RS 485 Communication with ABL Wallbox eMH1 (built 2020 or later)

Hardware:

Raspberry Pi

Raspberry 1373331 Pi 3 Modell B+ Mainboard, 1 GB

USB / RS 485 Interface:

DSD TECH USB to RS485 RS422 Converter Adapter with FTDI FT232 Chip for Windows 10, 8, 7, XP and Mac OS X

---

https://smile.amazon.de/gp/product/B07B416CPK/ref=ppx_yo_dt_b_search_asin_title?currency=EUR&ie=UTF8&language=en_GB&psc=1

Price: €17.99 Delivery at no extra cost for Prime members

- Industrial grade USB to RS485 RS422 converter: Allows you to easily connect any RS485 or RS422 device directly to your computer's USB port.
- The main chip is FTDI FT232RL: ESD protection with built-in 15KV and TVS surge protection. It's enough for you to cope with different occasions.
- LED for RX TX and Power.
- Compatible with Windows 10, 7 (32/64bit) Vista 2008, XP, 2003, Mac etc.
- GUARANTEE: We offer a 12 month warranty for this USB to RS422 / RS485 converter. If you have any questions, please contact us and we will resolve your problem within 24 hours.

Driver: https://ftdichip.com/drivers/vcp-drivers/

Software:

MINIMALMODBUS
Python Code
Aus <https://minimalmodbus.readthedocs.io/en/stable/internalminimalmodbus.html>

Python Code

```
import ABLminimalmodbus
        # ABLmininmalmodbus…renamed and modified copy of minimalmodbus; stored in same path as
          example code
        # code modified in line 1407 fpr the case of no answer on RS485 from eMH1 minimalmodbus
        # to yield response [4095, 65535, 65535, 65535, 65535])
        # 1407:   if not answer:
        # 1408:         answer = ">01030A0FFFFFFFFFFFFFFFFFFEC\r\n"
        # 1409:    return answer
import serial
```

```
VOLTAGE  = 230 #V

def Initialize_Wallbox(ABL):

   ABL.serial.baudrate = 38400
   ABL.serial.parity = serial.PARITY_EVEN
   ABL.serial.bytesize = 8
   ABL.serial.stopbits = 1
   return

def Monitor_Wallbox(ABL):

   Power = 0
   i_Phase = 0
   I_soll = 0
   Statustext = "EVCC no answer"

   ABL.read_registers(15,5)                                    # Dummy call to wake up EVCC
   if (ABL.read_registers(15,5)== 65535): Statustext = "EVCC no answer"          # see comment above
   else:
      value_list = ABL.read_registers(46,5)
      Status = int(m.fmod(value_list[0],256))

      if (Status == 161): Statustext = "Wait for connection"
      if (Status == 177): Statustext = "Wait for enablement"
      if (Status == 178): Statustext = "Charging enabled"
      if (Status == 194): Statustext = "Charging"
      if (Status == 194):
         i = 2
         Current = 0
         while i <= 4:
            Current= float(value_list[i])/10          #current read out in A provided by eMH1 current sensors
            if Current > 1.0 : i_Phase = i_Phase + 1      # determine single, dual, triple phase charging
            Power = Power + Current * VOLTAGE         # VOLTAGE either fixed or read by other device
            i=i+1
         I_set = round((value_list[1]-144*256)/16.66667,1)  #set point of charging current
   Output  = [Power, Statustext, i_Phase, I_set]
   return Output


def Set_Current (I_Set)
   Value_List = Monitor_Wallbox(ABL)
   if Value_List[1] != "EVCC no answer":
      ABL.write_registers(20,[int(float(I_Set)*16.6667)]) #set setpoint for current
      Value_List = Monitor_Wallbox(ABL)
      Output = [True, Value_List[3]]
  else:
     Output = [False, 0.0]

#--- Main ---

# Initialize Modbus communication
```

```
ABL = ABLminimalmodbus.Instrument(serial_name, ABL_slave_address, ABLminimalmodbus.MODE_ASCII)
Initialize_Wallbox(ABL)

I_set = 6                                                    # example Set_point 6A
Value_List_1 = Set_Current(I_set)
Success = Value_List_1[0]
Current_Set_point = Value_List_1[1]

print("Set Current Succes:", Success, "Current Set Point: " round( Current_Set_point,2), " A")

Value_List_2 = Monitor_Wallbox (ABL)
Actual_Power = Value_List_2[0]
Statustext = Value_List_2[1]
Number_of_Charging_Phases = Value_List_2[2]
Current_Set_point = Value_List_2[3]

print("Status:", Statustext)
print("Actual Power:", round[Actual_Power,2)," kW")
print("Number of phases:", Number_of_Charging_Phases)
Current Set Point: " round( Current_Set_point,2), " A")

#---- End----
```

Author:
DI Dr. Peter Steinrück
Technology and Strategy Advisor

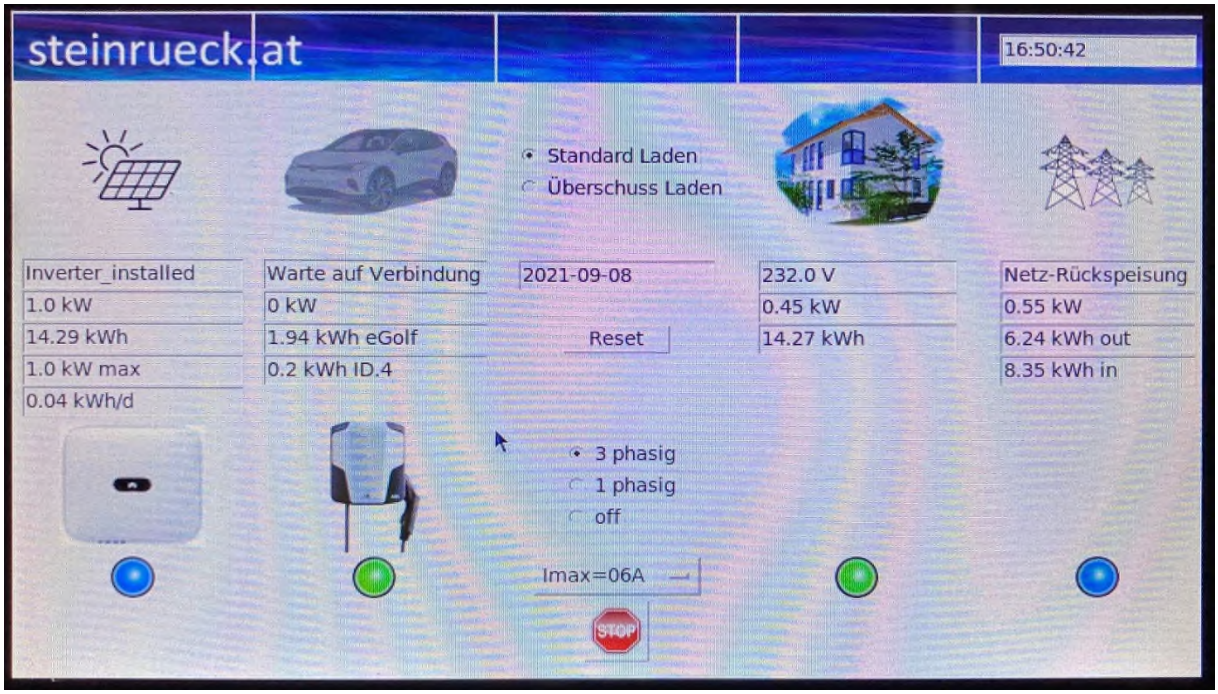
Leschetitzkygasse 55 Haus 32
1180 Vienna, Austria

Phone: +43 664 431 12234
eMail: psteinrueck@steinrueck.at

www.steinrueck.at

This code is part of a python script which has been developed for controlling and monitoring of PV charging of EVs. For “available power charging” English for German “Überschußladen” the PV produced power exceeding the household demand is used to charge the EV. To minimize losses to the grid the charging current tracks the available PV power. In case of too low PV supply charging is firstly cut back to single phase charging and secondly shut off.

The user interface is shown below:



In case you need more information please contact the author via e-Mail.