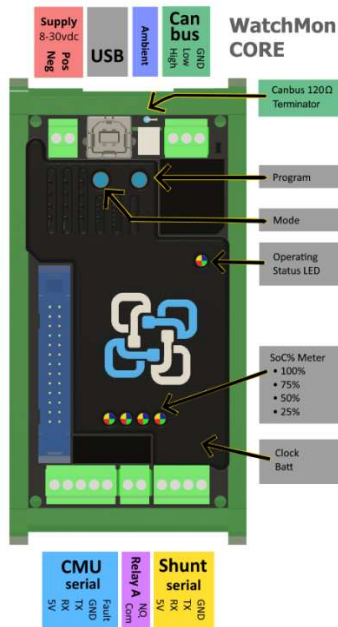


In meinem Fall wird das WatchMonCore mit einem ShuntMon 500A und 4 Stk. CellMate K9 betrieben.

An jedem CellMate K9 sind 16 Stk. EVE Power Zellen 3,2V LiFeP04 - 315Ah vorhanden.

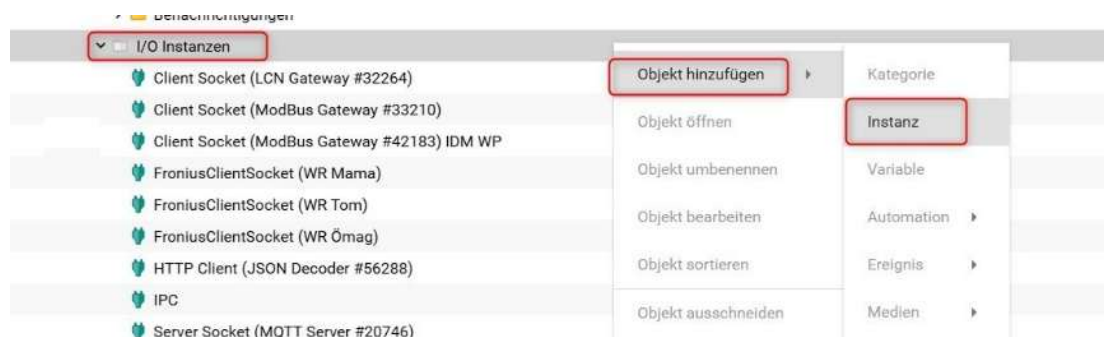
Somit ist bei allen 4 Batteriebänken eine theoretische Kapazität von 64,5 kWh angeschlossen.

Die Ladung und Entladung übernimmt ein Deye Hybrid-Wechselrichter SUN-12K-SG04LP3-EU.



Das Batrium ist im Netzwerk über Wlan mit der IP 192.168.1.173 eingebunden.

erstes wird im IP-Symcon bei den I/O Instanzen ein neuer UDP Socket erstellt





Instanz hinzufügen

Instanzen repräsentieren die verschiedenen Geräte in Ihrem Projekt

Typ:
I/O

Hersteller
- Alle Hersteller -

Schnellfilter
UDP

Hersteller	Gerät
(I/O)	UDP Socket

Modul: UDP Socket

Autor: Symcon GmbH

Name

UDP Socket

Ort

I/O Instanzen

Visuelle Einstellungen

Weitere Einstellungen

ABBRECHEN

OK

Symcon

OBJEKTBAUM X REGISTER VARIABLE BATRIUM X UDP SOCKET BATRIUM X +

Instanzkonfiguration

☒ Schnittstelle ist geöffnet

☒ Socket öffnen

Sende-Host

192.168.1.173

Sende-Port

18542

Empf.-Host

Alle

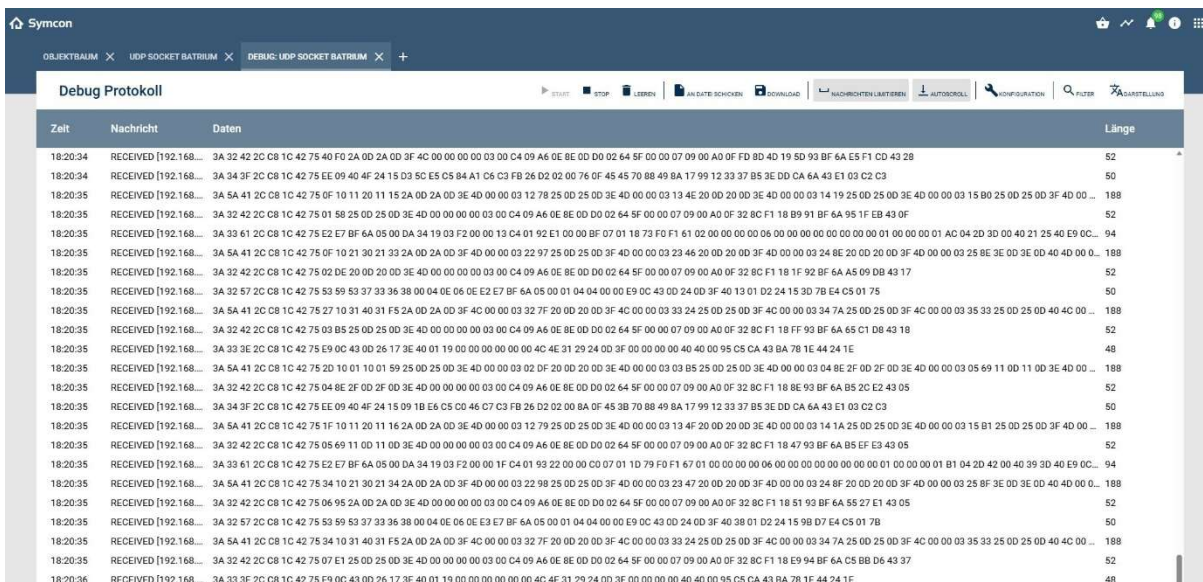
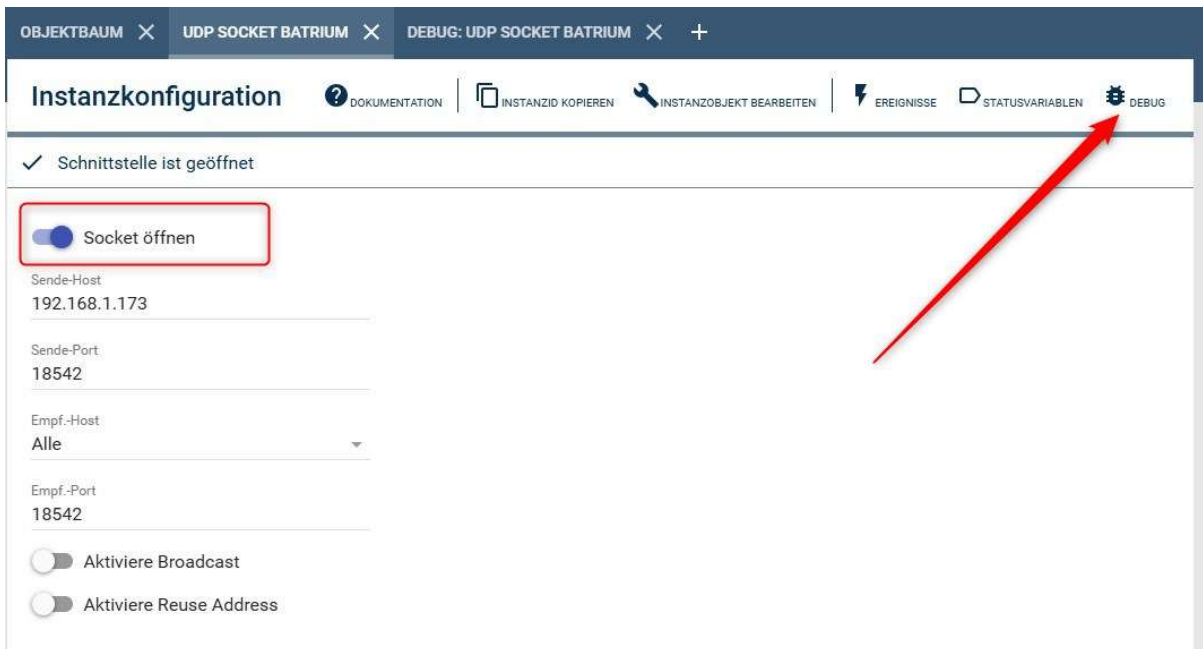
Empf.-Port

18542

☐ Aktiviere Broadcast

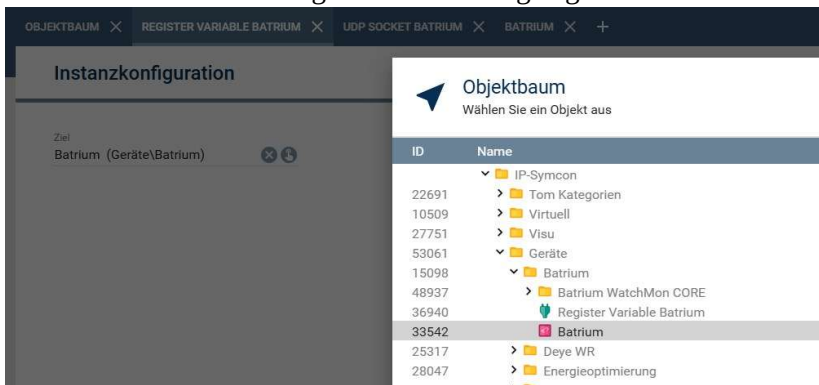
☐ Aktiviere Reuse Address

Wenn dieser angelegt ist und die Schnittstelle geöffnet ist, muss man im Debug bereits die Daten vom Batrium sehen.



Als nächstes habe ich eine neue Kategorie für das Batrium angelegt.

Darin wurde eine neue Register Variable angelegt die mit dem neue PHP-Skript verbunden ist.



Sobald dann das Skript gestartet wird, werden alle Unterkategorien mit den Werten angelegt.

Das Skript selbst, wurde mit dem Debug Log von der I/O Instanz mit der KI erstellt.

Hier mal das PHP das für meine Anlage erstellt wurde:

<?php

/*

Batrium WatchMon CORE - UDP-Auswertung fuer IP-Symcon

Einrichtung:

1. UDP Socket (Port 18542) -> Register Variable (Variablentyp: String)
2. Dieses Skript anlegen.
3. Ereignis auf die Variable der Register Variable anlegen:
Ausloeser "Bei Aktualisierung" (NICHT "Bei Aenderung"), Aktion = dieses Skript.

Ausgewertet werden (Format laut Batrium "WatchMon Wifi UDP protocol sw1.0.30",
gepueft an einem Mitschnitt deiner Firmware):

0x415A Zellen Kurzstatus (16 Zellen je Paket)

0x3E33 Systemuebersicht (Min/Max/Avg Zellspannung, Temperaturen, Zellzahlen)

0x5732 System Discovery: SoC, Shunt-Spannung, Strom, Leistung, Betriebsstatus
(Strom: + = Laden, - = Entladen)

*/

// ----- Einstellungen -----

\$cellsPerBank = 16; // Zellen je Batteriebank (4 Baenke x 16 = 64 Zellen)

\$debug = false; // true = unbekannte/fehlerhafte Pakete ins Meldungsfenster loggen

// ----- Daten holen -----

\$data = \$_IPS['VALUE'] ?? '';

if (!is_string(\$data) || \$data === '') {

return;

}

// Falls die Daten als Hex-Text ankommen ("3A 5A 41 ..."), zurueck in Binaer wandeln

if (\$data[0] !== ':' && preg_match('/^[0-9A-Fa-f\s]+\$', \$data)) {

\$bin = hex2bin(preg_replace('/\s+/', '', \$data));

if (\$bin !== false) {

\$data = \$bin;

}

}

\$len = strlen(\$data);

```

if ($len < 8 || $data[0] !== ':' || $data[3] !== ',') {
    if ($debug) {
        IPS_LogMessage('Batrium', 'Ungueltiges Paket, Laenge ' . $len);
    }
    return;
}

$type = ord($data[1]) | (ord($data[2]) << 8);
if ($type !== 0x415A && $type !== 0x3E33 && $type !== 0x5732) {
    if ($debug) {
        IPS_LogMessage('Batrium', sprintf('Nicht ausgewerteter Typ 0x%04X, %d Byte', $type, $len));
    }
    return;
}

```

// ----- Hilfsfunktionen -----

```

if (!function_exists('BAT_U16')) {

    function BAT_U16($d, $o)
    {
        return ord($d[$o]) | (ord($d[$o + 1]) << 8);
    }

    function BAT_Profile($name)
    {
        if (IPS_VariableProfileExists($name)) {
            return;
        }
        switch ($name) {
            case 'BAT.Volt':
                IPS_CreateVariableProfile($name, 2);
                IPS_SetVariableProfileText($name, ", ' V'");
                IPS_SetVariableProfileDigits($name, 3);
                break;
            case 'BAT.mV':
                IPS_CreateVariableProfile($name, 1);

```

```

    IPS_SetVariableProfileText($name, ", ' mV');

    break;
case 'BAT.Amp':
    IPS_CreateVariableProfile($name, 2);
    IPS_SetVariableProfileText($name, ", ' A');
    IPS_SetVariableProfileDigits($name, 2);

    break;
case 'BAT.Watt':
    IPS_CreateVariableProfile($name, 2);
    IPS_SetVariableProfileText($name, ", ' W');
    IPS_SetVariableProfileDigits($name, 0);

    break;
case 'BAT.SoC':
    IPS_CreateVariableProfile($name, 2);
    IPS_SetVariableProfileText($name, ", ' %');
    IPS_SetVariableProfileDigits($name, 1);
    IPS_SetVariableProfileValues($name, 0, 100, 0.5);

    break;
case 'BAT.OpStatus':
    IPS_CreateVariableProfile($name, 1);
    foreach ([0 => 'Timeout', 1 => 'Leerlauf', 2 => 'Laden', 3 => 'Entladen', 4 => 'Voll',
        5 => 'Leer', 6 => 'Simulator', 7 => 'Critical (wartend)', 8 => 'Critical (aus)',
        9 => 'MQTT offline', 10 => 'Auth Setup'] as $v => $t) {
        IPS_SetVariableProfileAssociation($name, $v, $t, ", -1);
    }

    break;
case 'BAT.ShuntStatus':
    IPS_CreateVariableProfile($name, 1);
    foreach ([0 => 'Timeout', 1 => 'Entladen', 2 => 'Leerlauf', 4 => 'Laden'] as $v => $t) {
        IPS_SetVariableProfileAssociation($name, $v, $t, ", -1);
    }

    break;
case 'BAT.PowerRate':
    IPS_CreateVariableProfile($name, 1);
    foreach ([0 => 'Aus', 2 => 'Begrenzt', 4 => 'Normal'] as $v => $t) {

```

```

        IPS_SetVariableProfileAssociation($name, $v, $t, ", -1);
    }
    break;
case 'BAT.CellStatus':
    IPS_CreateVariableProfile($name, 1);
    $assoc = [
        0 => 'Keiner', 1 => 'Spannung zu hoch', 2 => 'Temperatur zu hoch', 3 => 'OK',
        4 => 'Timeout', 5 => 'Spannung zu niedrig', 6 => 'Deaktiviert', 7 => 'Im Bypass',
        8 => 'Bypass Start', 9 => 'Bypass Ende', 10 => 'Setup fehlt', 11 => 'Keine Konfig',
        12 => 'Ausserhalb Limits', 255 => 'Undefiniert'
    ];
    foreach ($assoc as $v => $t) {
        IPS_SetVariableProfileAssociation($name, $v, $t, ", -1);
    }
    break;
}
}

```

```

function BAT_Cat($parent, $ident, $name, $pos = 0)
{
    global $BAT_IDS, $BAT_DIRTY;
    $key = 'c' . $parent . '/' . $ident;
    if (isset($BAT_IDS[$key]) && IPS_ObjectExists($BAT_IDS[$key])) {
        return $BAT_IDS[$key];
    }
    $id = @IPS_GetObjectIDByIdent($ident, $parent);
    if ($id === false) {
        $id = IPS_CreateCategory();
        IPS_SetParent($id, $parent);
        IPS_SetIdent($id, $ident);
        IPS_SetName($id, $name);
        IPS_SetPosition($id, $pos);
    }
    $BAT_IDS[$key] = $id;
    $BAT_DIRTY = true;
}

```

```

    return $id;
}

// Variable holen bzw. anlegen und Wert nur bei Aenderung schreiben
// $type: 0 = Bool, 1 = Integer, 2 = Float, 3 = String
function BAT_Set($parent, $ident, $name, $type, $profile, $value, $pos = 0)
{
    global $BAT_IDS, $BAT_DIRTY;
    $key = 'v' . $parent . '/' . $ident;
    $id = false;
    if (isset($BAT_IDS[$key]) && IPS_VariableExists($BAT_IDS[$key])) {
        $id = $BAT_IDS[$key];
    } else {
        $id = @IPS_GetObjectIDByIdent($ident, $parent);
        if ($id === false) {
            if ($profile !== '' && strpos($profile, 'BAT.') === 0) {
                BAT_Profile($profile);
            }
            $id = IPS_CreateVariable($type);
            IPS_SetParent($id, $parent);
            IPS_SetIdent($id, $ident);
            IPS_SetName($id, $name);
            IPS_SetPosition($id, $pos);
            if ($profile !== '') {
                IPS_SetVariableCustomProfile($id, $profile);
            }
        }
        $BAT_IDS[$key] = $id;
        $BAT_DIRTY = true;
    }
    if (GetValue($id) != $value) {
        SetValue($id, $value);
    }
}
}

```

```

// ----- Struktur / ID-Cache -----

$parentOfScript = IPS_GetParent($_IPS['SELF']);
$root = @IPS_GetObjectIDByIdent('BatriumRoot', $parentOfScript);
if ($root === false) {
    $root = IPS_CreateCategory();
    IPS_SetParent($root, $parentOfScript);
    IPS_SetIdent($root, 'BatriumRoot');
    IPS_SetName($root, 'Batrium WatchMon CORE');
}
$cacheVar = @IPS_GetObjectIDByIdent('IdCache', $root);
if ($cacheVar === false) {
    $cacheVar = IPS_CreateVariable(3);
    IPS_SetParent($cacheVar, $root);
    IPS_SetIdent($cacheVar, 'IdCache');
    IPS_SetName($cacheVar, 'ID-Cache (intern)');
    IPS_SetHidden($cacheVar, true);
}
$BAT_IDS = json_decode(GetValueString($cacheVar), true);
if (!is_array($BAT_IDS)) {
    $BAT_IDS = [];
}
$BAT_DIRTY = false;

// ----- 0x415A: Zellen Kurzstatus -----
if ($type === 0x415A) {
    // Offset 8 = CMU-Port RX Node-ID (variabel, KEINE Recordlaenge), 9 = Anzahl Records
    $recLen = 11;
    $count = ord($data[9]);
    if ($recLen < 11 || $len < 12 + $count * $recLen) {
        if ($debug) {
            IPS_LogMessage('Batrium', "0x415A unvollstaendig: Laenge $len, Records $count x $recLen");
        }
        return;
    }
}

```

```

$bankVolts = [];

for ($i = 0; $i < $count; $i++) {
    $o    = 12 + $i * $recLen;
    $node = ord($data[$o]);
    if ($node < 1) {
        continue;
    }
    $vMin = BAT_U16($data, $o + 2);
    $vMax = BAT_U16($data, $o + 4);
    $temp = ord($data[$o + 6]) - 40;
    $bTemp = ord($data[$o + 7]) - 40;
    $bmA    = BAT_U16($data, $o + 8);
    $stat    = ord($data[$o + 10]);

    $bank    = intdiv($node - 1, $cellsPerBank) + 1;
    $cell     = ($node - 1) % $cellsPerBank + 1;
    $bankId   = BAT_Cat($root, 'Bank' . $bank, 'Bank ' . $bank, $bank);
    $p        = 'Z' . sprintf('%02d', $cell);
    $pos      = 10 + $cell * 10;

    BAT_Set($bankId, $p . '_V', "Zelle $cell Spannung", 2, 'BAT.Volt', round($vMax / 1000, 3), $pos);
    BAT_Set($bankId, $p . '_T', "Zelle $cell Temperatur", 2, '~Temperature', (float)$temp, $pos + 1);
    BAT_Set($bankId, $p . '_S', "Zelle $cell Status", 1, 'BAT.CellStatus', $stat, $pos + 2);
    BAT_Set($bankId, $p . '_B', "Zelle $cell Bypass", 1, "", $bmA, $pos + 3);

    $bankVolts[$bank][$node] = $vMax;
}

// Bank-Statistik, sobald alle Zellen einer Bank im Paket waren
foreach ($bankVolts as $bank => $cells) {
    if (count($cells) < $cellsPerBank) {
        continue;
    }
    $bankId = BAT_Cat($root, 'Bank' . $bank, 'Bank ' . $bank, $bank);
    $min = min($cells);
    $max = max($cells);

```

```

    BAT_Set($bankId, 'Summe', 'Bankspannung (Summe Zellen)', 2, 'BAT.Volt', round(array_sum($cells) / 1000, 3),
1);
    BAT_Set($bankId, 'Min', 'Zelle min', 2, 'BAT.Volt', round($min / 1000, 3), 2);
    BAT_Set($bankId, 'Max', 'Zelle max', 2, 'BAT.Volt', round($max / 1000, 3), 3);
    BAT_Set($bankId, 'Diff', 'Zellspreizung', 1, 'BAT.mV', $max - $min, 4);
}
}

// ----- 0x3E33: Systemuebersicht -----
if ($type === 0x3E33) {
    if ($len < 37) {
        if ($debug) {
            IPS_LogMessage('Batrium', "0x3E33 zu kurz: $len");
        }
        return;
    }
    $sys = BAT_Cat($root, 'System', 'System', 0);
    $vMin = BAT_U16($data, 8);
    $vMax = BAT_U16($data, 10);
    $vAvg = BAT_U16($data, 28);

    BAT_Set($sys, 'VMin', 'Zellspannung min', 2, 'BAT.Volt', round($vMin / 1000, 3), 1);
    BAT_Set($sys, 'VMax', 'Zellspannung max', 2, 'BAT.Volt', round($vMax / 1000, 3), 2);
    BAT_Set($sys, 'VAvg', 'Zellspannung Durchschnitt', 2, 'BAT.Volt', round($vAvg / 1000, 3), 3);
    BAT_Set($sys, 'VDiff', 'Zellspreizung', 1, 'BAT.mV', $vMax - $vMin, 4);
    BAT_Set($sys, 'VMinNode', 'Zelle mit Min-Spannung (Node)', 1, "", ord($data[12]), 5);
    BAT_Set($sys, 'VMaxNode', 'Zelle mit Max-Spannung (Node)', 1, "", ord($data[13]), 6);
    BAT_Set($sys, 'TMin', 'Zelltemperatur min', 2, '~Temperature', (float)(ord($data[14]) - 40), 7);
    BAT_Set($sys, 'TMax', 'Zelltemperatur max', 2, '~Temperature', (float)(ord($data[15]) - 40), 8);
    BAT_Set($sys, 'TAvg', 'Zelltemperatur Durchschnitt', 2, '~Temperature', (float)(ord($data[30]) - 40), 9);
    BAT_Set($sys, 'CntInitBypass', 'Zellen ueber Start-Bypass', 1, "", ord($data[31]), 10);
    BAT_Set($sys, 'CntFinalBypass', 'Zellen ueber End-Bypass', 1, "", ord($data[32]), 11);
    BAT_Set($sys, 'CntBypass', 'Zellen im Bypass', 1, "", ord($data[33]), 12);
    BAT_Set($sys, 'CntOverdue', 'Zellen ueberfaellig', 1, "", ord($data[34]), 13);
    BAT_Set($sys, 'CntActive', 'Zellen aktiv', 1, "", ord($data[35]), 14);
    BAT_Set($sys, 'CntSystem', 'Zellen im System', 1, "", ord($data[36]), 15);

```

```

}

// ----- 0x5732: System Discovery (SoC, Shunt, Betriebsstatus) -----
if ($type === 0x5732) {
    if ($len < 50) {
        if ($debug) {
            IPS_LogMessage('Batrium', "0x5732 zu kurz: $len");
        }
        return;
    }

    $sys = BAT_Cat($root, 'System', 'System', 0);

    $soc = ord($data[41]) * 0.5 - 5;           // 0,5 %/Bit, Offset -5 %
    $volt = BAT_U16($data, 42) / 100;         // Shunt-Spannung (Faktor 100 laut Setup)
    $f = unpack('g', substr($data, 44, 4));  // Strom in mA, + Laden / - Entladen
    $amp = $f[1] / 1000;
    $watt = $volt * $amp;
    $time = unpack('V', substr($data, 20, 4));

    BAT_Set($sys, 'SoC', 'Ladezustand (SoC)', 2, 'BAT.SoC', round($soc, 1), 20);
    BAT_Set($sys, 'ShuntVolt', 'Batteriespannung (Shunt)', 2, 'BAT.Volt', round($volt, 2), 21);
    BAT_Set($sys, 'ShuntAmp', 'Batteriestrom (+Laden / -Entladen)', 2, 'BAT.Amp', round($amp, 2), 22);
    BAT_Set($sys, 'ShuntWatt', 'Batterieleistung (+Laden / -Entladen)', 2, 'BAT.Watt', round($watt, 0), 23);
    BAT_Set($sys, 'ShuntStatus', 'Shunt Status', 1, 'BAT.ShuntStatus', ord($data[48]), 24);
    BAT_Set($sys, 'OpStatus', 'Systemstatus', 1, 'BAT.OpStatus', ord($data[24]), 25);
    BAT_Set($sys, 'BattOk', 'Critical Battery OK', 0, "", ord($data[26]) === 1, 26);
    BAT_Set($sys, 'ChargeRate', 'Laden erlaubt', 1, 'BAT.PowerRate', ord($data[27]), 27);
    BAT_Set($sys, 'DischargeRate', 'Entladen erlaubt', 1, 'BAT.PowerRate', ord($data[28]), 28);
    BAT_Set($sys, 'DeviceTime', 'Geraetezeit (UTC)', 1, '~UnixTimestamp', (int)$time[1], 29);
}

// ----- Cache sichern -----
if ($BAT_DIRTY) {
    SetValueString($cacheVar, json_encode($BAT_IDS));
}

```

| | | | | | | | |
|-------|----|---|---------|----------------------|------------|------------|--|
| 15098 | 0 | ▼ Batrium | | | | | |
| 48937 | 0 | ▼ Batrium WatchMon CORE | | | | | |
| 39038 | 0 | ▼ System | | | | | |
| 42100 | 1 | 🔌 Zellspannung min | Float | 3,300 V | 18:39:37 | 18:39:37 | |
| 44964 | 2 | 🔌 Zellspannung max | Float | 3,310 V | 18:38:27 | 18:38:27 | |
| 30787 | 3 | 🔌 Zellspannung Durchschnitt | Float | 3,305 V | 18:39:09 | 18:39:09 | |
| 43409 | 4 | 🔌 Zellspreizung | Integer | 10 mV | 18:39:37 | 18:39:37 | |
| 58276 | 5 | 🔌 Zelle mit Min-Spannung (Node) | Integer | 2 | 18:30:35 | 18:30:35 | |
| 45153 | 6 | 🔌 Zelle mit Max-Spannung (Node) | Integer | 1 | 18:38:27 | 18:38:27 | |
| 22516 | 7 | 🔌 Zelltemperatur min | Float | 22,0 °C | 12:25:10 | 12:25:10 | |
| 43833 | 8 | 🔌 Zelltemperatur max | Float | 24,0 °C | 15:19:34 | 15:19:34 | |
| 57167 | 9 | 🔌 Zelltemperatur Durchschnitt | Float | 22,0 °C | 18:33:53 | 18:33:53 | |
| 56387 | 10 | 🔌 Zellen ueber Start-Bypass | Integer | 0 | 18:18:49 | 18:18:49 | |
| 59490 | 11 | 🔌 Zellen ueber End-Bypass | Integer | 0 | 18:18:49 | 18:18:49 | |
| 11745 | 12 | 🔌 Zellen im Bypass | Integer | 0 | 18:18:49 | 18:18:49 | |
| 20846 | 13 | 🔌 Zellen ueberfaellig | Integer | 0 | Nie | Nie | |
| 52188 | 14 | 🔌 Zellen aktiv | Integer | 64 | 01.10.2026 | 01.10.2026 | |
| 46420 | 15 | 🔌 Zellen im System | Integer | 64 | 01.10.2026 | 01.10.2026 | |
| 13950 | 20 | 🔌 Ladezustand (SoC) | Float | 99,5 % | 18:23:15 | 18:23:15 | |
| 28992 | 21 | 🔌 Batteriespannung (Shunt) | Float | 53,15 V | 18:41:04 | 18:41:04 | |
| 46020 | 22 | 🔌 Batteriestrom (+Laden / -Entladen) | Float | -18,32 A | 18:41:04 | 18:41:04 | |
| 19049 | 23 | 🔌 Batterieleistung (+Laden / -Entladen) | Float | -974 W | 18:41:04 | 18:41:04 | |
| 38221 | 24 | 🔌 Shunt Status | Integer | Entladen | 18:06:24 | 18:06:24 | |
| 57492 | 25 | 🔌 Systemstatus | Integer | Leer | 17:59:25 | 17:59:25 | |
| 26525 | 26 | 🔌 Critical Battery OK | Boolean | An | 01.10.2026 | 01.10.2026 | |
| 53107 | 27 | 🔌 Laden erlaubt | Integer | Normal | 01.10.2026 | 01.10.2026 | |
| 27265 | 28 | 🔌 Entladen erlaubt | Integer | Normal | 01.10.2026 | 01.10.2026 | |
| 37761 | 29 | 🔌 Geratezeit (UTC) | Integer | 02.10.2026, 19:41:03 | 18:41:04 | 18:41:04 | |

| | | | | | | | |
|-------|---|-----------------------------|-------------------|----------------------------|------------|------------|--|
| 15098 | 0 | ▼ Batrium | | | | | |
| 48937 | 0 | ▼ Batrium WatchMon CORE | | | | | |
| 39038 | 0 | > System | | | | | |
| 11499 | 0 | 🔌 ID-Cache (intern) | String | {\"c48937/System\":3903... | 01.10.2026 | 01.10.2026 | |
| 56040 | 1 | > Bank 1 | | | | | |
| 44943 | 2 | > Bank 2 | | | | | |
| 14602 | 3 | > Bank 3 | | | | | |
| 55947 | 4 | > Bank 4 | | | | | |
| 36940 | 0 | 🔌 Register Variable Batrium | Register Variable | | | 01.10.2026 | |
| 33542 | 0 | 🔌 Batrium | PHP-Skript | 33542.ips.php | 18:41:23 | 01.10.2026 | |
| 25317 | 0 | > Deye WR | | | | | |



