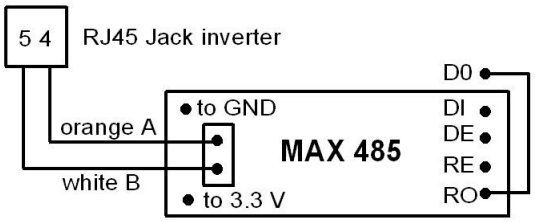
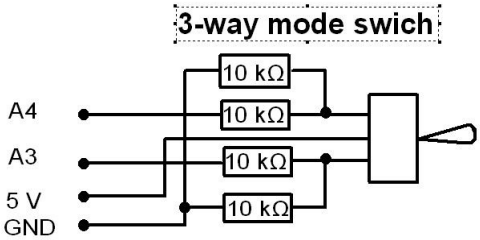
// **_250810A_Solax_ClockStarted_SocketSwitchAndLog**

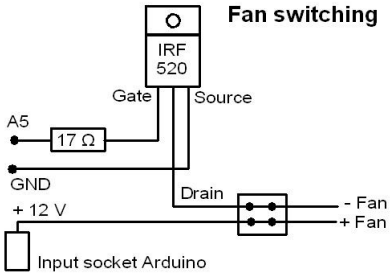
Inverter connection



// **Arduino > MAX 485**
// D0 < RX < RO receive out
// D1 > TX > DI data in
// D2 > MAX_DEpin > DE data enable
// D3 > MAX_REpin > RE receive enable
// DE and RE are LOW in receive mode

// RS485 A > orange > RJ45 pin 4 SOLAX Inverter
// RS485 B > white > RJ45 pin 5 SOLAX inverter
// 3.3 V > yellow > MAX VCC (Voltage Common Collector)
// GND > black > MAX GND
// **Arduino** **> Real Time Clock module DS 1302** CR2032 battery
// D5 RTC_CEpin > white > CE Chip Enable
// D6 > orange > SCLK Clock signal
// D7 > brown > I/O data
// 5V block > yellow > VCC
// GND block > black > GND
// **Arduino** > **Voltmeter displaying temperature**
// D8 Display_VCC > red > VCC meter (on/off time controlled)
// D9 Displ_value > yellow > Data meter
// GND block > black > GND meter
// **Arduino** > **SDCard logger module**
// D10 SD_CS > brown > CS Chip Select
// D11 > red > MOSI Master Out Slave In
// D12 > grey > MISO Master In Slave Out
// D13 > purple > CLK Clock signal
// GND > grey > GND
// 5V > yellow > VCC
// A2 GREEN_Log > 1kΩ > **green LED** on SD GND pin D
// D4 BLUE_SD_RC1> 1kΩ > **blue LED** on SD GND pin
// **Arduino** > **433MHz TX module** on headerpins A0 and A1
// A0 RC_TX_VCC > VCC
// A1 RC_TX_DAT > DAT
// GND header > blue > GND
// **Arduino** **> 3-way switch**
// A3 Mode_ONpin > blue > 3-way switch up
// A4 Mode_OFFpin > brown> switch down
// 5V > yellow > middle pin
// GND > black > pulldown resistors





// A5 FanPin > 17Ω > IRF520 Gate
// GND > IRF520 Source
// IRF520 Drain > connection block > Fan -
// +12V input socket > red > connection block > Fan +

```
#include <SPI.h>                // used for SD
#include <SD.h>
#include "DS1302.h"             // real time clock module
#include <SoftwareSerial.h>

#define RC_TX_VCC   A0
#define RC_TX_DAT   A1
#define GREEN_Log   A2
#define Mode_ONpin  A3
#define Mode_OFFpin A4
#define FanPin      A5

#define MAX_SSerialRX 0
#define MAX_SSerialTX 1
#define MAX_DEpin     2
#define MAX_REpin     3
#define BLUE_SD_RC1   4
#define RTC_CEpin     5
#define RTC_IOpin     6
#define RTC_SLKpin    7
#define Display_VCC   8
#define Display_value 9
#define SD_CS         10

SoftwareSerial RS485Serial(MAX_SSerialRX, MAX_SSerialTX);
DS1302 rtc(RTC_CEpin, RTC_IOpin, RTC_SLKpin);
Time t = rtc.time();
File logfile;

// Declarations for inverter data retrieving
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];
byte InByte = 0;
bool dataValid = false;

// Inverter/Sockets on-off settings and declarations
byte hour_SolaxStart = 9; // recording and socketswitching
byte hour_SolaxEnd  = 21; // end daytime/ RCsocket switching
int power_RCA_Off = 1450; // boiler 1385 (power in watts)
int power_RCA_On  = 1550; // measured 250614
int power_RCBPlus_Off = 1950; // when RCA is on
int power_RCBPlus_On  = 2050; // when RCA is on
int power_RCB_Off =  650; // when RCA is off
int power_RCB_On  =  750; // when RCA is off
int power_RCC_Off = 1150; // when RCA is off & RCB on
int power_RCC_On  = 1250; // when RCA is off & RCB on
```

```
int power_Solax   =    0;        // data from inverter
int  voltvalue = 0;
byte temp_Fan_On  = 35;      // ºC
byte temp_Fan_Off = 33;
byte temp_Solax    = 0;
int switchdelay = 6000;         // millisecs delay before switching
int loopdelay   = 9000;         // 9 seconds
byte MenuChoice = 2;            // switch condition
bool daytime = false;           // monitorring inverter off
bool socketA_allowedOn  = false; // set by powervalue inverter or
bool socketB_allowedOn  = false; // 3-way menuchoice switch
bool socketC_allowedOn  = false;
bool socketA_switchedOn = false; // runtime situation
bool socketB_switchedOn = false;
bool socketC_switchedOn = false;
```

// **RC code setting for 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
                   {0,0,1,1,1,1,1,1,0,0,1,1,1,0,1,1,0,0,1,1,1,1,1,0,0},// 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] =
  {{0,0,1,1,1,0,1,0,1,1,1,1,0,1,1,1,1,1,1,1,1,1,0,0,1,1,0,1,0,1,0,1},// 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
   {1,0,0,0,1,0,1,1,1,0,0,0,1,1,1,0,1,0,1,0,0,1,1,0,0,0,1,0,0,1,0,0,1,1},// B_OFF
   {1,1,1,1,1,1,1,1,1,1,1,0,1,1,0,0,1,1,0,1,0,1,0,1,0,0,1,1,1,1,0,1,1,0,1,1},// C_ON
   {1,1,0,1,0,0,1,1,0,0,1,0,0,0,0,1,1,0,0,0,0,0,0,0,1,1,0,0,1,1,0,1,1}};// C_OFF
```

// **Function to send the TX signal dependend on socket and command**

```
void switchRC(byte commandNumber)
{
  byte bitNumber;
```

```
  byte bitPattern;
  for (bitPattern = 0; bitPattern < 6; bitPattern++)  // 24 bit part 6 times
  {
   // start with extra LOW part
   digitalWrite(RC_TX_DAT, HIGH);
   delayMicroseconds(StartDelay24LowUp);
   digitalWrite (RC_TX_DAT, LOW);
   delayMicroseconds(StartDelay24LowDown);
   for (bitNumber = 0; bitNumber < 24; bitNumber++)
   {
    if (S24[commandNumber][bitNumber] == HIGH)
    {
     digitalWrite (RC_TX_DAT, HIGH);
     delayMicroseconds(Bit24HighUp);
     digitalWrite (RC_TX_DAT, LOW);
     delayMicroseconds(Bit24HighDown);
    }
    else
    {
     digitalWrite (RC_TX_DAT, HIGH);
     delayMicroseconds(Bit24LowUp);
     digitalWrite (RC_TX_DAT, LOW);
     delayMicroseconds(Bit24LowDown);
    }
   }
  }
  for (bitPattern = 0; bitPattern < 4; bitPattern++)  // 32bits part 4 times
  {
   //start with  an extra long LOW
   digitalWrite (RC_TX_DAT, HIGH);
   delayMicroseconds(StartDelay32LowUp);
   digitalWrite (RC_TX_DAT, LOW);
   delayMicroseconds(StartDelay32LowDown);
   for (bitNumber = 0; bitNumber < 32; bitNumber++)
   {
    if (S32[commandNumber][bitNumber] == HIGH)
    {
     digitalWrite (RC_TX_DAT, HIGH);
     delayMicroseconds(Bit32HighUp);
     digitalWrite (RC_TX_DAT, LOW);
     delayMicroseconds(Bit32HighDown);
    }
    else
    {
     digitalWrite (RC_TX_DAT, HIGH);
     delayMicroseconds(Bit32LowUp);
     digitalWrite (RC_TX_DAT, LOW);
     delayMicroseconds(Bit32LowDown);
    }
   }
  }
}
```

```
void SetMenu()        // check the 3 way switch
{
  if(analogRead(Mode_ONpin) > 500) // (read values 0-1024)
    MenuChoice = 1; //Boilersocket always on (UP)
  if(analogRead(Mode_OFFpin) > 500)
    MenuChoice = 3; //Boilersocket always off (DOWN)
  if((analogRead(Mode_ONpin) < 500) && (analogRead(Mode_OFFpin) < 500))
    MenuChoice = 2; // Boilersocket power_Solax regulated
}

// Function to assign an address to the Solax inverter
void AssignAddress()
{
  digitalWrite(MAX_REpin, HIGH); // MAX 485 in send mode
  digitalWrite(MAX_DEpin, HIGH);
  RS485Serial.write(AddressInput,sizeof(AddressInput));
  delay(10000);
}

void setSwitchConditions(int mode)
{
  // set conditions
  if (mode == 1) // RCA(Boiler)always on
  {
    socketA_allowedOn = true;
    socketB_allowedOn = false;
    socketC_allowedOn = false;
    if(power_Solax >= power_RCBPlus_On)
      socketB_allowedOn = true;
    if((power_Solax < power_RCBPlus_Off) && (socketB_allowedOn == true))
    {
      delay(switchdelay); // delay and meassure again
      GetSolaxData();
      if(power_Solax < power_RCBPlus_Off)
        socketB_allowedOn = false;
    }
  }
  if (mode == 2) // all sockets power_Solax regulated
  {
    if(power_Solax >= power_RCA_On)
    {
      socketA_allowedOn = true;
      socketB_allowedOn = false;
      socketC_allowedOn = false;
    }
    if(power_Solax >= power_RCBPlus_On)
    {
      socketA_allowedOn = true;
      socketB_allowedOn = true;
      socketC_allowedOn = false;
    }
    if((power_Solax < power_RCBPlus_Off) && (socketB_allowedOn == true))
```

```
  {
   delay(switchdelay); // delay and meassure again
   GetSolaxData();
   if(power_Solax < power_RCBPlus_Off)
     socketB_allowedOn = false;
  }
  if((power_Solax >= power_RCB_On) && (power_Solax <= power_RCA_Off))
  {
   socketA_allowedOn = false;
   socketB_allowedOn = true;
   socketC_allowedOn = false;
  }
  if((power_Solax >= power_RCC_On) && (power_Solax <= power_RCA_Off))
  {
   socketA_allowedOn = false;
   socketB_allowedOn = true;
   socketC_allowedOn = true;
  }
  if((power_Solax < power_RCA_Off) && (socketA_allowedOn == true))
  {
   delay(switchdelay); // delay and meassure again
   GetSolaxData();
   if(power_Solax < power_RCA_Off)
     socketA_allowedOn = false;
  }
  if((power_Solax < power_RCC_Off) && (socketC_allowedOn == true))
  {
   delay(switchdelay); // delay and meassure again
   GetSolaxData();
   if(power_Solax < power_RCC_Off)
     socketC_allowedOn = false;
  }
  if((power_Solax < power_RCB_Off) && (socketB_allowedOn == true))
  {
   delay(switchdelay); // delay and meassure again
   GetSolaxData();
   if(power_Solax < power_RCB_Off)
     socketB_allowedOn = false;
  }
}
if(mode == 3)  // RCA (boiler) off switch B and C
{
  socketA_allowedOn = false;
  if(power_Solax >= power_RCB_On)
    socketB_allowedOn = true;
  if((power_Solax < power_RCB_Off) && (socketB_allowedOn == true))
  {
   delay(switchdelay); // delay and meassure again
   GetSolaxData();
   if(power_Solax < power_RCB_Off)
     socketB_allowedOn = false;
  }
```

```
    if(power_Solax >= power_RCC_On)
      socketC_allowedOn = true;
    if((power_Solax < power_RCC_Off) && (socketC_allowedOn == true))
    {
      delay(switchdelay); // delay and meassure again
      GetSolaxData();
      if(power_Solax < power_RCC_Off)
        socketC_allowedOn = false;
    }
  }
}
void switchRC_Sockets()
{
  if((socketA_allowedOn == true) && (socketA_switchedOn == false))
  {
    switchRC(0); //Switch Socket A ON
    digitalWrite(BLUE_SD_RC1, HIGH);
    switchRC(5); //Switch Socket C OFF because it switches ON with A
    socketA_switchedOn = true;
  }
  if((socketB_allowedOn == true) && (socketB_switchedOn == false))
  {
    switchRC(2); //Switch Socket B ON
    socketB_switchedOn = true;
  }
  if((socketC_allowedOn == true) && (socketC_switchedOn == false))
  {
    switchRC(4); //Switch Socket C ON
    socketC_switchedOn = true;
  }
  if((socketA_allowedOn == false) && (socketA_switchedOn == true))
  {
    switchRC(1); //Switch Socket A off
    digitalWrite(BLUE_SD_RC1, LOW);
    socketA_switchedOn = false;
  }
  if((socketC_allowedOn == false) && (socketC_switchedOn == true))
  {
    switchRC(5); //Switch Socket C Off
    socketC_switchedOn = false;
  }
  if((socketB_allowedOn == false) && (socketB_switchedOn == true))
  {
    switchRC(3); //Switch Socket B Off
    socketB_switchedOn = false;
  }
}
```

```
void GetSolaxData()
{
  byte index = 0;
  power_Solax = 0;
  dataValid = false;
  digitalWrite(MAX_REpin, HIGH); // MAX 485 in send mode
  digitalWrite(MAX_DEpin, HIGH);
  delay(200);
  RS485Serial.write(RequestData,sizeof(RequestData));
  digitalWrite(MAX_REpin, LOW); // MAX 485 in receive mode
  digitalWrite(MAX_DEpin, LOW);
  delay(500);
  while(RS485Serial.available())
  {
    InByte = (byte)RS485Serial.read();
    DataInput[index] = InByte;
    index++;
  }   // update the data values of interest and voltmeter(temp)
  power_Solax = DataInput[28] + 256 * DataInput[27];
  if((power_Solax > 0) && (power_Solax < 4000))
  {
    temp_Solax = DataInput[10];
    voltvalue = temp_Solax * 5.1;
    if (voltvalue >= 255)
      voltvalue -= 255;
    analogWrite(Display_value,voltvalue);
    dataValid = true;
    digitalWrite(Display_VCC,HIGH); // Display on
  }
  if(dataValid == false)
    digitalWrite(Display_VCC,LOW); // Display off
}
void(* resetFunc) (void) = 0; // used for daily reset

void LogData()
{
  // log time, watts, yield_day, yield total and temp
  char timestring[5];
  sprintf(timestring,"%02d:%02d",t.hr,t.min);
  logfile.print(timestring + String("     "));
  sprintf(timestring,"");
  sprintf(timestring,"%04d",power_Solax);
  logfile.print(timestring + String("    "));
  // convert yieldvalue parts first to kWh to stay within 4 bytes data-limit of float
  float yieldtoday = 25.6 * DataInput[11] + DataInput[12]/10.0;
  logfile.print(yieldtoday + String("     "));
  yieldtoday = DataInput[34]/10.0 + 25.6 * DataInput[33] + 6553.6 * DataInput[32] +
       1677721.6 * DataInput[31];  // total yield of the inverter
  logfile.print(yieldtoday + String("       "));
  logfile.println(String(temp_Solax));
  logfile.flush();
}
```

```
void setup()
{
 RS485Serial.begin(9600); // set data rate
 dataValid = false;
 daytime   = false;
 pinMode(RC_TX_VCC,     OUTPUT); // A0
 pinMode(RC_TX_DAT,     OUTPUT); // A1
 pinMode(GREEN_Log,     OUTPUT); // A2
 pinMode(Mode_ONpin,    INPUT);  // A3
 pinMode(Mode_OFFpin,   INPUT);  // A4
 pinMode(FanPin,        OUTPUT); // A5
 pinMode(MAX_SSerialRX, INPUT);  // DO RO ReceiveOut
 pinMode(MAX_SSerialTX, OUTPUT); // D1 DI DataIn
 pinMode(MAX_DEpin,     OUTPUT); // D2 DE Data Enable
 pinMode(MAX_REpin,     OUTPUT); // D3 RE Receive Enable
 pinMode(BLUE_SD_RC1,   OUTPUT); // D4
 pinMode(Display_VCC,   OUTPUT); // D8
 pinMode(Display_value, OUTPUT); // D9
 pinMode(SD_CS,         OUTPUT); // D10 Chip select
 digitalWrite(MAX_REpin,   LOW); // receive mode
 digitalWrite(MAX_DEpin,   LOW);
 digitalWrite(FanPin,      LOW); // Fan off
 delay(2000);
 digitalWrite(BLUE_SD_RC1, LOW); // LED off
 digitalWrite(GREEN_Log,   LOW); // LED off
 digitalWrite(Display_VCC, LOW); // Display off
 digitalWrite(RC_TX_VCC,  HIGH); // TX RCSockets on
 switchRC(1);
 switchRC(3);
 switchRC(5);

 Time t = rtc.time();
 delay(500);
 char logname[12];
 sprintf(logname,"Solx%02d%02d.TXT", t.yr-2000, t.mon);
 if (!SD.begin(SD_CS)) // Initializing SD card..
 {
  digitalWrite(BLUE_SD_RC1, HIGH); // Card failed/ not present
  return;
 }
 logfile = SD.open(logname, FILE_WRITE);
 if(!logfile)
  digitalWrite(BLUE_SD_RC1, HIGH); // Couldn't create file
 logfile.println();
 logfile.println(logname);
 logfile.println();
 logfile.print("Time      Watt   kWh/d" + String(t.date));
 logfile.println("     kWh          °C");
 logfile.flush();
 delay(2000);
 AssignAddress();
}
```

```
void loop()
{
 t = rtc.time();
 delay(200);
 // * * * * 10 minutes before start * * * *
 if((t.hr == ( hour_SolaxStart - 1 )) && (t.min == 50))
 {
  if ((t.sec >= 0) && (t.sec < 10))
  {
   delay(60000); // be sure to startup for the day only once
   resetFunc();
  }
 }
 t = rtc.time();
 delay(200);
 // * * * *  daytime (start- to endtime) * * * *
 if ((t.hr >= hour_SolaxStart) && (t.hr < hour_SolaxEnd))
 {
  daytime = true;
  GetSolaxData();
  if(dataValid == true)
  {
   digitalWrite(Display_VCC,HIGH); // Display on
   if(DataInput[10] >= temp_Fan_On)
    digitalWrite(FanPin, HIGH); // Fan on
   if(DataInput[10] < temp_Fan_Off)
    digitalWrite(FanPin, LOW); // Fan off
   SetMenu(); // check the 3-way menu switch
   delay(200);
   setSwitchConditions(MenuChoice);
   switchRC_Sockets(); //switch the sockets if neccesary
   t = rtc.time();
   if((t.sec >= 0) && (t.sec < 10)) // once a minute
   {
    GetSolaxData(); // refresh data and temp readout
    if(dataValid == true)
    {   // log every 10 minutes
     if ((t.min ==  0) || (t.min == 10) ||
         (t.min == 20) || (t.min == 30) ||
         (t.min == 40) || (t.min == 50) )
     {
      // Show green LED to allow visual check of RTC timeshift
      digitalWrite(GREEN_Log, HIGH); // Green LED on
      LogData(); // write data on SD
      delay(5000);
      digitalWrite(GREEN_Log, LOW); // Green LED off
     }
     delay(9000); // to be sure to log once every 10 minutes
    }
   }
  }
 }
```

```
  // * * * * "night"time * * * *
  if (t.hr >= hour_SolaxEnd)
  {
    if (daytime == true)
    {
      dataValid = false;
      digitalWrite(FanPin, LOW);
      digitalWrite(GREEN_Log, LOW);
      digitalWrite(Display_VCC,LOW);
      digitalWrite(RC_TX_VCC,LOW);
      daytime - false;
    }
    delay(loopdelay);   // extra delay at night
  }
  delay(loopdelay);
}
/*
Sketch uses 21372 bytes (66%) of program storage space.
Global variables use 1522 bytes (74%) of dynamic memory, leaving 527 bytes for local variables
Sample Datalogging for day 20 of month 8:

Solx2508.TXT

Time      Watt   kWh/d20     kWh         °C
09:00     0299    0.20     10116.00      31
09:10     0237    0.20     10116.00      31
09:20     0222    0.30     10116.10      32
09:30     0356    0.30     10116.10      32
09:40     0469    0.40     10116.20      33
09:50     0603    0.50     10116.30      34
10:00     0507    0.50     10116.30      32
10:10     0406    0.60     10116.40      34
10:20     0615    0.70     10116.50      34
10:30     0737    0.80     10116.60      33
10:40     0935    0.90     10116.70      34
10:50     0522    1.10     10116.90      33
11:00     0600    1.20     10117.00      34
11:10     1203    1.30     10117.10      33
11:20     1471    1.40     10117.20      33
11:30     0543    1.50     10117.30      33
11:40     1475    1.70     10117.50      35
12:10     1553    2.20     10118.00      37
12:30     1734    2.70     10118.50      39
12:40     1745    3.00     10118.80      41
12:50     2053    3.20     10119.00      40
13:00     2021    3.50     10119.30      42
13:10     2050    3.80     10119.60      43
13:20     0542    4.00     10119.80      41
13:30     0336    4.10     10119.90      36 etc
(Time-Power-YieldDay20-TotalYieldInverter-TemperatureInverter)
*/
```