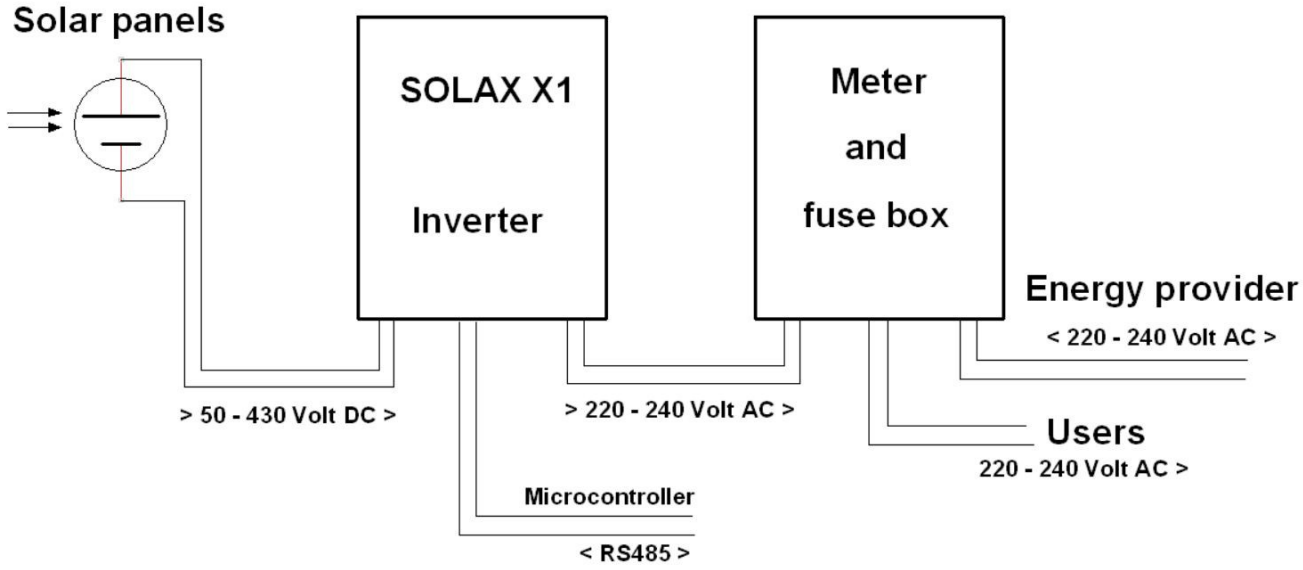
Getting data from a solar panel inverter

Purpose and setup

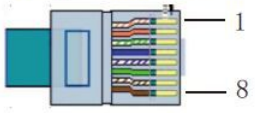
To use solar energy **direct or to store it in water,** some plug-in sockets of our power users are switched "on" or "off" by a programmable microcontroller. In our case we use an Arduino Uno and a Solax X1 inverter. Our boiler socket can be hard wired to the Arduino or used with a remote controlled (433MHz) socket.
To know **when** to switch the users, some of the internal data of the inverter is used.
My main interest is the delivered power in Watts and the temperature of the inverter to provide cooling with an external fan.



The inverter

To transform the 50–430 V DC from the solar panels to the 220-240 V AC of the power grid, a Solax X1 inverter is used. During the process heat is generated .
The internal data of the inverter can be reached via a RS485 communication port with a RJ 45 Jack.

The PIN definitions ofRS 485/Meter interface are as below.



PIN	1	2	3	4	5	6	7	8
Definition	X	Com/DRM0	GND_COM	Meter_A/ 485_A	Meter_B/ 485_B	E_Stop	RefGen	X

The RS485 data transfer is a differential system with two data lines A and B. (Pin 4 and 5 on the RJ45 plug) They are not grounded to be more resistant to interference. If line A is 5 volts, line B is 0 volts and vice versa. In our system, a MAX 485 module, connected to our Arduino microprocessor, converts the data stream into TTL with 5 volts relative to ground.

Communication protocol

The inverter uses a MODBUS protocol and has the be asked for it's data. The master (our Arduino microprocessor) sends a request for the data and the slave (The inverter) replies. The data packets contain a sender- and a destination- address.
The Solax protocol can be downloaded from the web and I used this version:
"SolaxPower_Single_Phase_External_Communication_Protocol_X1_V1.8.pdf"
In this protocol AP stands for access point: the master initiating the data transfer.

Getting the serial number of the inverter is a one time affair.
To retrieve the serial number of the inverter I define:

```
SoftwareSerial RS485Serial(SSerialRX, SSerialTX);
byte InByte = 0;
byte byteInput[25];
byte RequestSerialNumber[]=
{0xAA,0x55,0x01,0x00,  0x00,0x00, 0x10, 0x00, 0x00,  0x01,0x10};
//Header ,AccesPoint,   Solax X1 ,Contr, Func, Length, Checksum
```

and send the request:

```
 RS485Serial.write(RequestSerialNumber,sizeof(RequestSerialNumber));
```

And catch the response from the Inverter:

```
int index = 0;
 while(RS485Serial.available())
 {
  InByte = (byte)RS485Serial.read();
  byteInput[index] = InByte;
  index++;
 }
```

The response is used in the main controlling program to assign an address:

```
byte InByte = 0;
byte byteInput[12]; // the conformation send by the inverter is 12 bytes long
//                      Header   ,AccesPoint, Solax X1 ,Contr, Func, Length,
byte AddressInput[] = {0xAA,0x55,0x00,0x00, 0x00,0x00, 0x10,  0x01, 0x0F,
0x31,0x32,0x33,0x34,0x35,0x36,0x37,0x37,0x36,0x35,0x34,0x33,0x32,0x31,
0x0A,0x04,0x01};
// Serialnumber data 0 ------------------------------->13 ,address, checksum
```

In the setup part of the main program my AssignAddress() function is called and a conformation by the inverter received:

```
void AssignAddress()
{
 digitalWrite(REPin, RS485Transmit);
 digitalWrite(DEPin, RS485Transmit);
 RS485Serial.write(AddressInput,sizeof(AddressInput));
 delay(100);
 // * * * SWITCH TO RECEIVE MODE * * *
 digitalWrite(REPin, RS485Receive);
 digitalWrite(DEPin, RS485Receive);
 int index = 0;
 while(RS485Serial.available())
 {
  InByte = (byte)RS485Serial.read();
  ConformationInput[index] = InByte;
  index++;
 }
}
```

The address assignment has to be done each time the inverter has been off (at night). So I put the assignment in the setup part of the program in which I need inverter data.

Getting the data from the Inverter:

The RequestData[] array and DataInput are defined as:

```
byte RequestData[]={0xAA,0x55, 0x00,0x00, 0x00,0x0A , 0x11,  0x02, 0x00,  0x01,0x1C};
//                  Header   , Source   , SOLAX     ,Control,Func,Length, Check 2b
byte DataInput[63];
// the defined data to be updated by the inverter's data:
int temp_Solax = 33; // start with the fan running
int power_Solax = 0;
float voltage_Grid = 0.0;
float yield_Day = 0.0;
float yield_Total = 0.0;
long hours_Total = 0;
```

In the loop() of the main program I use a function to make my data requests and receive the data packet:

```
void GetSolaxData()
{
  digitalWrite(REPin, RS485Transmit); // set send mode
  digitalWrite(DEPin, RS485Transmit);
  RS485Serial.write(RequestData,sizeof(RequestData));
  digitalWrite(REPin, RS485Receive); // set receive mode
  digitalWrite(DEPin, RS485Receive);
  int index = 0;
```

```
  while(RS485Serial.available())
  {
    InByte = (byte)RS485Serial.read();
    DataInput[index] = InByte;
    index++;
  }
  // update the data values when there's output
  power_Solax = DataInput[28] + 256 * DataInput[27];
  if (power_Solax > 0)
  {
    temp_Solax = DataInput[10];
    voltage_Grid = (DataInput[24] + 256 * DataInput[23]) / 10.0;
    yield_Day  = (DataInput[12] + 256 * DataInput[11]) / 10.0;
    yield_Total = (DataInput[34] + 256 * DataInput[33] + 65536 * DataInput[32] +
      16777216 * DataInput[31]) / 10.0;
    hours_Total =  DataInput[38] + 256 * DataInput[37] + 65536 * DataInput[36] +
      16777216 * DataInput[35];
    Serial.println();
    Serial.print("Watts: ");
    Serial.print(power_Solax);
    Serial.println();
    Serial.print("Temperature: ");
    Serial.print(temp_Solax);
    Serial.println();
    Serial.print("gridVoltage: ");
    Serial.print(voltage_Grid);
    Serial.println();
    Serial.print("kWh this day: ");
    Serial.print(yield_Day);
    Serial.println();
    Serial.print("kWh total: ");
    Serial.print(yield_Total);
    Serial.println();
    Serial.print("Hours total: ");
    Serial.print(hours_Total);
    Serial.println();
  }
}
```

The data values are each 2 or 4 bytes HEX values. Some data fit in one byte like the temperature in DataInput[10].
The power is given in Watts and need both bytes. For example:
DataInput[27] = 0x06 = Decimal 6
DataInput[28] = 0xBF = Decimal 191
adds to a watts value of 6 x 256 + 191 = 1747 Watts. This is plenty to run my electric boiler of 1500 Watts so I can switch the socket of the boiler "on".

The complete program listings are written in a way that they provide info on your serial monitor while running.
They can easily be adopted to your own situation and needs.
https://www.bootprojecten.nl/solar-energy/getting-data-from-a-solar-panel-inverter

_250226_A_Solax_RequestSerialNumber
_250226_B_Solax_AssignAddress Sends address to Inverter and gets conformation
_250226_C_Solax_RequestData Shows the full data packet on your monitor

Important note: The Solax Inverter will lose its address assignment when "off" at night so in my controlling program below the assignment is done during the setup() of the program.
(So if you run **_250226_C_Solax_RequestData, _250226_B_Solax_AssignAddress** should have run the same day)

You can find more program examples and the last updated versions at:

https://www.bootprojecten.nl/solar-energy/solar-power-regulated-socket

https://www.bootprojecten.nl/solar-energy/solar-controlled-socket-shield-for-arduino

For feedback, questions or remarks you can mail in Dutch or English to:

Jeroen Droogh bootprojecten@gmail.com