

//_250406_Solax_SwitchFanAndSockets

// This program uses the data from the solar panel inverter to switch the sockets
// of power users "on" or "off" in such a way that the power of the solar panels
// is used directly during the time it is produced.
// The relay of the switchable boiler socket is hard wired to the microprocessor.
// Some other power users are plugged in remote controllerd sockets which are
// controole by a small 433MHz transmitter.
// To provide extra external cooling a fan is switched on at a certain temperature

// **Hardware connections:**

// Arduino MAX 485
// D5 DE Data Enable (DEPin)
// D4 RE Receive Enable(REPin)
// D3 RO Receive Out (SSerialRX)
// D6 DI Data In (SSerialTX)
// 3.3V VCC (5V also allowed)
// GND GND
// RS485 A SOLAX Inverter RJ45 pin 4
// RS485 B SOLAX inverter RJ45 pin 5
// D8 FanRelay Data
// 5V VCC of FanRelay,VCC of 433MHZ TX, Diode and LED of Boiler circuit
// GND GDD of FanRelay, Emittor, Pulldown resistors and GND 433MHZ TX
// D9 Basis of transistor of boiler circuit.
// D10 Mode-switch down Input 5V
// D11 Mode-switch up Input 5V
// D12 Data to 433MHz TX

// **Mode switch:**

// A 3-way switch let you choose from different CONDITIONS
// 3-WAY SWITCH UP: BOILER socket "on" independent on power_Solax
// one RC socket is switched "on", when there's excess power
// 3-WAY SWITCH DOWN: BOILER socket "off"
// The remote sockets are switched depending on the delivered power
// 3-WAY SWITCH MIDDLE position: BOILER "power regulated"
// one RC socket is switched "on", when there's excess power

```
#include <SoftwareSerial.h>
#define SSerialRX 3 // Receive Out
#define REPin 4 // Receive Enable
#define DEPin 5 // Data Enable
#define SSerialTX 6 // Data In
#define RS485Receive LOW
#define RS485Transmit HIGH
#define FanRelayPin 8
#define BoilerSocketPin 9
#define ModeAlwaysOffPin 10 // Input, switches 9 off when HIGH
#define ModeAlwaysOnPin 11 // Input, switches 5 on when HIGH
#define RCDatapin 12 // Data to the 433 MHz transmitter
SoftwareSerial RS485Serial(SSerialRX, SSerialTX);
```

```

byte AddressInput[] = {0xAA,0x55,0x00,0x00, 0x00,0x00, 0x10, 0x01, 0x0F,
//          Header, AccesPoint, Solax X1 ,Contr, Func, Length
    0x31,0x32,0x33,0x34,0x35,0x36,0x37,0x37,0x36,0x35,0x34,0x33,0x32,0x31, 0x0A,0x04,0x01};
// Serialnumber data 0 ----->13 ,address, checksum
byte ConformationInput[12]; // the conformation send by the inverter is 12 bytes long
byte RequestData[]={0xAA,0x55, 0x00,0x00, 0x00,0x0A , 0x11, 0x02, 0x00, 0x01,0x1C};
//          Header , Source , SOLAX ,Control,Func,Length, Check 2b
byte DataInput[63];

```

// Declaration and on-off settings (power in watts)

```

int power_Boiler_Off = 1550; //1475;
int power_Boiler_On  = 1600; //1525;
int power_RCA_Off = 550;
int power_RCA_On  = 600;
int power_RCB_Off = 1050;
int power_RCB_On  = 1100;
int power_RCC_Off = 1550;
int power_RCC_On  = 1600;
int power_BoilerPlusRC_Off = 2000;
int power_BoilerPlusRC_On  = 2050;
int temp_Fan_On = 32; // °C
int temp_Fan_Off = 27;
int temp_Solax = 33; // start with the fan running
long switchOffDelay = 2000; // millisecs delay before switching off
long loopdelay = 9000;

```

// Declarations and conditions

```

int MenuChoice = 2; // switch condition
int MenuChoice_old = 2;; // last switch condition
int power_Solax = 0; // data from inverter
float voltage_Grid = 0.0; // data from inverter
byte InByte = 0;
bool AddressAssigned = false;
bool boiler_allowedOn = false; // set by powervalue
bool socketA_allowedOn = false; // or switch
bool socketB_allowedOn = false;
bool socketC_allowedOn = false;
bool boiler_switchedOn = false; // runtime situation
bool socketA_switchedOn = false;
bool socketB_switchedOn = false;
bool socketC_switchedOn = false;

```

// Setting of the used RC switches

```

// delay times (pinData HIGH or LOW situations) in µ seconds:
#define Bit24HighUp 998 //44 samplepoints
#define Bit24HighDown 476 //21 samplepoints
#define Bit24LowUp 295 //13 samplepoints
#define Bit24LowDown 1179 //52 samplepoints

```

```
#define Bit32HighUp 1429 //66 samplepoints
#define Bit32HighDown 590 //26 samplepoints
#define Bit32LowUp 408 //18 samplepoints
#define Bit32LowDown 1610 //71 samplepoints
```

```
#define StartDelay24LowUp 400
#define StartDelay24LowDown 2250
#define StartDelay32LowUp 408
#define StartDelay32LowDown 7188
```

// commands for the 3 sockets; 24 bit pattern is send 6x, 32 bit pattern 4x

```
bool S24[6][24] = {{0,0,1,1,0,0,0,0,1,0,1,1,1,0,1,0,0,0,1,1,0,1,0,1}, // A_ON
    {0,0,1,1,0,1,1,1,0,0,0,0,1,1,0,1,0,1,0,0,0,1,0,1}, // A_OFF
    {0,0,1,1,1,0,1,1,1,1,0,0,0,0,1,1,0,1,1,0,1,1,0,0}, // B_ON
    {0,0,1,1,1,1,1,0,0,1,1,1,0,1,1,0,0,1,1,1,1,0,0}, // B_OFF
    {0,0,1,1,0,1,1,0,1,1,1,0,0,0,0,1,1,0,1,1,1,1,0}, // C_ON
    {0,0,1,1,0,1,0,1,0,1,1,0,1,0,0,0,0,0,0,1,1,1,0}}; // C_OFF
```

```
bool S32[6][32] =
    {{0,0,1,1,1,0,1,0,1,1,1,1,0,1,1,1,1,1,1,0,0,1,1,0,1,0,1,0,1}, // A_ON
    {1,0,1,1,1,1,0,0,1,1,1,0,1,1,1,0,0,1,1,1,1,0,0,1,0,1,0,1}, // A_OFF
    {1,1,1,1,0,0,0,1,0,1,0,1,1,1,1,1,1,1,1,0,1,0,1,1,0,1,0,0,1,1}, // B_ON
    {1,0,0,0,1,0,1,1,1,0,0,0,1,1,1,0,1,0,0,1,1,0,0,0,1,0,0,1,0,0,1,1}, // B_OFF
    {1,1,1,1,1,1,1,1,1,0,1,1,0,0,1,1,0,1,0,1,0,0,1,1,1,0,1,1,0,1,1}, // C_ON
    {1,1,0,1,0,0,1,1,0,0,1,0,0,0,0,1,1,0,0,0,0,0,0,1,1,0,0,1,1,0,1,1}}; // C_OFF
```

// Function to set the menu value according to the 3-way switch

void SetMenu()

```
{
    if(digitalRead(ModeAlwaysOnPin) == HIGH)
        MenuChoice = 1; //Boilersocket always on (UP)
    if(digitalRead(ModeAlwaysOffPin) == HIGH)
        MenuChoice = 3; //Boilersocket always off (DOWN)
    if((digitalRead(ModeAlwaysOnPin) == LOW) && (digitalRead(ModeAlwaysOffPin) == LOW))
        MenuChoice = 2; // Boilersocket power_Solax regulated
    Serial.println();
    Serial.print("ModeSwitch Socket On ");
    Serial.print(digitalRead(ModeAlwaysOnPin));
    Serial.println();
    Serial.print("ModeSwitch Socket Off ");
    Serial.print(digitalRead(ModeAlwaysOffPin));
    Serial.println();
}
```

// Function to assign address for Solax inverter

void AssignAddress()

```
{
    digitalWrite(REPin, RS485Transmit);
    digitalWrite(DEPin, RS485Transmit);
    RS485Serial.write(AddressInput, sizeof(AddressInput));
    delay(100);
}
```

```

digitalWrite(REPin, RS485Receive); // RECEIVE MODE * *
digitalWrite(DEPin, RS485Receive);
int index = 0;
while(RS485Serial.available())
{
    InByte = (byte)RS485Serial.read();
    ConformationInput[index] = InByte;
    index++;
}
if(ConformationInput[9] == 6)
    AddressAssigned = true;
}

```

// Function to get the data packet from the inverter

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 of interest
    power_Solax = DataInput[28] + 256 * DataInput[27];
    if (power_Solax > 0)
        temp_Solax = DataInput[10];
    voltage_Grid = (DataInput[24] + 256 * DataInput[23]) / 10.0;
    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();
}

```

void switchRCsockets(bool withBoiler)

```

{
    if (withBoiler == false) // switch the RC sockets while boiler off
    {

```

```

if(power_Solax >= power_RCA_On)
{
    socketA_allowedOn = true;
    if(socketA_switchedOn == false)
    {
        switchRC(0); // Socket A ON
        socketA_switchedOn = true;
    }
}
if((power_Solax < power_RCA_Off) && (socketA_switchedOn == true))
{
    delay(switchOffDelay); // delay and measure again
    GetSolaxData();
    if(power_Solax < power_RCA_Off)
    {
        socketA_allowedOn = false;
        switchRC(1); // Socket A OFF
        socketA_switchedOn = false;
    }
}
if(power_Solax >= power_RCB_On)
{
    socketB_allowedOn = true;
    if(socketB_switchedOn == false)
    {
        switchRC(2); // Socket B ON
        socketB_switchedOn = true;
    }
}
if((power_Solax < power_RCB_Off) && (socketB_switchedOn == true))
{
    delay(switchOffDelay); // delay and measure again
    GetSolaxData();
    if(power_Solax < power_RCB_Off)
    {
        socketB_allowedOn = false;
        switchRC(3); // Socket B OFF
        socketB_switchedOn = false;
    }
}
if(power_Solax >= power_RCC_On)
{
    socketC_allowedOn = true;
    if(socketC_switchedOn == false)
    {
        switchRC(4); // Socket C ON
        socketC_switchedOn = true;
    }
}
}

```

```

if((power_Solax < power_RCC_Off) && (socketC_switchedOn == true))
{
    delay(switchOffDelay); // delay and measure again
    GetSolaxData();
    if(power_Solax < power_RCC_Off)
    {
        socketC_allowedOn = false;
        switchRC(5); // Socket C OFF
        socketC_switchedOn = false;
    }
}
}
if(withBoiler == true) // switch the RC sockets while boiler on
{
    if(power_Solax >= power_BoilerPlusRC_On)
    {
        socketA_allowedOn = true;
        if(socketA_switchedOn == false)
        {
            switchRC(0); // Socket A ON
            socketA_switchedOn = true;
        }
    }
    if((power_Solax < power_BoilerPlusRC_Off) && (socketA_switchedOn == true))
    {
        delay(switchOffDelay); // delay and measure again
        GetSolaxData();
        if(power_Solax < power_BoilerPlusRC_Off)
        {
            socketA_allowedOn = false;
            switchRC(1); // Socket A OFF
            socketA_switchedOn = false;
        }
    }
}
Serial.println();
Serial.print(" boiler_switchedOn = ");
Serial.print(boiler_switchedOn);
Serial.println();
Serial.print(" socketA_switchedOn = ");
Serial.print(socketA_switchedOn);
Serial.println();
Serial.print(" socketB_switchedOn = ");
Serial.print(socketB_switchedOn);
Serial.println();
Serial.print(" socketC_switchedOn = ");
Serial.print(socketC_switchedOn);
Serial.println();
}

```

```

// Function to send the TX signal dependend on socket and command
void switchRC(int commandNumber)
{
    int bitNumber;
    int bitPattern;
    Serial.println();
    Serial.print("Function switchRC called. Commandnumber: ");
    Serial.print(commandNumber);

    for (bitPattern = 0; bitPattern < 6; bitPattern++) // 24 bit part 6 times
    {
        // start with extra LOW part
        digitalWrite(RCDataPin, HIGH);
        delayMicroseconds(StartDelay24LowUp);
        digitalWrite (RCDataPin, LOW);
        delayMicroseconds(StartDelay24LowDown);
        for (bitNumber = 0; bitNumber < 24; bitNumber++)
        {
            if (S24[commandNumber][bitNumber] == HIGH)
            {
                digitalWrite (RCDataPin, HIGH);
                delayMicroseconds(Bit24HighUp);
                digitalWrite (RCDataPin, LOW);
                delayMicroseconds(Bit24HighDown);
            }
            else
            {
                digitalWrite (RCDataPin, HIGH);
                delayMicroseconds(Bit24LowUp);
                digitalWrite (RCDataPin, LOW);
                delayMicroseconds(Bit24LowDown);
            }
        }
    }
    for (bitPattern = 0; bitPattern < 4; bitPattern++) // 32bits part 4 times
    {
        //start with an extra long LOW
        digitalWrite (RCDataPin, HIGH);
        delayMicroseconds(StartDelay32LowUp);
        digitalWrite (RCDataPin, LOW);
        delayMicroseconds(StartDelay32LowDown);
        for (bitNumber = 0; bitNumber < 32; bitNumber++)
        {
            if (S32[commandNumber][bitNumber] == HIGH)
            {
                digitalWrite (RCDataPin, HIGH);
                delayMicroseconds(Bit32HighUp);
                digitalWrite (RCDataPin, LOW);
                delayMicroseconds(Bit32HighDown);
            }
        }
    }
}

```

```

    }
    else
    {
        digitalWrite (RCDDataPin, HIGH);
        delayMicroseconds(Bit32LowUp);
        digitalWrite (RCDDataPin, LOW);
        delayMicroseconds(Bit32LowDown);
    }
}
}
}
}

```

void setup()

```

{
    pinMode(REPin, OUTPUT);
    pinMode(DEPin, OUTPUT);
    pinMode(SSerialRX, INPUT);
    pinMode(SSerialTX, OUTPUT);
    pinMode(FanRelayPin, OUTPUT);
    pinMode(BoilerSocketPin, OUTPUT); // to relay on the socket
    pinMode(ModeAlwaysOnPin, INPUT);
    pinMode(ModeAlwaysOffPin, INPUT);
    pinMode(RCDDataPin, OUTPUT);
    Serial.begin(9600);
    Serial.println("_250403_Solax_SwitchFanAndSockets");
    Serial.println(" Assigning address to Inverter and waiting for first data....");
    Serial.println();
    digitalWrite(REPin, RS485Receive);
    digitalWrite(DEPin, RS485Receive);
    digitalWrite(FanRelayPin, HIGH); // Fan off
    RS485Serial.begin(9600); // set data rate
    AssignAddress();
    delay(1000);
    if(AddressAssigned == true)
        Serial.println("Address Assigned ");
}

```

void loop()

```

{
    SetMenu();          // Check the menu switch
    delay(200);
    GetSolaxData();     // get the datapacket from the inverter
    delay(200);

    if (temp_Solax >= temp_Fan_On)
        digitalWrite(FanRelayPin, LOW); // Fan on
    if ((temp_Solax < temp_Fan_Off) && (power_Solax > 0))
        digitalWrite(FanRelayPin, HIGH); // Fan off
}

```

```

// * * Switch the sockets * *
if(MenuChoice == 1) // * * 3-Way switch Up Boilersocket powered
{
  boiler_allowedOn = true;
  digitalWrite(BoilerSocketPin,HIGH);
  boiler_switchedOn = true;
  switchRCsockets(boiler_switchedOn);
}
if(MenuChoice == 3) // * * 3-Way switch Down Boilersocket off
{
  boiler_allowedOn = false;
  digitalWrite(BoilerSocketPin,LOW);
  boiler_switchedOn =false;
  switchRCsockets(boiler_switchedOn);
}
if(MenuChoice == 2) // * * 3-Way switch neutral Boiler Solar regulated
{
  if(power_Solax >= power_Boiler_On)
  {
    boiler_allowedOn = true;
    if(boiler_switchedOn == false)
    {
      digitalWrite(BoilerSocketPin, HIGH);
      boiler_switchedOn = true;
    }
  }
  if((power_Solax < power_Boiler_Off) && (boiler_switchedOn == true))
  {
    delay(switchOffDelay); // delay and measure again
    GetSolaxData();
    if(power_Solax < power_Boiler_Off)
    {
      boiler_allowedOn = false;
      digitalWrite(BoilerSocketPin, LOW);
      boiler_switchedOn = false;
    }
  }
  switchRCsockets(boiler_switchedOn);
}
delay(loopdelay);
}

```

// Sketch uses 7802 bytes (24%) of program storage space.
 // Global variables use 1111 bytes (54%) of dynamic memory.