Treiber für verschiedene Zähler, Heizgeräte und reedartige Kontakte

Um diese Schnittstelle zu verwenden, schließen Sie das Messgerät an die verfügbaren GPIO-Pins des Geräts an. Diese GPIO müssen auf `Keine (0)` gestellt sein. Komponenten in der Tasmota-Vorlage oder im Modul. Wenn die Schnittstelle erkennt, dass ein Zählerdeskriptor GPIO mit einer Tasmota GPIO-Einstellung kollidiert, erzeugt die Schnittstelle einen "duplizierten GPIO-definierten" Fehler im Protokoll und der Zählerdeskriptor wird ignoriert.

Die Smart Meter Schnittstelle bietet die Möglichkeit, viele Arten von Zählern an Tasmota anzuschließen.

Die folgenden Arten von Zählerprotokollen werden unterstützt:.
- ASCII OBIS-Telegramme, die von vielen intelligenten Zählern und auch von der P1-Zählerschnittstelle gesendet werden.
- Binäres SML OBIS-Telegramm, das von vielen intelligenten Zählern gesendet wird.
- Binäres EBUS-Telegramm, das von vielen Heizungen und Wärmepumpen (z.B. Vaillant, Wolf) gesendet wird.
- Binäres MODBUS-Telegramm, das von vielen Leistungsmessern verwendet wird.
- Binäres RAW-Telegramm dekodiert alle Arten von Binärdaten, z.B. EMS-Heizbusse.
- Zählerschnittstelle (verwendet Tasmota-Zählerspeicher) für z.B. Reed-Kontakte entweder im Polling- oder IRQ-Modus.

Es gibt viele verschiedene Zähler, die das gleiche Protokoll verwenden. Es gibt eine Vielzahl von Varianten und Anwendungsfällen. Ein Zähler kann mit Hilfe der Kompilierungszeit `#define` pragmas definiert werden. Dies erfordert eine Neukompilierung der Firmware, um Änderungen vornehmen zu können.

Diese Schnittstelle bietet die Möglichkeit, diese Definitionen durch Zählerdeskriptoren zu spezifizieren. Diese Methode verwendet den Editor scripting language, um die Deskriptoren zu definieren. Auf diese Weise wird nur eine Firmware-Binärversion benötigt und eine Änderung kann einfach "on the fly" vorgenommen werden.

Wenn im Skript kein Abschnitt `>M` gefunden wird oder wenn die Skriptsprache nicht kompiliert wird, kehrt der Treiber zu den standardmäßigen `#define` Definition(en) zurück.

Deskriptorsyntax

> `>D`
> Dieser Abschnitt muss vorhanden sein, ist hier aber leer

Deklarieren Sie ein Skript `>B` (Boot) Abschnitt, um die Schnittstelle zu informieren, um die Zählerbeschreibung(en) auszulesen.

> `>B`
> =>sensor53](Befehle#sensor53) r

Deklarieren Sie einen Skriptabschnitt `>M` mit der Anzahl der angeschlossenen Zähler (n = `1..5`).

> `>M <n>` `

Zählerdeklaration

`+<M>,<rxGPIO>,<type>,<flag>,<parameter>,<jsonPrefix>{,<txGPIO>,<txPeriod>,<cmdTelegram>}` ``

- `<M>` - Zählernummer
- `<GPIO>` - GPIO Pin für die Eingangsdaten
- `<type>` - Zählertyp des Zählers
    - `o` = OBIS ASCII Art der Kodierung
    - `s` = SML binäre Smart-Message-Codierung (binär)
    - `c` = Zählertyp
    - `e` = EBus Binärcodierung
    - `m` = MODBus Binärcodierung

    - `r` = Roh-Binärcodierung (beliebiges Binärtelegramm)
- `<flag>` - Schalterflag für Counter
    - 0=ohne Pullup
    - 1=mit Pullup
- `<parameter>` - Parameter je nach Zählertyp
    - für `o,s,e,e,m,r` : serielle Baudrate
    - für `c` :
    - positiver Wert = Zählerabfrageintervall
    - negativer Wert = Entprellzeit (Millisekunden) für irqgesteuerte Zähler
- `` - Präfix für Web UI und MQTT JSON Payload. Bis zu 7 Zeichen
- `<GPIO>` - GPIO Pin für Ausgangsdaten
- `<txPeriod>` - Anzahl der 100ms-Schritte. Sendeinterval der Übertragung von Befehlen an den Zähler
- `<cmdTelegram>` ` - durch Komma getrennte hexadezimale Byteblöcke zum Senden an das Messgerät. Für Modbus ist z.B. jeder Block ein Befehl, um ein bestimmtes Register vom Zähler abzurufen.

Beispiele:
`+1,3,o,0,9600,OBIS`
`+1,3,m,0,9600,MODBUS,1,1,01040000,01040002,01040004,01040006,01040008,0104000a,0104000c,0104000e,01040010` `

Zählerkennzahlen

Jeder Zähler liefert typischerweise mehrere Kennzahlen (Spannung, Leistung,Strom, etc.), die er misst. Für jede zu erfassende Metrik muss ein Eintrag N (N = `1..16`) angegeben werden. Ein Eintrag definiert, wie die Daten dekodiert und in Variablen abgelegt werden.

`<M>,<Decoder>@<scale>,<Label>,<UoM>,<var>,<präzision>`

- `<M>` ` - Zählernummer, zu der dieser Decoder gehört.
- `<Decoder>` ` - Dekodierungsspezifikation. Dekodierung von OBIS als ASCII; SML, EBUS, MODBUS; RAW als HEX ASCII
    - OBIS: ASCII OBIS-Code, der mit dem Zeichen `(` Zeichen abgeschlossen wird, das den Beginn des Zählerwertes anzeigt.
    - SML: SML binäres OBIS als Hex abgeschlossen mit `0xFF` , das den Beginn des SML-codierten Wertes anzeigt.
    - EBUS,MODBUS,RAW: Hex-Werte des EBUS,MODBUS,RAW-Blocks zum Vergleich
    - `xx` bedeutet ignore Wert
    - `ss` = extrahiere dieses signierte Byte.
    - `uuu` = extrahiere dieses unsignierte Byte.
    - `uuuuuuu` = extrahiere dieses unsignierte Wort.
    - `ssssss` = dieses signierte Wort extrahieren
    - `fffffffffffff` = diesen Float-Wert extrahieren
    - `FFffffffffff` = diesen Float-Wert reverse extrahieren
    - Dekodieren eines 0/1 Bits wird durch ein `bx:` (x = `0..7`) nach dem @ Zeichen angezeigt, das das entsprechende Bit aus einem Byte extrahiert.
      z.B. 1,xxxx5017xxuu@b0:1,Solarpumpe,,Solarpumpe,Solarpumpe,0
    - im Falle von MODBus bezeichnet `ix:` den Index (x = `0..n`), der sich auf den angeforderten Block im Sendeabschnitt der Zählerdefinition bezieht. z.B. 1,010404ffffffffxxxxxx@i0:1,Spannung P1,V,Spannung_P1,2
- `@` Dekodierung Definition Abbruchzeichen
- `<scale>` - Skalierungsfaktor (Divisor)
  Dies kann ein Bruchteil (z.B. 0,1 => Ergebnis * 10) oder ein negativer Wert sein.
  Wenn Sie ein Zeichenkettenergebnis (z.B. einen seriellen Zähler) dekodieren, verwenden Sie für diesen Parameter das Zeichen `#` (nur in einer Zeile pro Meter). Für OBIS benötigen Sie ein `)` Terminierungszeichen nach dem `#` Zeichen.
- `<label>` - Web UI Label (max. 23 Zeichen)

- `<UoM>` - Mengeneinheit (max. 7 Zeichen)
- `<var>` - MQTT Variablenname (max. 23 Zeichen)
- `<präzision>` - Anzahl der Dezimalstellen
  Fügen Sie 16 hinzu, um die Daten sofort zu übertragen. Andernfalls wird es bei `TelePeriod` übertragen.

z.B. `1,1-0:1.8.0*255(@1,Verbrauch,KWh,Total_in,4`

Das Zeichen `#` beendet die Liste.

Spezielle Befehle

mit dem Zeichen '=' am Anfang einer Zeile können Sie eine spezielle Dekodierung durchführen.

- `=m` führen Sie die Arithmetik (`+,-,*,/`) durch. Verwenden Sie `#` vor einer Zahl, um einen konstanten Wert zu bestimmen.
  Beispiel:
  `=m 3+4+5/#3` Ergebnis des Decodereintrags 3,4,5 addieren und durch 3 dividieren (d.h. Durchschnitt)
- `=d` Differenz zwischen den im Zeitintervall dekodierten metrischen Werten berechnen
  Beispiel:
  `=d 3 10` berechnet 10 Sekunden Intervalldifferenz des Decodereintrags 3
- `=h` html Text (bis zu 30 Zeichen)
  fügt eine HTML-Zeile zwischen den Einträgen ein (diese Zeilen zählen nicht als Decodereintrag).