//_250418_Solax_SwitchFanAndRCSockets

```
// This program uses the data from the solar panel inverter to switch RC 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 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
// a green LED will indicate the use of the inverter's data.
// a blue LED will indicate that enough power is delivered for RCA (boiler socket)

// Hardware connections:
// Arduino   MAX 485
// D4       DE  Data Enable   (DEPin)
// D3       RE  Receive Enable(REPin)
// D2       RO  Receive Out   (SSerialRX)
// D5       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
// D7        Green LED to indicate datareception from inverter > 50 W
// D8        To FET switching the fan
// D9        To Blue LED: Boiler socket "on"
// 5V        VCC of 433MHZ TX, LED of Boiler circuit, MODE switch centre pin
// GND       GDD of FanFET, Emittor, Pulldown resistors and GND 433MHZ TX
// 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 RCA "on" independent on power_Solax
//      RCB socket is switched "on", when there's excess power
// 3-WAY SWITCH DOWN: BOILER socket RCA "off"
//      RCB and RCC are switched depending on the deliverded power
// 3-WAY SWITCH MIDDLE position: BOILER RCA "power regulated"
//      RCB socket is switched "on", when there's excess power

#include <SoftwareSerial.h>
#define SSerialRX 2    // Receive Out
#define REPin 3        // Receive Enable
#define DEPin 4        // Data Enable
#define SSerialTX 5    // Data In
#define RS485Receive LOW
#define RS485Transmit HIGH
#define SolaxDatalLED 7 // indicator Solax provides data
#define FanRelayPin 8
#define BoilerOnPin 9
#define ModeAlwaysOffPin 10 // Input, set menu choice 3 when HIGH
#define ModeAlwaysOnPin 11  // Input, set menu choice 1 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 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_RCA_Off = 1550; // 1475 winter boiler socket
int power_RCA_On  = 1600; // 1525 winter
int power_RCBPlus_Off = 2000; // when used in mode 2
int power_RCBPlus_On  = 2050; // when RCA is on
int power_RCB_Off = 550; // when used in mode 3
int power_RCB_On  = 600; // when RCA is off
int power_RCC_Off = 1050; // mode 3 only
int power_RCC_On  = 1100;
int temp_Fan_On = 31; // ºC
int temp_Fan_Off = 27;
int temp_Solax = 30; // start with the fan runnung
int switchOffDelay = 2000; // millisecs delay before switching off
int loopdelay = 9000; // 9 seconds
```

// Declarations and conditions

```
int MenuChoice = 2; // switch condition
int power_Solax = 0; // data from inverter
float voltage_Grid = 0.0; // data from inverter
byte InByte = 0;
bool socketA_allowedOn = false; // set by powervalue or switch
bool socketB_allowedOn = false;
bool socketC_allowedOn = false;
//bool AddressAssigned = false;

bool socketA_switchedOn = false; // runtime situation
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
                   {0,0,1,1,1,1,1,0,0,1,1,1,0,1,1,0,0,1,1,1,1,1,0,0}, // B_OFF
                   {0,0,1,1,0,1,1,0,1,1,1,0,0,0,0,1,1,0,1,1,1,1,1,0}, // C_ON
                   {0,0,1,1,0,1,0,1,0,1,1,0,1,0,0,0,0,0,0,1,1,1,1,0}};// C_OFF
bool S32[6][32] =
  {{0,0,1,1,1,0,1,0,1,1,1,1,0,1,1,1,1,1,1,1,1,1,1,0,0,1,1,0,1,0,1,0,1,0,1}, // A_ON
   {1,0,1,1,1,1,1,0,0,1,1,1,0,1,1,1,0,0,1,1,0,1,1,1,1,1,1,0,0,1,0,1,0,1}, // A_OFF
   {1,1,1,1,1,0,0,0,1,0,1,0,1,1,1,1,1,1,1,1,1,1,1,0,1,0,1,1,0,1,0,0,1,1}, // B_ON
   {1,0,0,0,1,0,1,1,1,0,0,0,1,1,1,0,1,0,0,1,1,0,0,0,1,0,0,1,0,0,1,0,1,1}, // B_OFF
   {1,1,1,1,1,1,1,1,1,1,0,1,1,0,0,1,1,0,1,0,1,0,0,1,1,1,1,0,1,1,1,0,1,1}, // 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 an address to the Solax inverter
void AssignAddress()
{
  digitalWrite(REPin, RS485Transmit);
  digitalWrite(DEPin, RS485Transmit);
  RS485Serial.write(AddressInput,sizeof(AddressInput));
  delay(100);
  digitalWrite(REPin, RS485Receive);
  digitalWrite(DEPin, RS485Receive);
  delay(1000);
}
```

```
void GetSolaxData()
{
 digitalWrite(REPin, RS485Transmit);
 digitalWrite(DEPin, RS485Transmit);
 RS485Serial.write(RequestData,sizeof(RequestData));
 digitalWrite(REPin, RS485Receive);
 digitalWrite(DEPin, RS485Receive);
 int index = 0; // get the data packet:
 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(int mode)
{
 if (mode == 1) // RCA(Boiler)always on
 {
  if(socketA_switchedOn == false)
  {
   switchRC(0); // Socket A ON
   socketA_switchedOn = true;
   digitalWrite(BoilerOnPin, HIGH);  // Blue LED on
  }
  if(power_Solax >= power_RCBPlus_On)
  {
   socketB_allowedOn = true;
   if(socketB_switchedOn == false)
   {
    switchRC(2); // Socket B ON
    socketB_switchedOn = true;
   }
  }
  if((power_Solax < power_RCBPlus_Off) && (socketB_switchedOn == true))
  {
```

```
    delay(switchOffDelay); // delay and measure again
    GetSolaxData();
    if(power_Solax < power_RCBPlus_Off)
    {
     socketB_allowedOn = false;
     switchRC(3); // Socket B OFF
     socketB_switchedOn = false;
    }
   }
}
if (mode == 2) // all sockets power_Solax regulated
{
  if(power_Solax >= power_RCA_On)
  {
   socketA_allowedOn = true;
   if(socketA_switchedOn == false)
   {
    switchRC(0); // Socket A ON
    socketA_switchedOn = true;
    digitalWrite(BoilerOnPin, HIGH);  // Blue LED on
   }
  }
  if((power_Solax < power_RCA_Off) && (socketA_switchedOn == true))
  {
   delay(switchOffDelay); // delay and meassure again
   GetSolaxData();
   if(power_Solax < power_RCA_Off)
   {
    socketA_allowedOn = false;
    switchRC(1); // Socket A OFF
    socketA_switchedOn = false;
    digitalWrite(BoilerOnPin, LOW); // Blue LED off
   }
  }
  if(power_Solax >= power_RCBPlus_On)
  {
   socketB_allowedOn = true;
   if(socketB_switchedOn == false)
   {
    switchRC(2); // Socket B ON
    socketB_switchedOn = true;
   }
  }
  if((power_Solax < power_RCBPlus_Off) && (socketB_switchedOn == true))
  {
   delay(switchOffDelay); // delay and measure again
   GetSolaxData();
   if(power_Solax < power_RCBPlus_Off)
   {
    socketB_allowedOn = false;
    switchRC(3); // Socket B OFF
    socketB_switchedOn = false;
```

```
    }
   }
  }
if(mode == 3)  // RCA (boiler) off switch B and C earlier
{
 if(socketA_switchedOn == true)
 {
  switchRC(1); //Socket A OFF
  socketA_switchedOn == false;
  digitalWrite(BoilerOnPin, LOW);
 }
 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;
  }
 }
}
```

```
  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 depending 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 (RCDataPin, HIGH);
      delayMicroseconds(Bit32LowUp);
      digitalWrite (RCDataPin, LOW);
      delayMicroseconds(Bit32LowDown);
     }
    }
  }
}


void setup()
{
  pinMode(REPin, OUTPUT);
  pinMode(DEPin, OUTPUT);
  pinMode(SSerialRX, INPUT);
  pinMode(SSerialTX, OUTPUT);
  pinMode(FanRelayPin, OUTPUT);
  pinMode(ModeAlwaysOnPin, INPUT);
  pinMode(ModeAlwaysOffPin, INPUT);
  pinMode(RCDataPin,OUTPUT);
  pinMode(SolaxDatalLED, OUTPUT);
  pinMode(BoilerOnPin, OUTPUT);
  Serial.begin(9600);
  Serial.println("_250418_Solax_SwitchFanAndRCSockets");
  Serial.println();
  digitalWrite(REPin, RS485Receive);
  digitalWrite(DEPin, RS485Receive);
  digitalWrite(FanRelayPin, LOW); // Fan off
  digitalWrite(SolaxDatalLED, LOW);
  digitalWrite(BoilerOnPin, LOW); // Blue LED off
  RS485Serial.begin(9600); // set data rate
  Serial.println();
  Serial.println(" Assigning address to Inverter....");
  AssignAddress();
  delay(1000);
  GetSolaxData();
  delay(200);
}
```

```
void loop()
{
  GetSolaxData();
  delay(200);
  if(power_Solax == 0) // re-assign address when needed
  {
    delay(30000);
    GetSolaxData(); // check power_Solax again
    if(power_Solax == 0)
    {
      Serial.println();
      Serial.println("Assigning address again....");
      AssignAddress();
      delay(1000);
    }
  }
  SetMenu();          // Check the menu switch
  delay(200);
  if(power_Solax > 50) // Indicate data input with green LED
    digitalWrite(SolaxDatalLED, HIGH);
  if(power_Solax < 10)
    digitalWrite(SolaxDatalLED, LOW);
  if (temp_Solax >= temp_Fan_On)
    digitalWrite(FanRelayPin, HIGH); // Fan on
  if ((temp_Solax < temp_Fan_Off) && (power_Solax > 0))
    digitalWrite(FanRelayPin, LOW); // Fan off
  switchRCsockets(MenuChoice);
  delay(loopdelay);
}
// Sketch uses 7782 bytes (24%) of program storage space.
// Global variables use 1059 bytes (51%) of dynamic memory.
```