-Instanzen werden wie folgt angezeigt:

SIMULATION
Simulate0...Simulate3

LOGGING:
Channel0..Channel<n>

Property	Type	Flags	Description
Type	MODE	RW	Sensor type code
Datatype	MODE	RW	Sensor data format type
Address	REGISTER	RW	Sensor address on bus
Index	REGISTER	RW	Internal node index, normally 0
Delete	COMMAND	WO	Remove sensor entry from bus
Dump	COMMAND	RW	Dump sensor data (debugging only)
Flags	REGISTER	RO	Sensor data flags, see Flags0 modbus register
Threshold	ARRAY	RW	Threshold per sensor channel
Value	ARRAY	RW	Sensor float value array
RawValue	ARRAY	RW	Sensor raw integer value array

Tabelle 4.1: Struct Sensor

Value	Mode name	Description
1	CO2	CO2 single channel sensor
2	CO_NO	Dual channel sensor

Tabelle 4.2: Mode Sensor.Type – possible values

4.3 Web-Interface

Das Web-Interface wird vom Kunden designt.

4.3.1 Zugriff auf Web-Directory

- Per Samba/Windows share 'www' mounten. Eingabe eines Passworts ist nicht nötig
- Alternativ: Verbindung per ssh/scp. Root-Zugriff siehe Tabelle 3.3.
- Nach Möglichkeit während der Entwicklung im Verzeichnis `testing` arbeiten.

4.3.2 CGI-Interface

Der Aufruf des CGI-Interface ist aufgebaut wie folgt:

`/pynetpp.cgi?action=ACTION&prop=STRING&val=DATA`

Mögliche Kommandos für 'action':

get
: Property abfragen, ersetze STRING durch Property-Namen, ab &val weglassen.

set Property setzen, ersetze STRING durch Property-Namen, DATA durch Wert des Property

query
: Abfrage aller Properties, wenn prop gesetzt, zeige Kinder dieses Property, ansonsten alle Top-Level-Properties

Beispiele in HTML (Siehe config.html im WWW-Verzeichnis auf modhub):

Value	Mode name	Description
DC_INT	INT	16 bit integer format
DC_UINT	UINT	16 bit unsigned integer format
DC_FLOAT	FLOAT	Normalized float format

Tabelle 4.3: Mode `Sensor.Datatype` – possible values

```
<li><a href="/pynetpp.cgi?action=get&prop=Mode">Show Operation mode</a></li>
<li><a href="/pynetpp.cgi?action=query">Show all properties</a></li>
<li>Set LogFrequency to <a href="/pynetpp.cgi?action=set&prop=RRDLog.Frequency&val=2">2s</a></li>
<li>Set LogFrequency to <a href="/pynetpp.cgi?action=set&prop=RRDLog.Frequency&val=1">1s</a></li>
<li>Run <a href="/pynetpp.cgi?action=set&prop=Sensors.Probe&val=1">GfgBus Probe</a></li>
<li>Show <a href="/pynetpp.cgi?action=query&prop=Sensors">Sensor list</a></li>
<li>Turn slave
<a href="/pynetpp.cgi?action=set&prop=SlaveEnable&val=1">on</a> |
<a href="/pynetpp.cgi?action=set&prop=SlaveEnable&val=0">off</a>
</li>
```

4.4 Property-Referenz (englisch)

Die Referenz zu allen momentan implementierten Eigenschaften (Properties) gliedert sich in verschiedene Gruppen:

Parameters
: Betriebsparameter

StartupConfig
: Parameter, die Standardeinstellungen (beim Neustart) betreffen

SensorConfig
: Properties zur Sensorenkonfiguration.

Die Referenz ist gültig für:
modhub
Device Revision: 0.2

4.4.1 Parameters

Property	Type	Flags	Description
`Version`	STRING	RO	Firmware version string
`Mode`	MODE	RO	Current mode of operation
`Date`	STRING	RO	Current system date as string
`Modbus`	STRUCT	RW	Modbus configuration

Tabelle 4.4: Property group 'Parameters'

See Tabelle 4.4

Value	Mode name	Description
OPMODE_DEFAULT	DEFAULT	Default operation (configuration)
OPMODE_PROBE	PROBE	Bus probe for attached sensors
OPMODE_LOGGING	LOGGING	Logging mode (record sensor values)
OPMODE_SIMULATOR	SIMULATOR	Simulation mode, generate sensor values
OPMODE_DEBUG	DEBUG	Debug mode

Tabelle 4.5: Mode `Mode` – possible values

Value	Mode name	Description
96	9600	9600 bps
192	19200	19200 bps (default)
384	38400	38400 bps
1152	115200	115200 bps

Tabelle 4.6: Mode `Modbus.Baudrate` – possible values

Value	Mode name	Description
PARITY_NONE	NONE	No parity check
PARITY_EVEN	EVEN	Even parity
PARITY_ODD	ODD	Odd parity

Tabelle 4.7: Mode `Modbus.Parity` – possible values

Property	Type	Flags	Description
`Baudrate`	MODE	RW	Modbus baud rate setting
`SlaveId`	INT	RW	Modbus slave ID
`Parity`	MODE	RW	Parity configuration ('N', 'E', 'O')
`SlaveEnable`	BOOL	RW	Enable modus slave server
`ModbusDeviceName`	STRING	RW	Device name for Modbus terminal (Bus B)
`GfgbusDeviceName`	STRING	RW	Device name for GFGbus terminal (Bus A)

Tabelle 4.8: Struct `Modbus`

4.4.2 StartupConfig

Property	Type	Flags	Description
`FactoryReset`	COMMAND	WO	Factory reset, remove all user settings
`Store`	COMMAND	WO	Store all current settings in startup INI

Tabelle 4.9: Property group 'StartupConfig'

See Tabelle 4.9

4.4.3 SensorConfig

Property	Type	Flags	Description
SensorConfig	STRUCT	RW	Sensor configuration items
Sensors	STRUCT	RW	Sensor container. All dynamically created sensors are listed in here.
RRDLog	STRUCT	RW	Round robin database properties
Logging	STRUCT	RW	Logging parameters
Relais	ARRAY	RW	Relais monitor array

Tabelle 4.10: Property group 'SensorConfig'

See Tabelle 4.10

Property	Type	Flags	Description
Autoprobe	BOOL	RW	When set, slave automatically probes connected sensors at startup
SensorTimeout	INT	RW	GFG Sensor bus timeout in ms. After this timeout, sensor values are assumed ready and are written to the database.
ProbeTimeout	INT	RW	After this timeout [ms], the PROBE mode enters LOGGING mode, when sensors are detected.
DefaultThreshold	INT	RW	Default threshold for single channel sensor

Tabelle 4.11: Struct SensorConfig

Property	Type	Flags	Description
New	STRING	WO	Create a new sensor
Probe	COMMAND	RW	Run probe to populate sensor list. 1: Probing active, 0: Terminate probing

Tabelle 4.12: Struct Sensors

Property	Type	Flags	Description
State	BOOL	RO	Relais state (1: active, 0: off)

Tabelle 4.13: Array item Relais[i]

Property	Type	Flags	Description
Enable	BOOL	RW	RRD enable (default on)
Frequency	INT	RW	Logging frequency, default 1s
HeartBeat	INT	RW	Minimum RRD logging heart beat
Configure	COMMAND	WO	Configure new database. This clears all previous data from the log file.
Filename	STRING	RW	RRD file path
SimFilename	STRING	RW	Simulation RRD file path

Tabelle 4.14: Struct RRDLog

Property	Type	Flags	Description
LogRefresh	COMMAND	WO	Refresh error log. Updates LogBuffer (see event handling)
LogFilename	STRING	RW	Error log file path

Tabelle 4.15: Struct Logging

5 Modbus-Register

Wenn die SlaveEnable Option aktiviert ist, ist auf Port 2 der Modbus-Slave aktiv. Die Register definieren sich wie folgt und sind gültig für:
modbus_slave
Device Revision: 0.1rc1

5.1 Input-Register

Diese Register werden mit dem Funktionscode 0x04 angesprochen. Tabelle 5.4 zeigt die implementierten Register. Ein Register ist grundsätzlich ein Wort (16 bit) gross. Ansonsten ist in eckigen Klammern der Umfang (die Anzahl Worte, die der Registerbereich enthält) angegeben. Diese Angabe ist für Register-Felder (Arrays) relevant.
Das Gerät kann maximal Werte von 32 Sensoren a 4 Kanäle erfassen. Diese Werte sind ab Value gespeichert, falls gültig. Die Gültigkeit der Werte wird durch Bit DATA_VALID in dem entsprechenden Flags-Register (Tabelle 5.3) bestimmt.
Der allgemeine Betriebsstatus kann im Register Status (OPMODE) abgefragt werden. Der Inhalt des OPMODE Bitfeldes bestimmt dabei die Betriebsart ensprechend Tabelle 5.2.

`Status`
: Status-Register

`FlagsArray`
: Flags register für 32 Sensor-Einheiten [0..31]

`Flags0`
: Flags register für Sensor 0

`TypeCodeArray`
: Typecode-Register für 64 Kanäle (32 Sensoren a 2 Kanäle)

`TypeCode0`
: Typecode-Register für Kanal 0.0/0.1

`ValueArray`
: Werteblock Sensoren 0..31, Werte 0..3 (Wert A, B; Schwellwert A, B)

`Value0_A`
: Sensor 0 Wert A

`Value0_B`
: Sensor 0 Wert B (reserviert)

`Thresh0_A`
: Grenzwert Sensor 0 Kanal A (default: 19000)

`Thresh0_B`
: Grenzwert Sensor 0 Kanal B (reserviert)

`Value1_A`
: Sensor 1 Wert A

`_fortfahrend_`
...

`Thresh31_B`
Kanal 31 Grenzwert B

`ReservedRange`
Reserviert

`ModbusInputRegisters_LAST`
Platzhalter für letztes Register

Bit(s)	Name	Description
7:0	`OPMODE`	Code für momentane Betriebsart

Tabelle 5.1: Status register Offset: 0x00

Value	Mode name	Description
1	DEFAULT	Standard Betriebsart (TBD)
2	PROBE	Bus Probe (Suchen nach Sensoren)
3	LOGGING	Log-Modus (Sensorenwerte aufnehmen)
4	WAIT	(reserved)
6	SIMULATOR	Simulation
7	DEBUG	Debug-Modus
14	REBOOT	(reserved)
15	SHUTDOWN	(reserved)

Tabelle 5.2: Mode `OperationMode` – possible values

Bit(s)	Name	Description
5	`ALARM_OVER_B`	Alarm Kanal B Grenzwert überschritten (reserviert)
4	`ALARM_OVER_A`	Alarm Kanal A Grenzwert überschritten
0	`DATA_VALID`	Datum gültig

Tabelle 5.3: Flags0 register Offset: 0x100

5.1.1 Grenzwerte

Die Alarmfunktion für die Firmware-Version "rc1" umfasst eine einfache Grenzwertüberprüfung. Falls der Wert in Kanal A den Schwellwert A übersteigt, wird das entsprechende Bit `ALARM_OVER_A` gesetzt. Der Grenzwert ist momental für alle Sensoren derselbe (`DefaultThreshold`, Tabelle 4.11).

5.1.2 Unterstützte Sensoren

Je nach Sensorentyp müssen die Werte unterschiedlich interpretiert werden. Dies geschieht mittels des entsprechenden Typecode (Tabelle 5.5).

Offset [Span]	Name(Id)	Access	Description
0x00	Status	RO	Status-Register
0x100 [32]	FlagsArray	RO	Flags register für 32 Sensor-Einheiten [0..31]
0x100	Flags0	RO	Flags register für Sensor 0
0x120 [64]	TypeCodeArray	RO	Typecode-Register für 64 Kanäle (32 Sensoren a 2 Kanäle)
0x120	TypeCode0	RO	Typecode-Register für Kanal 0.0/0.1
0x220 [128]	ValueArray	RO	Werteblock Sensoren 0..31, Werte 0..3 (Wert A, B; Schwellwert A, B)
0x220	Value0_A	RO	Sensor 0 Wert A
0x221	Value0_B	RO	Sensor 0 Wert B (reserviert)
0x222	Thresh0_A	RO	Grenzwert Sensor 0 Kanal A (default: 19000)
0x223	Thresh0_B	RO	Grenzwert Sensor 0 Kanal B (reserviert)
0x224	Value1_A	RO	Sensor 1 Wert A
0x225	_fortfahrend_	RO	...
0x29f	Thresh31_B	RO	Kanal 31 Grenzwert B
0x2a0 [864]	ReservedRange	RO	Reserviert
0x5ff	ModbusInputRegisters_LAST	RO	Platzhalter für letztes Register

Tabelle 5.4: Address map ModbusInputRegisters

☞ In der "rc1"-Release ist nur der CO2-Typ unterstützt und die Typecode-Array-Werte sind u.U. nicht definiert!

Bit(s)	Name	Description
15:8	GAS	Code für Gas-Typ
7:0	UNIT	Code für die Umrechnungseinheit

Tabelle 5.5: TypeCode0 register Offset: 0x120

Value	Mode name	Description
55	CO2	Kohlendioxid
56	CO	Kohlenmonoxid
95	NO	Stickoxid

Tabelle 5.6: Mode `GasTypeCode` – possible values

Value	Mode name	Description
1	ppm	Parts per Million
5	ug	Mikrogramm

Tabelle 5.7: Mode `UnitCode` – possible values

6

Technische Daten

Parameter	Werte/Bereich
Versorgungsspannung	12-26V
Stromaufnahme	ca. 500mA, max 1A bei eingestecktem USB (*TBD*)

Tabelle 6.1: Versorgungsspannung

Parameter	Werte/Bereich
Galvanische Trennung	RS485-Bus-Signale getrennt von Versorgung und I/O
RS485 Ueberspannungsschutz dauernde Belastung	7V negativ, 12V positiv
RS485 Ueberspannungsschutz, kurzzeitig	1.5 kV
RS485 Bitraten	9600-115200

Tabelle 6.2: Spezifikationen RS485-Bus

7 Fehlerszenarien

Dokumentation Fehlerszenarien

7.1 LED-Anzeige

Im Laufe des Hochstarts blinken alle LED im Herzschlag-Rhythmus, bis das System hochgefahren ist.
Falls nur die LEDs (3, 5, 6) blinken, liegt eine Entwicklerversion vor. Diese Versionen sollten nicht ausgeliefert werden.

7.1.1 Betriebsart

```
( )  ( )
( )  ( )
(5)  (6)
( )
```

Grüne LED (5) aus
Konfigurationsmodus. In diesem Modus werden keine Daten geloggt

Grüne LED blinkt
Probe-Modus, Erkennung der angeschlossenen Sensoren, oder Simulationsmodus. Nach ca. 5 s (einstellbar) geht bei erfolgreicher Erkennung das System direkt in den Logging-Modus über. Siehe auch Register Tabelle 5.2.

Grüne LED dauernd an
Logging-Modus, Daten werden laufend geloggt und weitergereicht

Rote LED (6) an
Konfigurationsfehler. Neustart oder Intervention eines Technikers erforderlich (Log-Datenbank muss ev. neu erzeugt werden).

7.1.2 GfG-Bus

```
(1)  (2)
( )  ( )
( )  ( )
( )
```

Beide LED aus
Bus nicht konfiguriert oder inaktiv, siehe auch Betriebsart

Orange LED (1) blinkt
Aktivität auf dem Bus

Rote LED (2) blinkt
Möglicher Ausfall eines Sensors oder Leitungsstörung (kurzes Blinken), längere Blinkdauer (> 0.5s) signalisiert das Auftreten eines auf dem Bus versandten Alarm-Ereignisses (Relais-Schalten) oder einen Bus-Timeout.

Rote LED dauernd an
Leitungsstörung oder keine Sensoren aktiv. D- und D+ Anschlüsse und GND-Verbindung prüfen.

7.1.3 ModBus

```
( )  ( )
(3)  (4)
( )  ( )
( )
```

Beide LED aus
Bus Slave nicht aktiv

Orange LED (3) blinkt
Aktivität auf dem Bus

Rote LED (4) blinkt
Empfangsfehler, mögliche Leitungsstörung, oder Timeout. Falls keine Aktivität erfolgt (orange LED), bleibt der Fehlerstatus erhalten.

7.1.4 WLAN

Das WLAN-Netz ist in der vorliegenden Version dauernd an. Dementsprechend blinkt die LED (7).

```
( )  ( )
( )  ( )
( )  ( )
(7)
```

7.2 Netzwerk-Zugriff

Kein modhub WLAN sichtbar
Konfigurationsfehler. Siehe Abschnitt 2.2.1.

Webseite http://modhub zeigt keine Grafik bei "Draw"
Datum eventuell nicht korrekt oder Datenbank korrupt. Abhilfe: modhub:/media/log/*.rrd löschen oder `RRDLog.Configure` Property setzen.