```
-- ********** SETTINGS
************************************************************


function round(x, n)
   n = math.pow(10, n or 0)
   x = x * n
   if x >= 0 then x = math.floor(x + 0.5) else x = math.ceil(x - 0.5) end
   return x / n
end


-- Seconds to wait until next Modbus Query sec.
local Wait = 5;


-- ********** MODBUS PARAMETER
**************************************************
-- Read PV-Inverter Address and Modbus Port out of Virtual Device Settings
local thisdevice = fibaro:getSelfId()
--local InvIPadress = ("192.168.1.53")
local InvPort = fibaro:getValue(thisdevice, 'TCPPort')


-- A Modbus Application Header  is added to the start of the
-- TCP message. This header has the following data:


--Transaction Identifier (2 Bytes)
--Protocol Identifier (2 Bytes)( Ist immer 0 (Modbus Protokoll)
--Bytes to follow (2 Bytes)
--Slave ID or Unit Identifier (2 Bytes) -> Fronius: Unit ID (1 Byte): Dieses Feld wird zur Adressierung der an den Fronius Datamanager
  --angeschlossenen Geräte verwendet (Gateway-Funktion des Fronius Datamanagers).
  --Die Unit ID entspricht der Slave ID bei Modbus RTU. Der Wert wird vom Master vor-
  --gegeben und wird vom Slave unverändert mit der Antwort zurückgegeben.
```

```
--Function Code (2 Bytes)
--Register Adress (2 Bytes)
--Bytes to Read (2 Bytes)

-- Transaction Identifier, set by the Client to
-- uniquely identify each request. Default Value = 0
local TransId = 0;

-- Protocol Identifier, set by the Client.
-- Default Value = 0
local ProtId = 0;

-- Length: identifying the number of bytes
-- in the message to follow. Default Value = 6
local BytesToFlw = 6;

-- The SlaveID Address. SMA Inverter Default Value = 3
-- SunSpec Modbus Value = 126
local UnitIdentifier = 1;

-- The functions below are used to access the registers
-- outlined in the register map of the module for sending
-- and receiving data.
-------------------------------------------------------------
-- CODE         FUNCTION                                    REFERENCE
-- 01 (01H)     Read Coil (Output) Status          0xxxx
-- 03 (03H) Read Holding Registers                 4xxxx
-- 04 (04H)     Read Input Registers               3xxxx
-- 05 (05H)     Force Single Coil (Output)         0xxxx
-- 06 (06H)     Preset Single Register             4xxxx
-- 08 (08H)     Reset Slave                                 Hidden
```

```
-- 15 (0FH)        Force Multiple Coils (Outputs)   0xxxx
-- 16 (10H)        Preset Multiple Registers                  4xxxx
-- 17 (11H)        Report Slave ID                                        Hidden
------------------------------------------------------------
-- Default Value = 3 cause we just want to read data
local FunctionCode = 03;

-- The Data Address of the actual modbus register requested.
-- For Details see your Fronius manual with available registers!

local RegisterAdd;

-- The total number of registers requested. Default Value = 2
--local NumRegToRead = 1;

-- Most of the Parameters above are 2 bytes. So we have to
-- split the numbers into high and low byte,
-- see function below!
local TransIdHB;                --high byte Transaction Identifier
local TransIdLB;          --low byte Transaction Identifier
local ProtIdHB;                 --high byte Protocol Identifier
local ProtIdLB;                 --low byte Protocol Identifier
local BytesToFlwHB;     --high byte Bytes to follow
local BytesToFlwLB;             --low byte Bytes to follow
local StartAddHB;               --high byte Start Address
local StartAddLB;               --low byte Start Address
local NumRegToReadHB;           --high byte Number of Registers to read
local NumRegToReadLB;           --low byte Number of Registers to read

-- The Modbus Answer will be divided by that factor.
-- Valid Options: 1, 10, 100, 1000! Default Value = 1
```

```
local Factor = 1;



-- ********** FUNCTIONS **********

-- (1) - SPLIT NUMBER IN HIGH AND LOW BYTE FUCNTION----------
-- One byte has 8 bits and therefore 2^8=256 different states.
-- Numbers greater than 256 are represented in the next
-- higher byte! Formula: X = Y * (256) + Z
-- Example: 30051 = 177*(256) + 99
-- 30051 will be 117 high byte and 99 low byte

local function highbyte(regnum)
   local hb = 0;
          hb = math.floor(regnum/ 256);
          return hb;
end

local function lowbyte(regnum)
          local lb = 0;
          lb = regnum% 256;
          return lb;
end

-- (2) - DUMP STRING ----------------------------------------
-- Conversion to readable String
local function DumpString(str,spacer)
          return (
          string.gsub(str,"(.)",
      function (c)
          return string.format("%02X%s",string.byte(c), spacer or "");
```

```
      end)
   )
end



-- (3) - MODBUS INTERACTION ---------------------------------
-- Reads the Modbus Register and returns Value divided
-- Example Input (30051, 1) returns 8001
local function readRegister (modbusRegister, divider)
          local modbusAnswerRaw;
          local modbusAnswerNr;
          --Splitting in high and low byte:
          TransIdHB = tostring(highbyte(TransId));
          TransIdLB = tostring(lowbyte(TransId));
          ProtIdHB = tostring(highbyte(ProtId));
          ProtIdLB = tostring(lowbyte(ProtId));
          BytesToFlwHB = tostring(highbyte(BytesToFlw));
          BytesToFlwLB = tostring(lowbyte(BytesToFlw));
          StartAddHB = tostring(highbyte(modbusRegister));
          StartAddLB = tostring(lowbyte(modbusRegister));
          NumRegToReadHB = tostring(highbyte(NumRegToRead));
          NumRegToReadLB = tostring(lowbyte(NumRegToRead));
          UnitIdentifier = tostring(UnitIdentifier);
          FunctionCode = tostring(FunctionCode);

          --Connecting to Inverter
          tcpSocket = Net.FTcpSocket(InvIPadress, InvPort);

          sendbuffer=string.char(TransIdHB,TransIdLB,ProtIdHB,ProtIdLB,BytesToFlwHB,BytesToFlwLB,
UnitIdentifier,FunctionCode,StartAddHB,StartAddLB,NumRegToReadHB,NumRegToReadLB);

 --   fibaro:debug("sendbuffer: "..sendbuffer.." buffer");
```

```
tcpSocket:setReadTimeout(2000);
        --Wait to be sure connection is established! (1000=1sec)
        fibaro:sleep(250);
        --Send Query to modbus
        local bytes, errorCode = tcpSocket:write(sendbuffer);
        --Wait to be sure Modbus is answering! (1000=1sec)
        fibaro:sleep(500);
        --Read answer of modbus
        local result, err = tcpSocket:read();
        if err == 0 then
        --String Extract
        modbusAnswerRaw = string.sub((DumpString(result)), 19);
        modbusAnswerNr = tonumber(modbusAnswerRaw, 16);

   --fibaro:debug("modbusAnswerRaw: "..modbusAnswerRaw);
   --fibaro:debug("modbusAnswerNr: "..modbusAnswerNr);
     -- Divide by Factor and avoid division by zero
                if modbusAnswerNr > 0 and divider > 0 then
                        modbusAnswerNr = modbusAnswerNr / divider;
       end
        else
        fibaro:debug("Error!")
                fibaro:debug(err)
   modbusAnswerNr = 0
        end
        --Disconnecting Inverter
        tcpSocket:disconnect();

        return modbusAnswerNr;
```

```
end


------------------------------------------------------------
-- ########  MAIN PROGRAM PART ################################
------------------------------------------------------------

-- Endless Loop
while true do
 fibaro:debug("Start odczytu Modbus")
   NumRegToRead = 1;
         -- Register (40084 - 1) returns AC active power across all phases in Watt
         RegisterAdd = 40274
         Factor=1;
   InvIPadress = ("192.168.1.52")
         PvPowerWatt = readRegister (RegisterAdd,Factor);
   PvPowerWatt1 = PvPowerWatt
   fibaro:sleep(2*1000)
 ------------------------------- DC Current i Voltage Fal 1
   RegisterAdd = 40272
         Factor=1;
   InvIPadress = ("192.168.1.52")
         DCA1 = readRegister (RegisterAdd,Factor);
   if DCV1 ~= 66535 then
   fibaro:debug("DCA String 1: "..(DCA1/100).." Amper");
   end
   fibaro:sleep(2*1000)

   RegisterAdd = 40273
         Factor=1;
```

```
InvIPadress = ("192.168.1.52")
        DCV1 = readRegister (RegisterAdd,Factor);
if DCV1 ~= 66535 then
fibaro:debug("DCV String 1: "..(DCV1/100).." Volt");
end
fibaro:sleep(2*1000)

-------------------------------- DC Current i Voltage Fal 2 String 1
RegisterAdd = 40272
        Factor=1;
InvIPadress = ("192.168.1.53")
        DCA21 = readRegister (RegisterAdd,Factor);
if DCA21 ~= 66535 then
fibaro:debug("DCA String 21: "..(DCA21/100).." Amper");
end
fibaro:sleep(2*1000)

RegisterAdd = 40273
        Factor=1;
InvIPadress = ("192.168.1.53")
        DCV21 = readRegister (RegisterAdd,Factor);
if DCV2 ~= 66535 then
fibaro:debug("DCV String 21: "..(DCV21/100).." Volt");
end
fibaro:sleep(2*1000)

-------------------------------- DC Current i Voltage Fal 2 String 2
RegisterAdd = 40292
        Factor=1;
InvIPadress = ("192.168.1.53")
        DCA22 = readRegister (RegisterAdd,Factor);
```

```
  fibaro:debug("DCA String 22: "..(DCA22/100).." Amper");
  fibaro:sleep(2*1000)

  RegisterAdd = 40293
        Factor=1;
  InvIPadress = ("192.168.1.53")
        DCV22 = readRegister (RegisterAdd,Factor);
  fibaro:debug("DCV String 22: "..(DCV22/100).." Volt");
  fibaro:sleep(2*1000)
-----------------------------------------------------------
-----------------------------------------------------------
  RegisterAdd = 40257
  Factor=1;
  InvIPadress = ("192.168.1.52")
        local AC_Power_Scale_factor_1 = readRegister (RegisterAdd,Factor);
  AC_Power_Scale_factor_1 = 65536-AC_Power_Scale_factor_1
        fibaro:debug("AC_Power_Scale_factor_1: "..AC_Power_Scale_factor_1.." jednostek");
  PvPowerWatt1 = PvPowerWatt1 / (10 ^ AC_Power_Scale_factor_1)
  --PvPowerWatt1 = round(PvPowerWatt1, 0)
  fibaro:debug("PV-Leistung 1: "..PvPowerWatt.." Watt");
  fibaro:sleep(2*1000)

--FALOWNIK 2
  --fibaro:debug("StartAddHB: "..StartAddHB);
  --fibaro:debug("StartAddLB: "..StartAddLB);
  --fibaro:debug("NumRegToReadHB: "..NumRegToReadHB);
  --fibaro:debug("NumRegToReadLB: "..NumRegToReadLB);
  InvIPadress = ("192.168.1.53")
  RegisterAdd = 40274
  NumRegToRead = 1;
  Factor=1;
```

```
        PvPowerWatt = readRegister (RegisterAdd,Factor);
        fibaro:debug("PV-Leistung 2: "..PvPowerWatt.." Watt");
  PvPowerWatt2 = PvPowerWatt
  fibaro:sleep(2*1000)

  RegisterAdd = 40294
  PvPowerWatt = readRegister (RegisterAdd,Factor);
        fibaro:debug("PV-Leistung 3: "..PvPowerWatt.." Watt");
  PvPowerWatt3 = PvPowerWatt
  fibaro:sleep(2*1000)

  RegisterAdd = 40257
  Factor=1;
  InvIPadress = ("192.168.1.53")
        local AC_Power_Scale_factor_23 = readRegister (RegisterAdd,Factor);
  AC_Power_Scale_factor_23 = 65536-AC_Power_Scale_factor_23
        fibaro:debug("AC_Power_Scale_factor_23: "..AC_Power_Scale_factor_23.." jedn");
  PvPowerWatt2 = PvPowerWatt2 / (10 ^ AC_Power_Scale_factor_23)
  PvPowerWatt3 = PvPowerWatt3 / (10 ^ AC_Power_Scale_factor_23)
 --  PvPowerWatt3 = PvPowerWatt3 / (AC_Power_Scale_factor_23 *10)
  --PvPowerWatt2 = round(PvPowerWatt2, 0)
  --PvPowerWatt3 = round(PvPowerWatt3, 0)
  fibaro:sleep(2*1000)
  --PvPowerWatt1 = PvPowerWatt1/AC_Power_Scale_factor_1
  --PvPowerWatt2 = PvPowerWatt2/AC_Power_Scale_factor_23
  --PvPowerWatt3 = PvPowerWatt3/AC_Power_Scale_factor_23

local sep='\226\142\170'
PvPowerWatt1 = round(PvPowerWatt1,0)
PvPowerWatt2 = round(PvPowerWatt2,0)
PvPowerWatt3 = round(PvPowerWatt3,0)
```

```
-- Output to the device.
  if
  (PvPowerWatt1 ~= 655 and PvPowerWatt2 ~= 655 and PvPowerWatt3 ~= 655) then
  fibaro:call(thisdevice, "setProperty", "ui.Label1.value", PvPowerWatt1.." W "..sep..PvPowerWatt2.." W "..sep..PvPowerWatt3.." W")
  end

 --między godz 22 i 03 nie zapisywać mocy zero
if (
 ( (tonumber(os.date("%H%M")) >= tonumber(string.format("%02d%02d", "03", "00")) and tonumber(os.date("%H%M")) < tonumber(string.format("%02d%02d", "22", "00"))) and (PvPowerWatt1 ~= 0 and PvPowerWatt2 ~= 0 and PvPowerWatt3 ~= 0))
)
then

 local updatechart = Net.FHttp("192.168.1.13")
   if PvPowerWatt1 ~= 655 and PvPowerWatt1 < 4000 and PvPowerWatt1 >= 0 then
                payload = "/data_post_stringi.php?id=" .. 1 .. "&value=" .. PvPowerWatt1
                fibaro:debug(payload)
                response, status, errorCode = updatechart:GET(payload);
   end

   if PvPowerWatt2 ~= 655 and PvPowerWatt2 < 4500 and PvPowerWatt2 >= 0 then
                payload = "/data_post_stringi.php?id=" .. 2 .. "&value=" .. PvPowerWatt2
                fibaro:debug(payload)
                response, status, errorCode = updatechart:GET(payload);
   end

   if PvPowerWatt3 ~= 655 and PvPowerWatt3 < 3000 and PvPowerWatt3 >= 0 then
     payload = "/data_post_stringi.php?id=" .. 3 .. "&value=" .. PvPowerWatt3
                fibaro:debug(payload)
```

```
                response, status, errorCode = updatechart:GET(payload);
    end
fibaro:debug("Zapisano moce stringów do MySQL")
end

if (
 ( (tonumber(os.date("%H%M")) >= tonumber(string.format("%02d%02d", "03", "00")) and
tonumber(os.date("%H%M")) < tonumber(string.format("%02d%02d", "22", "00"))))
)
then
   local updatechart = Net.FHttp("192.168.1.13")
    if DCA1 ~= 65535 and DCA1 >= 0 then
      payload = "/data_post_stringi.php?id=" .. 51 .. "&value=" .. (DCA1/100)
                fibaro:debug(payload)
                response, status, errorCode = updatechart:GET(payload);
    end

    if DCV1 ~= 65535 and DCV1 >= 0 then
      payload = "/data_post_stringi.php?id=" .. 61 .. "&value=" .. (DCV1/100)
                fibaro:debug(payload)
                response, status, errorCode = updatechart:GET(payload);
    end

    if DCA21 ~= 65535 and DCA21 >= 0 then
      payload = "/data_post_stringi.php?id=" .. 52 .. "&value=" .. (DCA21/100)
                fibaro:debug(payload)
                response, status, errorCode = updatechart:GET(payload);
    end

    if DCV21 ~= 65535 and DCV21 >= 0 then
      payload = "/data_post_stringi.php?id=" .. 62 .. "&value=" .. (DCV21/100)
```

```
                    fibaro:debug(payload)
                    response, status, errorCode = updatechart:GET(payload);
    end

   if DCA22 ~= 65535 and DCA22 >= 0 then
      payload = "/data_post_stringi.php?id=" .. 53 .. "&value=" .. (DCA22/100)
                    fibaro:debug(payload)
                    response, status, errorCode = updatechart:GET(payload);
    end

    if DCV22 ~= 65535 and DCV22 >= 0 then
      payload = "/data_post_stringi.php?id=" .. 63 .. "&value=" .. (DCV22/100)
                    fibaro:debug(payload)
                    response, status, errorCode = updatechart:GET(payload);
    end

    fibaro:debug("Zapisano napięcia i prądy stringów do MySQL")
end

 --fibaro:call(selfId, "setProperty", "ui.updated.value", timestamp)
------------------------------------------------------------------------
 -- LIFETIME
-- Register 40276 - 1 -> returns AC Lifetime Energy

   NumRegToRead = 2;
          RegisterAdd = 40275
          Factor=1;
   InvIPadress = ("192.168.1.52")
          local PvPowerWatt = readRegister (RegisterAdd,Factor);
          fibaro:debug("Lifetime Energy String 1: "..(PvPowerWatt/1000).." kWh");
   Lifetime_Energy_1 = (PvPowerWatt/1000)
```

```
 fibaro:sleep(1*1000)
 --fibaro:debug("StartAddHB: "..StartAddHB);
 --fibaro:debug("StartAddLB: "..StartAddLB);
 --fibaro:debug("NumRegToReadHB: "..NumRegToReadHB);
 --fibaro:debug("NumRegToReadLB: "..NumRegToReadLB);
 InvIPadress = ("192.168.1.53")
       PvPowerWatt = readRegister (RegisterAdd,Factor);
       fibaro:debug("Lifetime Energy String 2: "..(PvPowerWatt/1000).." kWh");
 Lifetime_Energy_2 = (PvPowerWatt/1000)
 fibaro:sleep(1*1000)

 RegisterAdd = 40295
 local PvPowerWatt = readRegister (RegisterAdd,Factor);
 Lifetime_Energy_3 = (PvPowerWatt/1000)
       fibaro:debug("Lifetime Energy String 3: "..(PvPowerWatt/1000).." kWh");

-- Grab the todays value before new day starts
if os.date("%H:%M") == "23:59" then
 --- wysłać na serwer
 fibaro:setGlobal("Lifetime_Energy_1", Lifetime_Energy_1)
 fibaro:setGlobal("Lifetime_Energy_2", Lifetime_Energy_2)
 fibaro:setGlobal("Lifetime_Energy_3", Lifetime_Energy_3)
end

Prod_1_ToDay = Lifetime_Energy_1 - fibaro:getGlobal ("Lifetime_Energy_1")
Prod_2_ToDay = Lifetime_Energy_2 - fibaro:getGlobal ("Lifetime_Energy_2")
Prod_3_ToDay = Lifetime_Energy_3 - fibaro:getGlobal ("Lifetime_Energy_3")
Prod_1_ToDay = round(Prod_1_ToDay,1)
Prod_2_ToDay = round(Prod_2_ToDay,1)
Prod_3_ToDay = round(Prod_3_ToDay,1)
```

```
 fibaro:call(thisdevice, "setProperty", "ui.Label2.value", Prod_1_ToDay.." kWh
"..sep..Prod_2_ToDay.." kWh"..sep..Prod_3_ToDay.." kWh")


 --- Zapis do bazy


local currentDate = os.date("*t");

-- zapis między 4 a 23

--if (

--  (tonumber(os.date("%H%M")) >= tonumber(string.format("%02d%02d", "03", "00")) and
tonumber(os.date("%H%M")) <= tonumber(string.format("%02d%02d", "23", "00")))

--)

--then



--Zapis do SQL mocy dziennej

local updatechart = Net.FHttp("192.168.1.13")

   if PvPowerWatt1 ~= 655 and PvPowerWatt1 < 4000 and PvPowerWatt1 >= 0 then

                  payload = "/data_post_stringi.php?id=" .. 11 .. "&value=" .. Prod_1_ToDay

                  fibaro:debug(payload)

                  response, status, errorCode = updatechart:GET(payload);

   end

   if PvPowerWatt2 ~= 655 and PvPowerWatt2 < 4500 and PvPowerWatt2 >= 0 then

                  payload = "/data_post_stringi.php?id=" .. 12 .. "&value=" .. Prod_2_ToDay

                  fibaro:debug(payload)

                  response, status, errorCode = updatechart:GET(payload);

   end

   if PvPowerWatt3 ~= 655 and PvPowerWatt3 < 3000 and PvPowerWatt3 >= 0 then

     payload = "/data_post_stringi.php?id=" .. 13 .. "&value=" .. Prod_3_ToDay

                  fibaro:debug(payload)

                  response, status, errorCode = updatechart:GET(payload);

   end
```

```
    fibaro:debug("Zapisano produkcje stringów do MySQL")




  fibaro:log("Power updatet");
        --Wait for the next Loop (1000=1sec)
        fibaro:sleep(Wait*1000);
end
```